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

Bole Chang*
Linxin Xie*
1221193017@fafu.edu.cn
225420010@fzu.edu.cn
Fujian Agriculture And Forestry University
Fuzhou University
Fuzhou, China

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 becomes a fundamental issue in bipartite network analysis and has drawn a lot of attention recently.

However, a fundamental barrier is the exponential blowup 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 thresholds. We also present an unbiased online sampling method to take full advantage of shadows and minimize the 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 138X speedup on average, with up to 500X speedup, which indicates that Z-Shadow is suitable for large scale bipartite graphs, and it is memory saving, efficient and accurate.

CCS CONCEPTS

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

KEYWORDS

Bicliques, sampling, Füredi's Theorem

ACM Reference Format:
Bole Chang, Linxin Xie, Wei Li, Meng Qin, and Jianfeng Hou. 2024. Z-shadow: An Efficient Method for Estimating Bicliques in Massive Graphs Using Füredi's Theorem. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

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 [18]. 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 disjoint vertex sets 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 $K_{3,3}$. Comparatively, cliques in general graphs and bicliques are fundamental to identifying community structures in bipartite graphs. Therefore, biclique counting has become an essential aspect of network analysis, with applications ranging from graph neural networks (GNNs) and dense subgraph detection to cohesive subgraph analysis and other fields [37].

Despite the importance of biclique counting, existing methods are often limited by their reliance on iterative search techniques, which lead to combinatorial explosion as the number of bicliques increases exponentially in large, dense graphs. This rapid growth in computational complexity makes these methods unsuitable for massive real-world graphs. For example, Yang et al. introduced a 2-hop graph construction method to optimize the BCList algorithm, which is called BCList++, improving time efficiency by reducing the cost of finding 2-hop neighbors. However, this method does not address the core issue of redundant iterations and the extensive search space, limiting its scalability for large graphs. While some research has focused on exact biclique counting, approaches that prioritize scalability by trading off some precision for computational efficiency remain underexplored. Certainly, there are also some good studies. Zigzag[38] proposed by Ye et al has excellent performance

*Both authors contributed equally to this research.
†Corresponding author

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/XXXXXXX.XXXXXXX

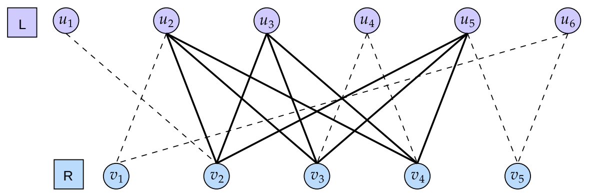


Figure 1: $K_{3,3}$ in a bipartite graph $G = (L, R, E)$

both in terms of efficiency and in terms of efficiency, but the data set it mainly targets is still sparse graph, and the method it uses is based on the nature of clique itself, so it is difficult to extend to other pattern estimation.

This research gap highlights the need for new methods that balance accuracy and efficiency by addressing the problem of search space explosion. Approaches leveraging graph properties, probabilistic techniques, or approximation algorithms may offer promising solutions. To this end, we propose Z-Shadow, an efficient method for estimating bicliques. Z-Shadow works by mapping target bicliques to smaller ones, reducing the task of counting bicliques in large graphs to estimating counts in smaller subgraphs. By decomposing the graph into smaller components and pruning irrelevant sections, the accuracy of the estimation is improved. For instance, consider selecting 3 vertices uniformly at random from set L and 3 vertices from set R to form a vertex subset A in the bipartite graph shown in Figure 1. The probability that the induced subgraph $G[A]$ forms a biclique is $\frac{1}{200}$. In contrast, if 3 vertices are selected uniformly at random from L and 2 vertices from R to form a vertex subset A in the bipartite graph shown in Figure 5, the probability that the induced subgraph $H[A]$ forms a biclique increases to $\frac{1}{3}$. We also incorporate the Füredi theorem to terminate the iteration early for sufficiently dense subgraphs, thereby reducing the search space and ensuring theoretical accuracy.

Our main contributions are as follows:

Integrating Extremal Graph Theory into Bipartite Graph Algorithms. Motivated by the k-clique counting randomized algorithm proposed in [15], our algorithm utilizes results for the Zarankiewicz problem in extremal graph theory. Seminal results by Füredi [12] 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 cannot be measured in terms of time and space complexity, but it can be seen in numeric experiments.

Efficient and Memory-Saving Sampling. The shadow construction algorithm presented in [15] is an algorithm with sufficient theoretical basis to construct the sample set. However, it requires significant memory to store shadows for a sparse bipartite graph. To overcome this obstacle, Jain and Seshadhri proposed an online algorithm [14] to address this issue. However, it is not the most space-efficient. In order to substantially reduce memory usage, we modified the online algorithm [14] with an unbiased sampling method. Even more, we release the space storing shadows as soon as they are fully utilized. **Online sampling** we proposed can reduce memory usage by up to 1500 times compared to **Z-Shadow(Gloabl)**.

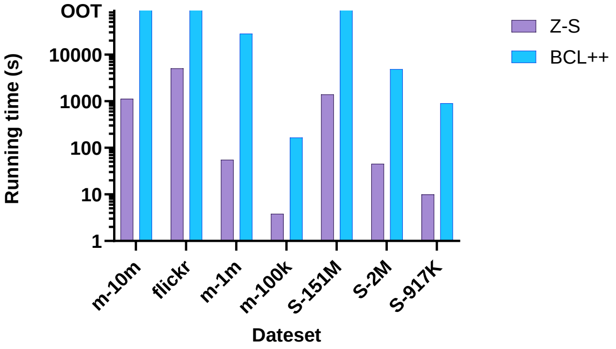


Figure 2: For $K_{4,4}$, the running time was quantitatively evaluated in seconds on seven datasets. If the runtime exceeds 24 hours, it is recorded as OOT.

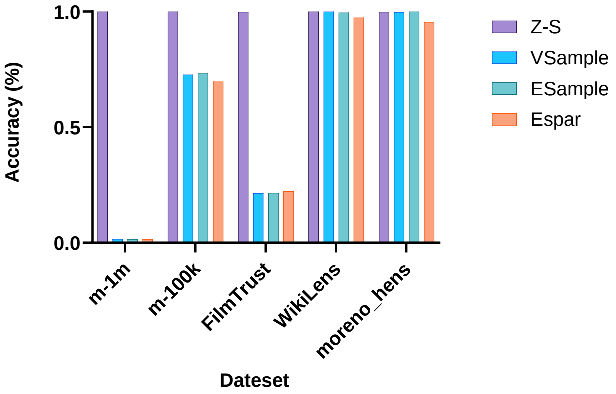


Figure 3: Quantitative evaluation of accuracy in terms of percent(%) for $K_{4,4}$. The baseline here is the BCL++ calculation.

Efficient for Massive Graphs. The existing biclique counting algorithm, BCList [37], is designed for sparse bipartite graphs by listing each subgraph. However, when dealing with dense bipartite graphs, this approach is computationally intensive and slow. Our Z-Shadow randomized algorithm, based on a theoretical threshold for sampling, is suitable for dense bipartite graphs. Then numerical experiments on dense datasets show that the Z-Shadow algorithm is typically 500 times faster than BCL++ [37]. For example, it takes only twenty minutes for $(4, 4)$-biclique estimation on a dataset with 151 million edges, whereas BCL++ takes more than 24 hours. Moreover, the variations of graph parameters have less impact on the speed of our algorithm. Figure 2 shows the time consumption of Z-Shadow and BCl++ for $s = 4, t = 4$.

Outstanding Accuracy. Our algorithm delivers accurate results with low variance. Figure 3 shows the accuracy of Z-S compared with other approximation algorithms conducted on five datasets. The errors of our algorithm are consistently below 0.5%, 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. In Section 3, we present definitions and notation from graph theory. In Section 4, we propose a randomized biclique estimation algorithm for dense and large bipartite graphs, along with theoretical analysis. In Section 5, we optimize our algorithm by reordering graph vertices and modifying the online method to address memory usage issues. Finally, Section 6 presents several numerical experiments.

2 RELATED WORK

Research in motif counting has traditionally focused on k-clique counting and butterfly (or $(2, 2)$-biclique) counting. The k-clique problem has received significant attention due to its importance in various network-based applications, such as community detection [11, 40], graph partitioning [25, 13], network embedding [39, 26], and recommendation systems [24, 29]. Existing research on k-cliques primarily focuses on exact and approximate counting. Exact counting algorithms, such as those proposed in [7, 21, 28, 16, 32], generally rely on exhaustive search tree traversals for each vertex to identify k-cliques. In contrast, approximation methods, particularly sampling techniques [17, 6, 34, 4], aim to reduce computational costs by trading off some precision.

In 2017, Jain and Seshadhri developed a fast approximation algorithm for clique counting using extremal combinatorial theory [15], currently one of the fastest available methods for this task. They further extended this approach to near-clique estimation in 2020 [14], proposing an online sampling algorithm to handle large graphs more efficiently.

The simplest biclique, the $(2, 2)$-biclique or "butterfly," has also been widely studied as a fundamental motif in bipartite graphs. Sanei-Mehri et al. proposed a fast butterfly counting algorithm in 2018 [27], sparking significant interest in accelerated and parallel approaches for $(2, 2)$-biclique counting across both CPU and GPU architectures [22, 36, 31, 33]. Additionally, this problem has been extended to applications such as (α, β)-core queries [5, 23] and fraud detection [30, 41].

However, counting larger bicliques, such as (s, t)-bicliques, presents a new challenge. Yang et al. [37] developed the BCList algorithm, which relies on depth-first exploration for counting (s, t)-bicliques in sparse bipartite graphs. This method was later extended with a layer-based search optimization in BCList++. Nevertheless, BCList++ struggles with dense and massive graphs, severely limiting its scalability. Although the estimation algorithm Zigzag [38] proposed by Ye et al. has excellent performance, it also focuses its work on sparse graphs. Despite the importance of biclique counting, no current method fully addresses the limitations in large-scale, dense bipartite graph scenarios.

3 PRELIMINARIES

This section outlines the key 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. In this paper, only simple bipartite graphs without loops and multiple edges are considered. Let $e(G) = |E(G)| = |E|$ 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, denote 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 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 copies of $K_{s,t}$ in $G[S]$ with s vertices in L and the t vertices in R, and let $\mathcal{N}(K_{s,t}, G) = \mathcal{N}(K_{s,t}, L \cup R)$. The problem of estimating bicliques in bipartite graphs can be formally defined as follows:

Problem 2. *Given a bipartite graph $G = (L, R, E)$ and parameters s, t, we study the problem of estimating the number of copies of (s, t)-bicliques with s vertices in L and the t vertices in R.*

This problem is challenging due to the combinatorial explosion of potential subgraphs in large and dense bipartite graphs.

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. This maximum is denoted by $\text{ex}(n, H)$. The Erdős-Stone Theorem gives an asymptotic formula for $\text{ex}(n, H)$ 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 $\text{ex}(n, H) = O(n^{2-\delta})$.

Turán type problems forbidding bipartite graphs are often called *degenerate.* Determining the growth rate of $\text{ex}(n, H)$ 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 [9, 3], asks for the exponent of $\text{ex}(n, C_{2k})$ is open for every k not in $\{2, 3, 5\}$ (see e.g. [10, 2, 35, 19, 20]).

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.* 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 [12], Füredi gave an upper bound of $Z(m, n, s, t)$, on which our algorithm relied.

Theorem 3 (Füredi(1996) [12]). *For positive numbers $m \geq s, n \geq t, s \geq t \geq 2$,*

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

Definition 4. *Let $G = (L, R, E)$ be a bipartite graph. A shadow **S** for $K_{s,t}$ is a multi-set of tuples (S, q) for some $1 \leq q \leq t - 1$ and*

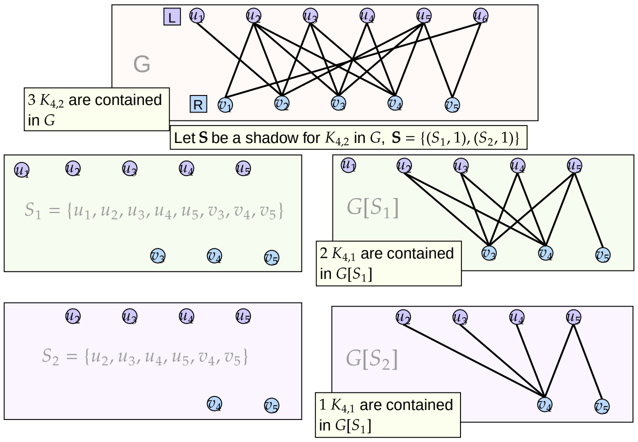

L
u_1 u_2 u_3 u_4 u_5 u_6
G
3 $K_{4,2}$ are contained in G
R
v_1 v_2 v_3 v_4 v_5
Let S be a shadow for $K_{4,2}$ in G, $\mathbf{S} = \{(S_1,1),(S_2,1)\}$
$S_1 = \{u_1,u_2,u_3,u_4,u_5,v_3,v_4,v_5\}$
$G[S_1]$
2 $K_{4,1}$ are contained in $G[S_1]$
$S_2 = \{u_2,u_3,u_4,u_5,v_4,v_5\}$
$G[S_2]$
1 $K_{4,1}$ are contained in $G[S_1]$

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

$S \subseteq (L \cup R)$ 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 **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 3. Let $|S_L| = m, |S_R| = n$. 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 can be used to detect whether a subgraph definitely contains $K_{s,q}$. If the number of edges of the subgraph $G[S]$ exceeds the threshold function, it can be judged that the subgraph must contain $K_{s,q}$, and this subgraph can enter the sampling section. In the sample stage, we pick a set of vertices of size $s + q$ for each element $(S, q) \in \mathbf{S}$ and check whether it forms a (s, q)-biclique to estimate the number of (s, q)-bicliques in $G[S]$.

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 $S \subseteq L \cup R$, let $S_L = S \cap L$ and $S_R = S \cap R$ for simplicity. For 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 5 shows a concrete example of these notations.

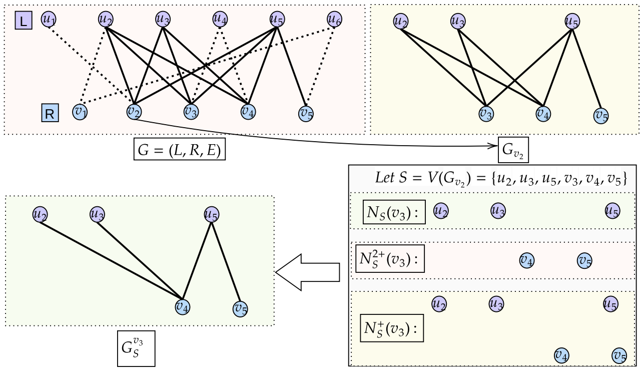

L
u_1 u_2 u_3 u_4 u_5 u_6
R
v_1 v_2 v_3 v_4 v_5
$G = (L, R, E)$
G_{v_2}
Let $S = V(G_{v_2}) = \{u_2, u_3, u_5, v_3, v_4, v_5\}$
$N_S(v_3):$
$N_S^{2+}(v_3):$
$N_S^+(v_3):$
$G_S^{v_3}$

Figure 5: Specific examples of notations mentioned in section 4.1

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

```
Input: A bipartite graph $G = (L, R, E)$, integer $s, t$
Output: A shadow $\mathbf{S}$ 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 $\mathbf{S}$
10:             else
11:                 Add $(N_S^+(v_i), q-1)$ to $\mathbf{T}$
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 $\mathbf{S}$
16:             else
17:                 Add $(N_S^+(v_i), 2)$ to $\mathbf{T}$
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 $\mathbf{S}$
```

Z-Shadow-Finder is the shadow construction process, whose goal is to get a reasonable set of samples. Since the alpha function can only handle the case of $s > t$, if the input $s < t$, you need to swap the L and R parts of the graph and swap s, t (line 1-2). **T** is the sample set to be verified. If there is a tuple (S, q), and $G[S]$ must contain $K_{s,q}$, the tuple would be added to S (line 3), otherwise enter T or break. Each tuple in **T** needs to iterate down to get new tuples, and the original tuple will be deleted (line 4-5). For $(S, q) \in \mathbf{T}$, Only if $d_S(v_i) > s, v_i \in S_R$, it is possible for $G_S^{v_i}$ to contain $K_{s,q-1}$ (line 6). q is divided into three cases in the algorithm:

Case 1: q>3 The threshold function is used to determine whether to enter S (line 7-11).

Case 2: q=3 The threshold function cannot play a good role in this case, because the threshold function is very relaxed on the small

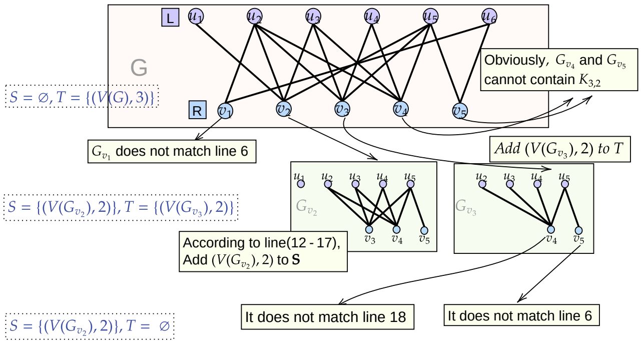

L
R
G
u_1 u_2 u_3 u_4 u_5 u_6
v_1 v_2 v_3 v_4 v_5
$S = \varnothing, T = \{(V(G), 3)\}$
Obviously, G_{v_4} and G_{v_5} cannot contain $K_{3,2}$
G_{v_1} does not match line 6
Add $(V(G_{v_3}), 2)$ to T
$S = \{(V(G_{v_2}), 2)\}, T = \{(V(G_{v_3}), 2)\}$
G_{v_2}
G_{v_3}
According to line(12 - 17), Add $(V(G_{v_2}), 2)$ to **S**
It does not match line 18
It does not match line 6
$S = \{(V(G_{v_2}), 2)\}, T = \varnothing$

Figure 6: An example of a specific implementation of Z-Shadow-Finder $(G, 3, 3)$

graph. Therefore, the method of selecting vertices to directly verify whether a biclique is formed is adopted (line 12-17).

Case 3: q=2 It is only necessary to contain $K_{s,1}$ in the induced subgraph (line 18-19).

Fig.6 shows an example of constructing a shadow of $K_{3,3}$ in G.

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

Theorem 5. *Let **S** be the shadow of $K_{s,t}$ constructed in **Z-Shadow-Finder**. Then, for any $(S, q) \in$ **S**, $G[S]$ contains a copy of $K_{s,q}$ and $\sum_{(S,q)\in \mathbf{S}} \mathcal{N}(K_{s,q}, S) = \mathcal{N}(K_{s,t}, G)$.*

Proof. The existence of $K_{s,q}$ in $G[S]$ is guaranteed by Theorem 3 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 of t-part in B have been ordered. In **Z-Shadow-Finder**, we check the vertices of t-part in B one by one in this order according to the threshold function $\alpha(\cdot)$ and q, and output a unique $(S, q) \in$ 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 (s, t)-bicliques in a bipartite graph, it suffices to count the number of $K_{s,q}$ in each (S, q) in the shadow of $K_{s,t}$ constructed in **Z-Shadow-Finder** by Theorem 5. 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 slightly, which are shown in Algorithm 2 and 3.

Sample (S) samples the tuple of the given **S** through the set theoretical probability distribution, and then extracts certain vertices from the vertices set in the tuple to verify whether biclique is formed. **Z-Shadow(Global)** forms a complete biclique estimation algorithm by calling **Z-Shadow-Finder** and **Sample**.

Theorem 6. *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 **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 6 is given in Appendix A.1.

Algorithm 2 Sample (S)

```
Input: A shadow S constructing by Z-Shadow-Finder
Output: A set of vertices $A$
For each $(S, q) \in \mathbf{S}$, set $\omega(S) = \binom{|S_L|}{s}\binom{|S_R|}{q}$ and $p(S) = \omega(S)/\sum_{(S',q)\in \mathbf{S}} \omega(S')$
Sample $(S, q)$ independently from S with the probability $p(S)$
Uniformly at random select $s$ 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 $\tau$
Output: An estimated value $\hat{K} = \frac{\sum_i X_i}{\tau} \sum_{(S',q)\in \mathbf{S}} \omega(S')$ for $K$
S $\leftarrow$ Z-Shadow-Finder$(G, s, t)$
for $i = 1, 2, \ldots, \tau$ do
    $A \leftarrow$ Sample(S)
    if $G[A]$ is an $(s, q)$-biclique then
        $X_i = 1$
    else
        $X_i = 0$
```

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-1 array, which can be ignored.

Coordinate format (COO), which is a sparse matrix storage format, is also used in our experiments. It is said that the COO format is the predecessor of the CSR. In 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 7. *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 8. *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 $v \in R'$, $d_H(u) \geq \alpha$ and $d_H(v) \geq \beta$. 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 9. *For $i \geq 1$, $|\mathbf{T}_{t-i}| \leq |R'|(\Delta_R^{2+})^{i-1}$.*

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 $|\mathbf{T}_{t-i}| \leq |\mathbf{T}_{t-i+1}|\Delta_R^{2+} \leq |R'|(\Delta_R^{2+})^{i-1}$. □

Corollary 10. $|\mathbf{T}| \leq |R'|\sum_{i=1}^{t-1}(\Delta_R^{2+})^{i-1}$.

Theorem 11. *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) has 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 7, 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,

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

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

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

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

This completes the proof.

□

Theorem 12. *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 improve the concentration 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 11 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. The related information is shown in Table 1. From observation, the number of edges in G_S^v is much smaller than $\alpha(N_S^+(v), 3, 4)$. By Theorem 14, 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 $d(v_1) \leq d(v_2) \cdots \leq d(v_{|R|})$, i.e. reorder in ascending order of degree. Take Figure 1 for example, the graph after reordering is shown as Figure 13.

There are two benefits to this:

- Assume that $u, v \in R$ and $u \in N_G^{2+}(v)$, G_S^v is likely to be a dense graph and it is easy to achieve the upper bound $\alpha(G_S^v, s, q)$.
- 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 $\mathbf{T}$.

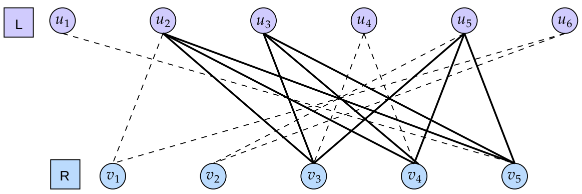


Figure 7: An example for graph reordering

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

5.2 Online Sampling

In Algorithm 3, 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 form dealing with the large graphs. To solve this problem, we propose an online sampling algorithm, inspired by the method in [14].

During online sampling, we set 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 $\Phi = \sum_{v \in R} \Phi_v$. In Algorithm 4, vertex sampling sequences P (line 6) can determine whether a shadow can be reused or abandoned. The online shadow construction method with an unbiased sampling substantially reduces the memory usage and releases the space storing shadows as soon as they are fully utilized. Moreover, the following two results guarantee the effect of our algorithm.

Theorem 13. *$G = (L, R, E)$ 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 14. *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 $\tau \geq \frac{3\Phi}{\epsilon^2 K} \ln \frac{2}{\delta}$, 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.

Theorem 15. *The memory complexity of **Online Sampling** is $O(\Delta_L \Delta_R (\Delta_R^{2+})^{t-2})$.*

Algorithm 4 Online Sampling

```
Input: A bipartite graph $G = (L, R, E)$ and integer $s, t, \tau$
Output: An estimate $\hat{K} = \frac{W}{\tau}\Phi$ for K
 1: Order $R$ by degree
 2: $W = 0$
 3: 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 $\Phi = \sum_{v \in R} \Phi_v$
 4: Independently sample $\tau$ vertices in $R$, say $w_1, \cdots, w_\tau$, with
    probability $p(w_i)$
 5: for $i = 1, 2, \ldots, \tau$ do
 6:     $v \leftarrow w_i$
 7:     if $w_i \neq w_{i-1}$ then
 8:         $\mathbf{S} \leftarrow$ Z-Shadow-Finder $(G_v, s, t-1)$
 9:     Let $\omega_v = \sum_{(S,q) \in \mathbf{S}} \binom{|S_L|}{p}\binom{|S_R|}{q}$
10:     if Sample($\mathbf{S}$) is a biclique then
11:         Set $X_i = \omega_v / \Phi_v$
12:     else
13:         Set $X_i = 0$
14:     $W = W + X_i$
15:     if $w_{i+1} \neq w_i$ then
16:         Delete $\mathbf{S}$
```

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. We will make the datasets and demo code public 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.

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

- **Z-S(AD)**: **Online Sampling** proposed in Section 5.2,which adopts ascending order of degree.
- **Z-S(R)**: **Online Sampling** in Section 5.2 adopts random ordering.
- **BCL**: The baseline method proposed in [37].
- **BCL++**: The state-of-the-art biclique counting algorithm proposed in [37].

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

Table 2: Basic properties of the dataset

dataset	$\lvert E\rvert$	$\lvert L\rvert$	$\lvert R\rvert$	$\overline{d}$	$\overline{d_L}$	$\overline{d_R}$	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

- **PMBE**: Adapted algorithm from maximal biclique enumeration proposed in [1].
- **VSample**: A vertex-based estimation algorithm designed concerning section 5.1 in [27].
- **ESample**: A edge-based estimation algorithm designed concerning section 5.2 in [27].
- **ESpar**: An estimation algorithm based on edge sparsification, which is designed concerning section 6.1 in [27].

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

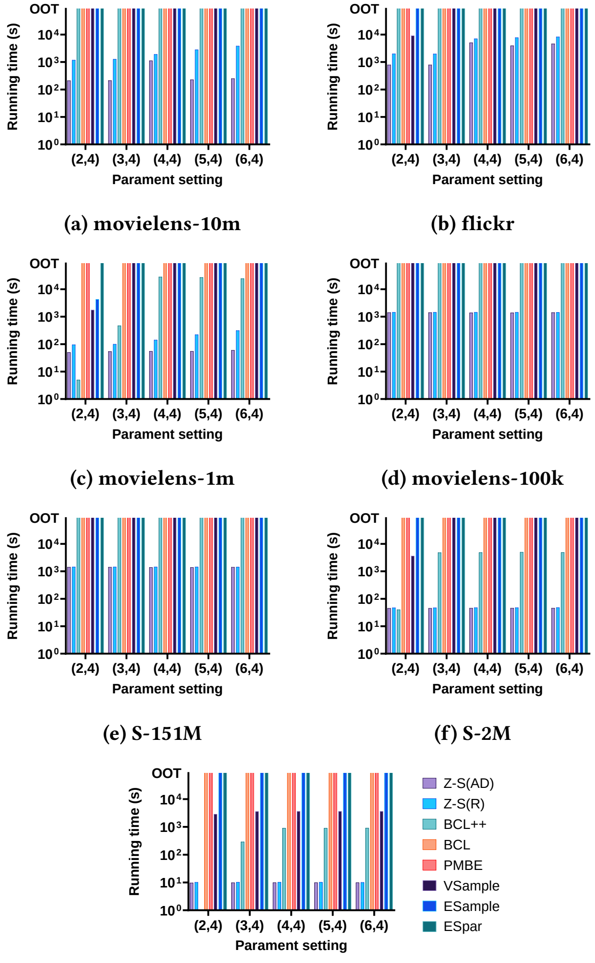
```
Input: A bipartite graph $G = (L, R, E)$
Output: An estimated value $\hat{K}$ of $K$
for $i = 1, \cdots, \tau$ do
    Choose an edge $e = (u, v)$ from $E$ uniformly at random, which $u \in L, v \in R$.
    $N_{>v}(u) = \{w | w \in N(u) \land w > v\}$,  $G[N^+(e)] = G[N_{>v}(u) \cup N_{>u}(v)]$
    $A_i \leftarrow$ 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 $p, 0 \le q \le 1$
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



(a) movielens-10m

(b) flickr

(c) movielens-1m

(d) movielens-100k

(e) S-151M

(f) S-2M

(g) S-917K

Figure 8: Quantitative evaluation of running time in terms of second ↓

In this section, 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 time is set to $10M$. The experimental results are presented in Fig.8. If the runtime exceeds 24 hours, it is recorded as OOT, and the corresponding bars are not shown in the figure.

As observed, Z-Shadow outperforms other algorithms in most cases, particularly on large, dense graphs. Whether it is a contrast accurate algorithm or an approximation algorithm, Z-S has a considerable advantage in running time. In this article, speedup is defined as a multiple of time reduction. On some datasets, Z-shadow is able to achieve speedups of over 500x compared to BCL++. And Z-Shadow achieves 138X speedup on average For a dataset with 151 million edges, Z-Shadow's runtime is only 1400 seconds, while BCL takes more than 24 hours for all parameter settings. Furthermore, the results demonstrate that our algorithm's performance is unaffected by variations in s when t is fixed, which is not the case for BCL, as shown in Fig.8a. This also highlights the positive impact of vertex ordering.

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 as $10M$. Let K be the exact value calculated by BCL++ and $\hat{K}$ be the estimated value, the accuracy of $\hat{K}$ is defined as $1 - \frac{|K-\hat{K}|}{K}$. The results are presented in Figure.9.As shown in the figure, Z-Shadow maintains a relatively stable accuracy, with an error rate of always within 0.5% on all experimental data sets. Other estimation algorithms have large errors on large graphs and are unstable when parameters change. These algorithms have little accuracy at specific parameter Settings for specific graphs.

6.4 Memory Reduction

We evaluate the memory usage of Z-S compared with the following algorithm.

- **Z-S(G)**: **Z-Shadow(Global)** in Section 4.2 removes vertex reordering.

Since Z-S(G) runs out of memory on most graphs, we calculate the memory usage of Z-S(G) by summing the memory usage of Z-S. Also because it is out of memory, we do not cover Z-S(G) in other parts of the experiment.

As shown in Fig. 10, the memory reduction factor is highly dependent on the nature of the graph itself and can be up to 5,000 times. Which property is related to it is still a question worth exploring.

6.5 Impact of Sampling Times

In this section, we will explore the effect of sample size on both accuracy and running times. The experiments are conducted on the same five datasets used in Section 5.3. We adjust the sample size $\tau \in \{100K, 1M, 10M, 100M\}$. Two of the datasets are used as examples to investigate the effect of sample size on the experiment time. Fig.11 shows the results on the accuracy and Fig.12 shows that of running time.

As shown in Fig.11, when the sample size reaches $100K$, 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 Fig.12, when the sampling size is between $100K$ and $10M$, the running time remains stable with minimal variation. However, a significant increase in running time is observed when the sampling size reaches $100M$.

This increase in time is primarily due to the additional time required for sampling. To illustrate this, Table 3 shows the time consumption for dataset M-1m. 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 $10M$ appropriately balances accuracy and time consumption.

Table 3: 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

In Section 6.2, it can be observed that Z-S improves compared to Z-S(NS). In this section, we explore the relationship between the acceleration effect of vertex rearrangement and the hit ratio of bounds defined by Frudie's theorem.

As can be seen from the table 4, the effect of vertex reordering on the hit rate of the boundary is very significant, and the hit rate of the boundary can be increased by 1500 times.

We evaluated the effects of several different sorting applications on Z-S, including ascending order of degree, descending order of degree, descending core ordering, random ordering. Z-S applying these reordering methods are denoted as Z-S(AD), Z-S(DD), Z-S(C), Z-S(R) respectively in the experimental results. The running time of various reordering methods is shown in Fig.13.

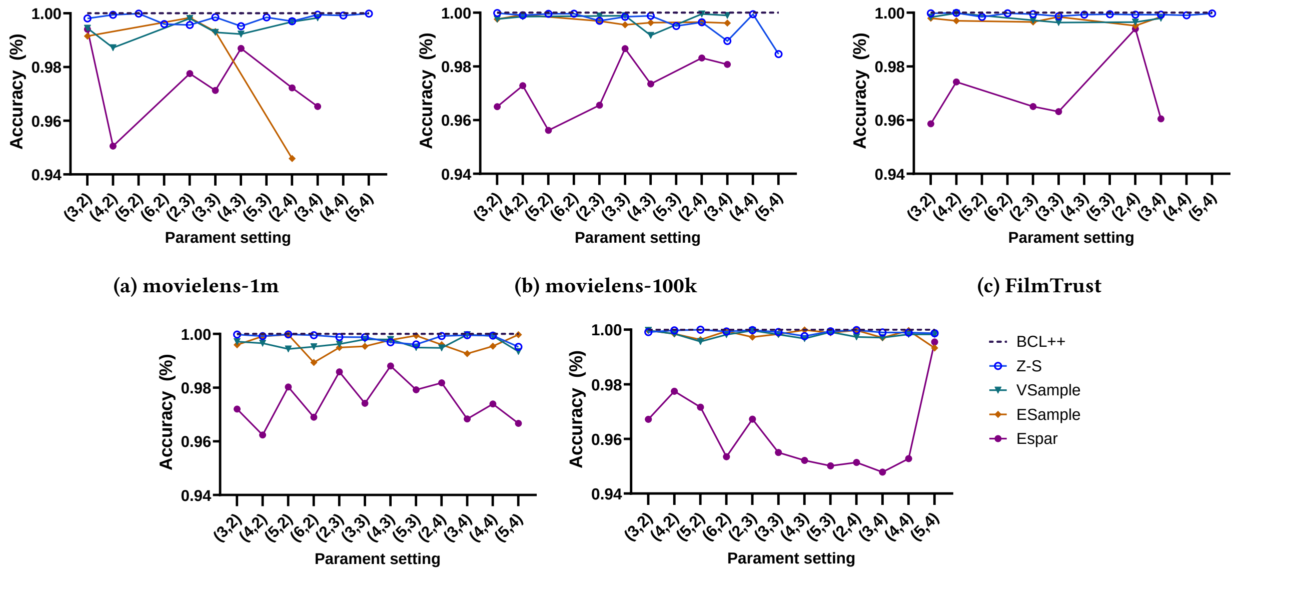
From the results shown in the figure, Z-S(AD) dominates. Although Z-S(C) is similar to Z-S(AD), core-ordering is an expensive process. It can be observed in the figure that there is no significant difference in the influence brought by each reordering method in some synthetic graphs, which is closely related to the fact that the degree of the synthetic graph is generated by the normal distribution. The synthetic graph generated by normal distribution is more evenly distributed with less obvious skew, while the real dataset is the opposite.

7 CONCLUSION

In this paper, we present 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 500x speedup over existing methods in experiments shown in section 6.2. The results in section 6.2 shows Z-S achieves 138X speedup on average while maintaining high accuracy , with errors consistently below 0.5%, which is shown in section 6.3.

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 valuable tool for the large-scale analysis of bipartite networks, with potential applications across various fields where such structures are prevalent.

Looking forward, there are several avenues for future research. One possible direction is to extend the Z-shadow algorithm to handle other types of graph structures, such as weighted or directed bipartite graphs, which would broaden its applicability to a wider range of network analysis problems. Another area worth exploring



(a) movielens-1m

(b) movielens-100k

(c) FilmTrust

(d) WikiLens

(e) moreno_hens

Figure 9: Quantitative evaluation of accuracy in tems of percent ↓

Table 4: The Effect of vertex reordering on the hit ratio of bounds

	movielens-1m		movielens-10m		movielens-100k		flickr		S-151M		S-917K		S-2M	
	sort	unsort	sort	unsort	sort	unsort	sort	unsort	sort	unsort	sort	unsort	sort	unsort
(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

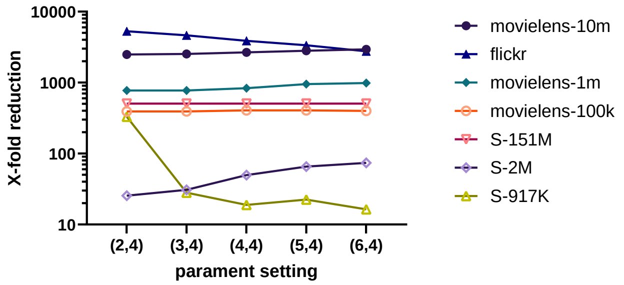


Figure 10: Z-S Memory reduction compared to Z-S (G)

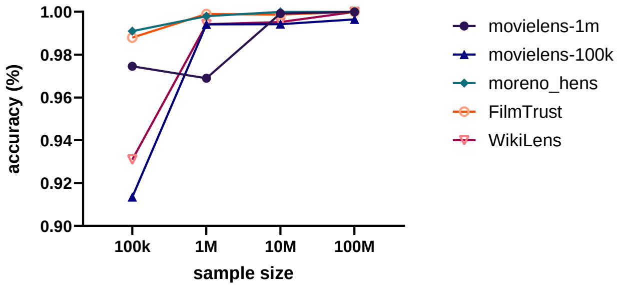
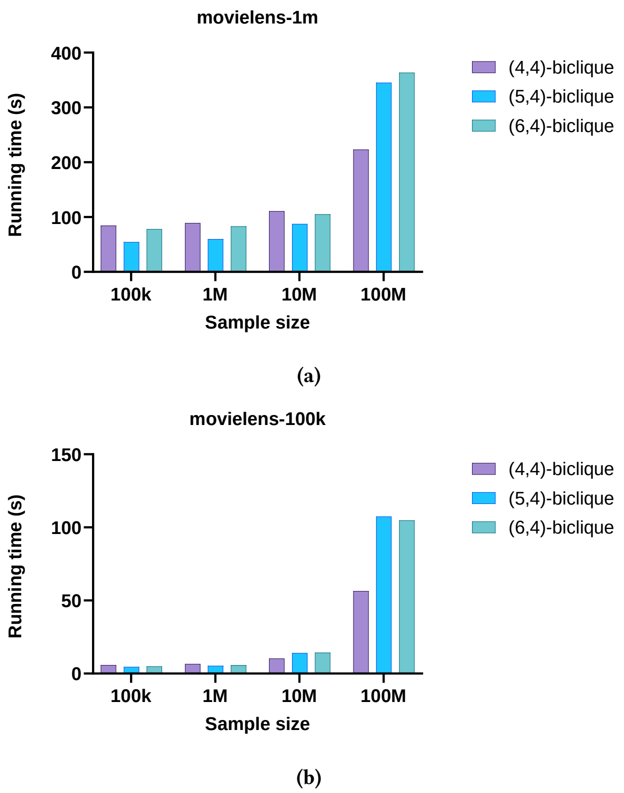


Figure 11: Effect of sample size on accuracy

is the integration of parallel and distributed computing techniques to further enhance the scalability of the method for extremely large datasets. Additionally, investigating potential optimizations in the sampling process could further reduce computational overhead, making the algorithm even more efficient.

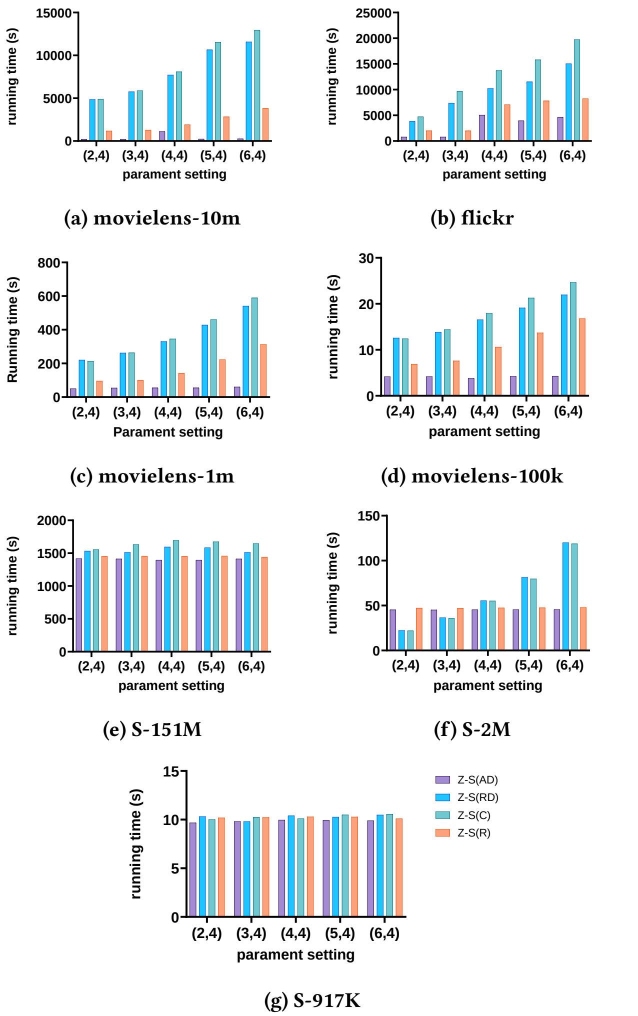
In conclusion, Z-shadow provides a powerful and scalable solution for biclique estimation in large bipartite graphs, offering both theoretical innovations and practical improvements over existing methods. Its combination of speed, accuracy, and memory efficiency makes it an invaluable tool for researchers and practitioners working with complex network data, particularly in domains where large bipartite structures are common, such as bioinformatics, social network analysis, and recommendation systems.



(a)

(b)

Figure 12: Effect of sample size on time



(a) movielens-10m

(b) flickr

(c) movielens-1m

(d) movielens-100k

(e) S-151M

(f) S-2M

(g) S-917K

Figure 13: The running time of various reordering methods is quantitatively evaluated in seconds↓

A MISSING PROOFS

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

A.1 Proof of Theorem 6

In this subsection, we prove Theorem 6. 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 $1 \le r \le \tau$,

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

where the last equation holds by Theorem 5. Again, by Theorem 5, we know $\mathcal{N}(K_{s,q}, G[S]) \ge \gamma\omega(S)$, which implies that

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

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

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

Note that all X_r are independent. Applying multiplicative Chernoff bound [8] for X yields that

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

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

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

and then

$$\mathbb{P}\left[\theta(1-\epsilon) \le \frac{\sum_{1\le r\le\tau} X_r}{\tau} \le \theta(1+\epsilon)\right] \ge 1 - \frac{\delta}{2}.$$

Recall that $\hat{K} = \frac{\sum_{1\le r\le\tau} X_r}{\tau} \sum_{(S,q)\in\mathbf{S}} \binom{|S_L|}{s}\binom{|S_R|}{q}$. We conclude that

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

which is equivalent

$$\mathbb{P}\left[(1-\epsilon)K \le \hat{K} \le K(1+\epsilon)\right] \ge 1-\delta.$$

This completes the proof.

A.2 Proof of Theorem 13

In this subsection, we prove Theorem 13. Let q be a integer and $1 \le q < t$. 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, $\mathbb{P}[X_i \ne 0] = \frac{\mathcal{N}(K_{s,q-1}, G_v)}{\omega_v}$.

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 7, 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

$$\mathbb{E}[W] = \mathbb{E}\left[\sum_{i=1}^{\tau} X_i\right] = \sum_{i=1}^{\tau} \mathbb{E}[X_i] = \frac{\tau K}{\Phi}.$$

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

A.3 Proof of Theorem 14

Now we prove Theorem 14. Note that $\hat{K} = \frac{W}{\tau}\Phi = \frac{\sum_{1\le i\le\tau} X_i}{\tau}\Phi$, $\mathbb{E}\left[\sum_{i=1}^{\tau} X_i\right] = \frac{\tau K}{\Phi}$ in A.2. Let $X = \sum_{1\le i\le\tau} X_i$. Then

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

Applying multiplicative Chernoff bound [8] for X yields that

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

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

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

A.4 Analysis Related to Other Sampling Algorithms

Lemma 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 $R = \left\{v_1, \cdots, v_{|R|}\right\}$. 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 $1 \le i \le \tau$, $\mathbb{E}[A_i] = \frac{\sum_{i=1}^{|R|} X_i}{|R|} = \frac{K}{|R|}$. 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 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 $E\left[\hat{K}\right] = K$.*

The proof can be obtained by analogy with the proof for Lemma 1.

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

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

□

When a graph G without isolated vertex, if

$$e(G) < \min\{m, n\} + p * (q-1)$$

,then G is $K_{p,q}$-free

REFERENCES

[1] Aman Abidi et al. "Pivot-based maximal biclique enumeration". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. IJCAI'20. Yokohama, Yokohama, Japan, 2021. ISBN: 9780999241165.

[2] Clark T Benson. "Minimal regular graphs of girths eight and twelve". In: *Canadian Journal of Mathematics* 18 (1966), pp. 1091–1094.

[3] John A Bondy and Miklós Simonovits. "Cycles of even length in graphs". In: *Journal of Combinatorial Theory, Series B* 16.2 (1974), pp. 97–105.

[4] Lijun Chang, Rashmika Gamage, and Jeffrey Xu Yu. "Efficient k-Clique Count Estimation with Accuracy Guarantee". In: *Proc. VLDB Endow.* 17.11 (Aug. 2024), pp. 3707–3719. ISSN: 2150-8097.

[5] Chen Chen et al. "Efficient critical relationships identification in bipartite networks". In: *World Wide Web-internet And Web Information Systems* 25.2, SI (Mar. 2022), pp. 741–761.

[6] Seshadhri Comandur, Ali Pinar, and Tamara G. Kolda. "Wedge sampling for computing clustering coefficients and triangle counts on large graphs". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 7 (2013).

[7] Maximilien Danisch, Oana Balalau, and Mauro Sozio. "Listing k-cliques in sparse real-world graphs". In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 589–598.

[8] Devdatt P. Dubhashi, Alessandro Panconesi, and Aravind Srinivasan. "Concentration of Measure for the Analysis of Randomized Algorithms". In: *SIGACT News* 1 (2010), p. 41.

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

[10] Paul Erdös. "On a problem in graph theory". In: *The Mathematical Gazette* 47.361 (1963), pp. 220–223.

[11] Yixiang Fang et al. "Efficient algorithms for densest subgraph discovery". In: 12.11 (2019). ISSN: 2150-8097.

[12] Zoltán Füredi. "An Upper Bound on Zarankiewicz' Problem". In: *Combinatorics, Probability and Computing* 5.1 (1996), pp. 29–33.

[13] Felipe Glaria et al. "Compact structure for sparse undirected graphs based on a clique graph partition". In: *Information Sciences* 544 (2021), pp. 485–499.

[14] Shweta Jain and C Seshadhri. "Provably and efficiently approximating near-cliques using the Turán shadow: PEANUTS". In: *Proceedings of The Web Conference 2020*. 2020, pp. 1966–1976.

[15] Shweta Jain and C. Seshadhri. "A Fast and Provable Method for Estimating Clique Counts Using Turán's Theorem". In: *The Web Conference*. 2017.

[16] Shweta Jain and C. Seshadhri. "The Power of Pivoting for Exact Clique Counting". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 268–276. ISBN: 9781450368223.

[17] Madhav Jha, Seshadhri Comandur, and Ali Pinar. "Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts". In: *Proceedings of the 24th International Conference on World Wide Web* (2014).

[18] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. "Basic notions for the analysis of large two-mode networks". In: *Social networks* 30.1 (2008), pp. 31–48.

[19] Felix Lazebnik and Vasiliy A Ustimenko. "New examples of graphs without small cycles and of large size". In: *European Journal of Combinatorics* 14.5 (1993), pp. 445–460.

[20] Felix Lazebnik, Vasiliy A Ustimenko, and Andrew J Woldar. "Polarities and 2k-cycle-free graphs". In: *Discrete Mathematics* 197 (1999), pp. 503–513.

[21] Rong-Hua Li et al. "Ordering heuristics for k-clique listing". In: 13.12 (2020). ISSN: 2150-8097.

[22] Rundong Li et al. "Approximately Counting Butterflies in Large Bipartite Graph Streams". In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2022), pp. 5621–5635.

[23] Boge Liu et al. "Efficient (α, β)-core Computation: an Index-based Approach". In: *Web Conference 2019: Proceedings of The World Wide Wed Conference (WWW 2019)*. 2019, pp. 1130–1141.

[24] Samuel Manoharan et al. "Patient diet recommendation system using K clique and deep learning classifiers". In: *Journal of Artificial Intelligence* 2.02 (2020), pp. 121–130.

[25] RA Rossi and R Zhou. *System and method for compressing graphs via cliques*. 2019.

[26] Ryan A Rossi, Nesreen K Ahmed, and Eunyee Koh. "Higher-order network representation learning". In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 3–4.

[27] Seyed Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. "Butterfly Counting in Bipartite Networks". In: *ACM* (2018).

[28] Jessica Shi, Laxman Dhulipala, and Julian Shun. "Parallel Clique Counting and Peeling Algorithms". In: *Conference on Applied and Computational Discrete Algorithms*. 2020.

[29] Phonexay Vilakone et al. "An efficient movie recommendation algorithm based on improved k-clique". In: *Human-centric Computing and Information Sciences* 8 (2018), pp. 1–15.

[30] Haibo Wang et al. "Deep Structure Learning for Fraud Detection". In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 567–576.

[31] Kai Wang et al. "Accelerated butterfly counting with vertex priority on bipartite graphs". In: *The VLDB Journal* 32 (2022), pp. 257–281.

[32] Kaixin Wang, Kaiqiang Yu, and Cheng Long. "Efficient k-Clique Listing: An Edge-Oriented Branching Strategy". In: *Proc. ACM Manag. Data* 2.1 (Mar. 2024).

[33] Mengyuan Wang et al. "Toward Accurate Butterfly Counting with Edge Privacy Preserving in Bipartite Networks". In: *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. 2024, pp. 2289–2298.

[34] Pinghui Wang et al. "MOSS-5: A Fast Method of Approximating Counts of 5-Node Graphlets in Large Graphs". In: *IEEE Transactions on Knowledge and Data Engineering* 30.1 (2018), pp. 73–86.

[35] Rephael Wenger. "Extremal graphs with no C4's, C6's, or C10's". In: *Journal of Combinatorial Theory, Series B* 52.1 (1991), pp. 113–116.

[36] Qingyu Xu et al. "Efficient load-balanced butterfly counting on GPU". In: *Proc. VLDB Endow.* 15.11 (July 2022), pp. 2450–2462. ISSN: 2150-8097.

[37] Jianye Yang et al. "(p, q)-biclique counting and enumeration for large sparse bipartite graphs". In: *The VLDB Journal* 32.5 (2023), pp. 1137–1161.

[38] Xiaowei Ye et al. "Efficient biclique counting in large bipartite graphs". In: *Proceedings of the ACM on Management of Data* 1.1 (2023), pp. 1–26.

[39] Yanlei Yu et al. "Rum: Network representation learning using motifs". In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. 2019, pp. 1382–1393.

[40] Long Yuan et al. "Index-Based Densest Clique Percolation Community Search in Networks". In: *IEEE Transactions on Knowledge and Data Engineering* 30.5 (2018), pp. 922–935.

[41] Ge Zhang et al. "eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks". In: 40.3 (2022).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009