Estimating Biclique Counts with Accuracy Guarantees

RASHMIKA GAMAGE, The University of Sydney, Australia
LIJUN CHANG, The University of Sydney, Australia

Efficiently counting bicliques in large bipartite graphs is a fundamental problem with applications in network analysis, bioinformatics, and social sciences. However, existing exact algorithms do not scale well to large graphs, and current approximation algorithms lack formal accuracy guarantees. In this paper, we present the first approximation algorithm for the (p, q)-biclique counting problem that offers formal accuracy guarantees. Our approach introduces a novel sampling framework, termed BC-Shadow, which refines the sample space using edge-oriented techniques to strategically balance computational costs across algorithmic stages. This refinement increases the density of bicliques in the sample space, reducing the number of samples required for accurate estimation. Our algorithm adaptively determines the number of successful samples that are needed to satisfy predefined error and failure probability, enabling real-time adjustment to graph properties. We further enhance sampling efficiency with a new sampling structure, named zstar, which establishes a one-to-one correspondence with (p, q)-bicliques, eliminating redundancies and improving accuracy. Comprehensive theoretical analyses confirm the algorithm's accuracy and running time guarantees, while extensive experiments on large real-world datasets demonstrate its scalability and effectiveness.

CCS Concepts: • **Mathematics of computing → Approximation algorithms**; **Graph algorithms**.

Additional Key Words and Phrases: Bipartite Graphs, Biclique Counting, Approximation Algorithms

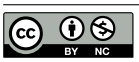
ACM Reference Format:
Rashmika Gamage and Lijun Chang. 2025. Estimating Biclique Counts with Accuracy Guarantees. *Proc. ACM Manag. Data* 3, 6 (SIGMOD), Article 326 (December 2025), 26 pages. https://doi.org/10.1145/3769791

1 Introduction

Efficient enumeration and counting of bicliques in large bipartite graphs have gained significant research attention due to their extensive applications in areas such as network analysis, bioinformatics, data mining, social sciences, and recommendation systems [4, 17, 21, 27, 31]. Formally, a (p, q)-biclique in a bipartite graph $G = (U, V, E)$ is a pair of subsets $B_U \subseteq U$ and $B_V \subseteq V$ such that every vertex in B_U is directly connected to every vertex in B_V via an edge, with $|B_U| = p$ and $|B_V| = q$. This paper studies the problem of counting all (p, q)-bicliques in a large-scale bipartite graph for a given p and q. Bicliques serve as a critical structural pattern for discovering dense clusters [26, 29], analyzing network motifs, and interpreting complex interactions within large-scale datasets [3, 24, 36]. These structures are essential across diverse applications, ranging from biological data integration to higher-order clustering in complex networks [28, 36].

For example, biclique counting has been used in computing higher-order clustering coefficients for bipartite graphs [34, 36]. These coefficients are defined as the ratio between the number of (p, q)-bicliques and the number of wedges, a structural pattern closely related to bicliques. This ratio reflects the likelihood that a wedge closes into a biclique, thereby revealing fundamental characteristics of the bipartite graph's structure. In particular, the distribution of higher-order clustering coefficient for $p = q \in [2, 10]$ based on approximate biclique counting was plotted in [34]

Authors' Contact Information: Rashmika Gamage, The University of Sydney, Sydney, Australia, rashmika.gamage@sydney.edu.au; Lijun Chang, The University of Sydney, Sydney, Australia, Lijun.Chang@sydney.edu.au.



ACM 2836-6573/2025/12-ART326
https://doi.org/10.1145/3769791

for twelve real bipartite graphs from four different domains. The results show that bipartite graphs from the same domain tend to exhibit similar distributions, whereas graphs from different domains typically show distinct distributions.

Another related problem is biclique sampling, which aims to uniformly sample a specific number of (p, q)-bicliques from a bipartite graph. This technique is useful for accelerating the computation of (p, q)-biclique densest subgraph, a problem introduced in [22]. The (p, q)-biclique density of a graph is defined as the number of (p, q)-bicliques divided by the number of vertices in the graph. Since computing the exact (p, q)-biclique densest subgraph is computationally challenging, an approximation algorithm is proposed in [22], which samples a number of (p, q)-bicliques uniformly at random and then performs the computation based on the sampled bicliques. We note that our algorithm proposed in this paper can be directly applied to uniformly sample a desired number of (p, q)-biclques. Therefore, our techniques can serve as an essential component for solving the (p, q)-biclique densest subgraph problem.

Despite their practical importance, the biclique counting problem remains computationally challenging. Traditional exact biclique counting methods rely heavily on exhaustive enumeration. The BCList and BCList++ [30] algorithms utilize depth-first expansion with candidate set pruning, offering improved performance on sparse graphs. However, due to the exponential blowup of the biclique space, these methods are impractical for large or dense graphs. EPivoter [34], a novel exact algorithm based on edge pivoting, is an efficient approach for biclique counting. While EPivoter avoids full enumeration by leveraging a clever combinatorial counting strategy across pivoted biclique representations, its performance is still tied to the need for enumerating a large subset of bicliques. This makes it sensitive to graph density and structural irregularities.

To address the challenges of exact biclique counting in large bipartite graphs, sampling-based methods such as ZigZag have been proposed. ZigZag [34] introduces a sampling structure, called *zigzag*. The core idea is that every (p, q)-biclique contains *at least one* zigzag, which makes it possible to estimate the biclique count by uniformly sampling zigzags. Assuming $p \leq q$, a zigzag is defined as a vertex-ordered path consisting of p vertices from U and p vertices from V. A (p, q)-biclique (B_U, B_V) contains $\binom{q}{p}$ distinct zigzags, each corresponding to a subset of B_V of size p. Given a zigzag Z, let $Z_U = Z \cap U$, $\Lambda(Z_U)$ denote the set of common neighbors of Z_U, and $c_{p,q}(Z)$ denote the number of (p, q)-bicliques that contain Z. If Z forms a biclique, then $c_{p,q}(Z) = \binom{|\Lambda(Z_U)|-p}{q-p}$; otherwise, $c_{p,q}(Z) = 0$. The sample space $\mathcal{H}$ consists of all zigzags in G. ZigZag uniformly samples a set $\mathcal{T}$ of zigzags from $\mathcal{H}$, and estimates the number of (p, q)-bicliques using the unbiased estimator $\frac{|\mathcal{H}| \sum_{Z \in \mathcal{T}} c_{p,q}(Z)}{|\mathcal{T}| \binom{q}{p}}$. While efficient, this method has several limitations. Most notably, *ZigZag does not provide formal accuracy guarantees*. It requires end-users to manually specify the number of samples $t = |\mathcal{T}|$, and the quality of the estimate heavily depends on this choice. Setting t too small leads to poor approximation quality, while a large t increases computational overhead. However, determining an appropriate value of t is non-trivial and can vary significantly across datasets.

Z-Shadow [5] is another approach for estimating the number of (p, q)-bicliques in a graph. A crucial step in the Z-Shadow framework involves constructing a *shadow*, a multiset of tuples (S, ℓ) with $S \subseteq U \cup V$ and $0 < \ell < q$. The sample space $\mathcal{S}$ is defined as the union of all subsets $S' \subseteq S$ that contain exactly p vertices from S_U (i.e., $S \cap U$) and ℓ vertices from S_V, for each (S, ℓ) in the shadow. It is shown in [5] that there is a one-to-one correspondence between bicliques in the sample space and (p, q)-biclques in G. Therefore, Z-Shadow estimates the (p, q)-biclique count by sampling t elements from $\mathcal{S}$, computing the fraction that are bicliques, and scaling by $|\mathcal{S}|$. To ensure an accurate estimation with relative error at most ϵ and failure probability at most δ, Z-Shadow constructs the shadow in such a way that each tuple (S, ℓ) corresponds to a vertex-induced subgraph $G[S]$ that is *sufficiently dense to guarantee the presence of a (p, ℓ)-biclique*. This determination is

guided by a threshold function $\alpha(S, p, \ell)$, derived from Füredi's upper bound on Zarankiewicz's problem [12], which provides extremal limits on the number of edges in a bipartite graph without containing a (p, ℓ)-biclique. As a result, let $cnt_{p,q}(G)$ be the number of (p, q)-bicliques in G, then $\frac{cnt_{p,q}(G)}{|\mathcal{S}|} \geq \min_{(S,\ell)} \frac{1}{\binom{|S_U|}{p} \cdot \binom{|S_V|}{q}}$; denote this lower bound by μ_{lb}. Applying the Chernoff bound, setting $t = \frac{3 \ln(2/\delta)}{\mu_{\text{lb}} \cdot \epsilon^2}$ is sufficient to ensure the desired estimation accuracy. Z-Shadow has two main limitations. First, ensuring that each tuple (S, ℓ) in the shadow is sufficiently dense to guarantee the presence of a (p, ℓ)-biclique can make the shadow construction computational expensive — often requiring an impractically long time, as confirmed by our experiments. Second, the derived lower bound μ_{lb} is overly conservative, leading to a prohibitively large number of required samples t. To mitigate this, *Z-Shadow sets t manually as a fixed constant in its implementation.* However, this heuristic approach eliminates formal accuracy guarantees and undermines the reliability of the resulting estimates.

Our Approach. In this paper, we introduce a novel sampling structure, called *zstar*, for estimating biclique counts. A zstar is a graph structure consisting of p vertices from U and q vertices from V, formed by concatenating a vertex-ordered path with a star. While similar to the zigzag structure of [34], the zstar has two key advantages: (1) each (p, q)-biclique contains exactly one zstar; and (2) each zstar is contained in at most one (p, q)-biclique. These properties enable the use of a dynamic sampling strategy based on the stopping rule theorem [9], which allows the required number of samples t to be determined adaptively to meet a target accuracy. In contrast, the zigzag structure is incompatible with this theorem. We further propose a novel edge-oriented sample space refinement strategy that reduces the sample space size and increases the density of bicliques within it. This leads to a significant reduction in the number of samples required for accurate estimation. Finally, we design a two-stage sampling framework that effectively balances the computational cost between sample space refinement and zstar-based count estimation, resulting in improved overall efficiency without compromising accuracy.

Our main contributions are summarized as follows:

- We propose the first approximation algorithm that provides formal accuracy guarantees for estimating biclique counts.
- We introduce the *zstar structure*, a new sampling abstraction that establishes a one-to-one correspondence with bicliques, eliminating redundancy and enhancing estimation accuracy.
- We propose a novel *edge-oriented sample space refinement strategy* that increases the density of bicliques within the sample space, thereby significantly reducing the number of samples needed for accurate estimation.
- We design a *two-stage sampling framework* that effectively balances the computational cost between sample space refinement and zstar-based count estimation, resulting in improved overall efficiency without compromising accuracy.
- We demonstrate through extensive experiments on large real-world datasets that our approach outperforms state-of-the-art methods in both performance and estimation reliability.

Source code for the proposed algorithms can be found at https://anonymous.4open.science/r/biclique-approximation.

Related Work. Our work on biclique counting is closely related to two other lines of research: maximal biclique enumeration and maximum edge biclique computation. Maximal biclique enumeration focuses on listing all maximal bicliques, i.e., bicliques that are not contained in other bicliques. Recent advances utilize pivoting [1] and core-based pruning [8] to improve efficiency. Complementary work by Yao et al. [32] extends this direction to similar bicliques, incorporating vertex

similarity constraints to enhance pattern relevance in applications such as anomaly detection. In terms of scalability, parallel and hardware-accelerated solutions have recently pushed performance further. Pan et al. [23] integrate an aggressive merge-based pruning strategy into the search tree, presenting an Aggressive MBE Algorithm (AMBEA) that achieves up to 5.3× speedup over prior methods and shows improved scalability on large graphs. These techniques cannot be used for counting bicliques of a specific size due to different problem nature.

Maximum edge biclique computation, which aims to find a biclique with the maximum number of edges, is an NP-hard problem. Nevertheless, practical algorithms have emerged due to the problem's importance. Lyu et al. [19, 20] combine graph decomposition with aggressive pruning to tackle this problem at billion-scale, reporting success on graphs with millions of vertices. Variants of this problem impose additional structural constraints. For example, Chen et al. [7] design exact algorithms for the maximum balanced biclique problem, enforcing the constraint of equal-sized partitions. Zhao et al. [37] extend this to the weighted setting, introducing methods to find the maximum k-balanced biclique in weighted bipartite graphs. Beyond algorithmic techniques, index-based methods have also emerged to support biclique queries: Ye et al. [33] propose BCviz, a linear-space index that enables fast retrieval and visualization of cohesive subgraphs, including bicliques, in large bipartite graphs.

In contrast to these enumeration and optimization tasks, our goal is the efficient and accurate estimation of (p, q)-biclique counts. Our method, BC-Shadow, complements existing efforts by offering a provably accurate summary of the biclique landscape, enabling scalable structural analysis and motif-based metrics in large bipartite graphs. In this paper, we focus on (p, q)-bicliques with $p, q \in [3, 10]$, following the setting used in [34]. Consequently, we do not employ existing pruning techniques (e.g., those proposed in [19]) that remove vertices and edges guaranteed not to participate in any (p, q)-biclique. Nevertheless, such pruning techniques can be valuable for reducing the search space when counting (p, q)-bicliques with large p and q, which we leave as an avenue for future work.

Recent advancements have also seen significantly improved approximation algorithms with formal accuracy guarantees for counting related substructures such as k-cliques [10, 13, 14, 28, 35], triangles and butterflies [18, 25, 28], and graphlets [11, 16]. Techniques leveraging Turán's theorem [13, 15], color-coding [35], and edge-oriented adaptive sampling strategies [6, 34] have demonstrated practical efficiency while providing strong theoretical guarantees. However, approximation algorithms specifically designed for counting (p, q)-bicliques in bipartite graphs—while offering formal accuracy guarantees—remain largely underexplored.

2 Preliminaries

In this paper, we focus on a large unweighted and undirected bipartite graph $G = (U, V, E)$, where U and V are disjoint sets of vertices, and E is the set of edges. For each vertex $u \in U$ (or V), its set of neighbors is denoted by $N(u) = \{v \mid e(u, v) \in E\}$, and its degree is $d(u) = |N(u)|$. The set of common neighbors of a vertex subset S is denoted by $\Lambda(S) = \bigcap_{u \in S} N(u)$. Given two vertex subsets, $B_U \subseteq U$ and $B_V \subseteq V$, the subgraph of G induced by (B_U, B_V) is denoted by $G[B_U, B_V] = (B_U, B_V, \{e(u, v) \in E \mid u \in B_U, v \in B_V\})$.

Definition 1 (Biclique). *A bipartite graph $g = (U', V', E')$ is a* biclique *if every vertex $u \in U'$ is adjacent to every vertex $v \in V'$, i.e., $E' = \{(u, v) \mid u \in U', v \in V'\}$.*

Any biclique in G apparently must be a vertex-induced subgraph. Thus, we simply refer to a biclique $G[B_U, B_V]$ of G by its vertex sets (B_U, B_V). A biclique (B_U, B_V) is called a (p, q)-*biclique* if $|B_U| = p$ and $|B_V| = q$. We denote the set of all (p, q)-bicliques in G by $\mathcal{B}_{p,q}(G)$, and let $cnt_{p,q}(G) = \left|\mathcal{B}_{p,q}(G)\right|$ be the number of (p, q)-bicliques in G. Consider the graph in Figure 1, it has six

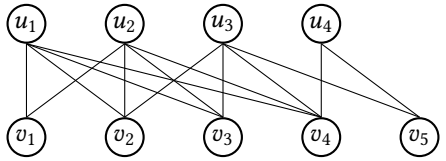


Fig. 1. An example graph

Table 1. Frequently used notations

Notation	Meaning
$G = (U, V, E)$	A bipartite graph
$N_{>v}(u)$	u's neighbors that rank higher than v
$\Lambda(S)$	The set of common neighbors of S, i.e., $\bigcap_{u \in S} N(u)$
$\mathcal{B}_{p,q}(G)$	The set of all (p, q)-bicliques in G
$cnt_{p,q}(G)$	Total number of (p, q)-bicliques in G, i.e., $\lvert\mathcal{B}_{p,q}(G)\rvert$
$\widehat{cnt}_{p,q}(G)$	An estimation of $cnt_{p,q}(G)$
$\Delta(G)$	The set of all h-zstars in G for $h = p$
$\mathcal{S}_{p,q}(G)$	Sample space of $\mathcal{B}_{p,q}(G)$ satisfying $\mathcal{S}_{p,q}(G) \supseteq \mathcal{B}_{p,q}(G)$
$\mathcal{P}_{p',q'}(S_U, S_V)$	The set of sampling structures in $S_U \cup S_V$
$\mathbb{S}_{p,q}(G)$	BC-Shadow
(R_U, R_V, S_U, S_V)	A sample subspace of $\mathbb{S}_{p,q}(G)$
t	The total number of samples taken from $\mathcal{S}_{p,q}(G)$
s	The number of samples that are (p, q)-bicliques

$(2, 3)$-bicliques, i.e., $\mathcal{B}_{2,3}(G) = \big\{(\{u_1, u_2\}, \{v_1, v_2, v_3\}), (\{u_1, u_2\}, \{v_1, v_2, v_4\}), (\{u_1, u_2\}, \{v_1, v_3, v_4\}), (\{u_1, u_2\}, \{v_2, v_3, v_4\}), (\{u_2, u_3\}, \{v_2, v_3, v_4\}), (\{u_1, u_3\}, \{v_2, v_3, v_4\})\big\}$.

Problem Statement. Given a large bipartite graph G, two integers p and q, an error parameter $\epsilon \in (0, 1)$ and a failure probability $\delta \in (0, 1)$, we study the problem of estimating the number of (p, q)-bicliques in G. That is, our goal is to design a randomized algorithm that outputs an estimated value $\widehat{cnt}_{p,q}(G)$ satisfying

$$\Pr\left(\left|\widehat{cnt}_{p,q}(G) - cnt_{p,q}(G)\right| > \epsilon \cdot cnt_{p,q}(G)\right) \leq \delta$$

For simplicity, we assume that $p \leq q$. Our method can also be applied when $p > q$ by simply swapping U and V. Frequently used notations are summarized in Table 1. We will use u to refer to vertices of U, and v to vertices of V.

2.1 Accuracy Guarantee

The existing algorithms [5, 34] require the number of samples to be manually set by end-users, and do not provide any formal assurance on the accuracy of their estimated biclique counts. In this subsection, we introduce an adaptive sampling strategy motivated by the stopping rule theorem that dynamically determines the number of samples required to achieve a desired level of accuracy.

Theorem 1 (Stopping Rule Theorem [9]). *Let X be a random variable distributed in $[0, 1]$ with mean $\mu_X > 0$, and $X_1, X_2, \ldots$ be independently and identically distributed according to X. Let N_X be the smallest integer such that $\sum_{i=1}^{N_X} X_i \geq \gamma = 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$ and $\tilde{\mu}_X = \frac{1}{N_X}\sum_{i=1}^{N_X} X_i$. Then,*

(1) $\Pr(|\tilde{\mu}_X - \mu_X| > \epsilon \cdot \mu_X) \leq \delta$.
(2) $\frac{\gamma}{\mu_X} \leq \mathbb{E}[N_X] \leq \frac{\gamma+1}{\mu_X}$.

Let $\mathcal{S}_{p,q}(G)$ be a sample space constructed for estimating biclique counts. For concreteness, let's assume for now that $\mathcal{S}_{p,q}(G)$ consists of all possible pairs of vertex subsets (B_U, B_V) with $B_U \subseteq U$, $|B_U| = p$, $B_V \subseteq V$, and $|B_V| = q$, which gives $|\mathcal{S}_{p,q}(G)| = \binom{|U|}{p} \cdot \binom{|V|}{q}$; we will propose more advanced sample spaces in Section 3. Define the biclique density within $\mathcal{S}_{p,q}(G)$, denoted by $\mu_{\mathcal{S}_{p,q}(G)}$, as

$$\mu_{\mathcal{S}_{p,q}(G)} = \frac{cnt_{p,q}(G)}{|\mathcal{S}_{p,q}(G)|} \tag{1}$$

which is between 0 and 1. We can approximate $cnt_{p,q}(G)$ by approximating $\mu_{\mathcal{S}p,q(G)}$. Let X_i be an indicator variable for each pair (B_U^i, B_V^i) sampled uniformly at random (u.a.r.) from $\mathcal{S}_{p,q}(G)$:

$$X_i = \begin{cases} 1, & \text{if } (B_U^i, B_V^i) \text{ forms a biclique in } G, \\ 0, & \text{otherwise.} \end{cases}$$

Then, $\mathbb{E}[X_i] = \mu_{\mathcal{S}p,q(G)}$ and $\widehat{\mu}_{\mathcal{S}p,q(G)} = \frac{1}{t}\sum_{i=1}^{t} X_i$ is an unbiased estimator for $\mu_{\mathcal{S}p,q(G)}$, where t is the number of samples. Consequently, $\widehat{cnt}_{p,q}(G) = |\mathcal{S}_{p,q}(G)| \cdot \widehat{\mu}_{\mathcal{S}p,q(G)}$ is an unbiased estimator for the total number of (p,q)-bicliques in G.

The accuracy of the estimation depends on the number of samples t. In general, a larger t leads to a more accurate estimate. However, increasing t also results in higher computational cost. Determining a minimum value of t that guarantees a desired level of accuracy, specified by parameters (ϵ, δ), is non-trivial. One approach, adopted by the existing works such as [5, 34], is to leverage concentration inequalities, such as the Chernoff bound. Specifically, we have $\Pr\left(\left|\widehat{\mu}_{\mathcal{S}p,q(G)} - \mu_{\mathcal{S}p,q(G)}\right| > \epsilon \cdot \mu_{\mathcal{S}p,q(G)}\right) \leq 2 \cdot \exp\left(-\frac{\epsilon^2 \cdot \mu_{\mathcal{S}p,q(G)} \cdot t}{3}\right)$. Then, to achieve a failure probability of at most δ, we need $2 \cdot \exp\left(-\frac{\epsilon^2 \cdot \mu_{\mathcal{S}p,q(G)} \cdot t}{3}\right) \leq \delta$ and consequently $t \geq \frac{3\ln(2/\delta)}{\epsilon^2 \cdot \mu_{\mathcal{S}p,q(G)}}$. However, the biclique density $\mu_{\mathcal{S}p,q(G)}$ is unknown. Chang et al. [5] address this by constructing the sample space in a way that enables the computation of μ_{lb}, a lower bound on $\mu_{\mathcal{S}p,q(G)}$, which is then used to determine an appropriate value of t. This approach has two drawbacks: (1) the sample space construction is computationally expensive, as it must be carefully designed to ensure the lower bound can be obtained; and (2) the derived lower bound μ_{lb} is highly conservative, potentially resulting in an excessively large t.

Instead, we employ an adaptive sampling strategy, motivated by the stopping rule theorem (i.e., Theorem 1). Specifically, we fix the number of required successful samples as $\gamma = 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$ (i.e., samples that correspond to actual (p,q)-bicliques), and continue sampling from $\mathcal{S}_{p,q}(G)$ until γ successful samples have been collected. We then estimate the biclique count as $\widehat{cnt}_{p,q}(G) = |\mathcal{S}_{p,q}(G)| \cdot \frac{\gamma}{t}$, where t is the total number of samples drawn. The accuracy is guaranteed by Theorem 1. Algorithm 1 summarizes the details.

Remarks. The main contribution of this paper is the design of a novel sampling structure and sample space that can be efficiently constructed and seamlessly integrated with Algorithm 1 to enable biclique count estimation with theoretical accuracy guarantees. It is worth noting that the sampling structure proposed in [34] is incompatible with Algorithm 1, as will be discussed in Section 3.1.1.

Let $X_1, X_2, \ldots, X_n$ be independent samples drawn from a distribution $X \sim \mathcal{D}$. Recently, ScaleGPM [2] proposed modeling the empirical mean $\mu := \frac{1}{n}\sum_{i=1}^{n} X_i$ and dynamically assessing whether μ has sufficiently converged to $\mathbb{E}[X]$. Specifically, it is shown in [2] that as $n \to \infty$, $\frac{\mu - \mathbb{E}[X]}{\mathbb{V}ar[\mu]}$ converges in distribution to a standard normal distribution (by the central limit theorem), and that the sample variance $\frac{1}{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i^2 - \mu^2\right)$ converges in probability to $\mathbb{V}ar[\mu]$ (by the law of large numbers). Based on these results, given n samples, both μ and the empirical variance can be used to construct a

Algorithm 1: BicliqueEstimation($G, p, q, \epsilon, \delta, \mathcal{S}_{p,q}(G)$)

Input: A bipartite graph G, two integers p, q, error parameters $\epsilon \in (0,1)$ and $\delta \in (0,1)$, and a sample space $\mathcal{S}_{p,q}(G)$

Output: An estimate $\widehat{cnt}_{p,q}(G)$ of $cnt_{p,q}(G)$

$s \leftarrow 0$; $t \leftarrow 0$; $\gamma \leftarrow 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$;

while $s < \gamma$ **do**

  Sample an element (B_U, B_V) from $\mathcal{S}_{p,q}(G)$ u.a.r.;

  if (B_U, B_V) *forms a* (p,q)*-biclique in* G **then** $s \leftarrow s + 1$;

  $t \leftarrow t + 1$;

return $\widehat{cnt}_{p,q}(G) \leftarrow |\mathcal{S}_{p,q}(G)| \cdot \frac{s}{t}$;

confidence interval for $\mathbb{E}[X]$ at a user-specified confidence level $1-\delta$. The algorithm terminates the sampling process once the estimated relative error falls below a user-given threshold ϵ. However, we remark that since these theoretical guarantees hold only in the asymptotic regime $n \to \infty$, the accuracy of the estimate for $\mathbb{E}[X]$ is not formally guaranteed in finite-sample settings. In contrast, Theorem 1 holds for finite samples.

3 Our Approach

In this section, we first introduce a new sampling structure, called zstar, in Section 3.1, then propose an edge-oriented sample space refinement strategy in Section 3.2, and finally discuss how to balance the computational cost between sample space refinement and zstar-based count estimation in Section 3.3. Our techniques require the vertices on each side of the bipartite graph G to be arranged in a particular order; that is, $u_1 \prec u_2 \prec \cdots \prec u_{|U|}$ for U and $v_1 \prec v_2 \prec \cdots \prec v_{|V|}$ for V. For simplicity, we adopt the natural (increasing) ordering of the vertices throughout Sections Section 3.1–3.3, and discuss how to order the vertices in Section 3.4.

3.1 A Novel Sampling Structure

As discussed in Section 2.1, the naive sample space $\mathcal{S}_{p,q}(G)$, consisting of all possible subsets of p vertices from U and q vertices from V, is sufficient to guarantee the desired estimation accuracy. However, this results in a prohibitively large sample space of size $\binom{|U|}{p} \cdot \binom{|V|}{q}$, leading to an extremely small biclique density $\mu_{\mathcal{S}_{p,q}(G)} = \frac{cnt_{p,q}(G)}{|\mathcal{S}_{p,q}(G)|}$. As shown in Theorem 1, the sample complexity (i.e., the number of samples required) is inversely proportional to $\mu_{\mathcal{S}_{p,q}(G)}$. Consequently, the naive sample space becomes computationally impractical for large graphs. To mitigate this, we reduce the sample space by eliminating vertex combinations that are guaranteed to not form bicliques. This is achieved through a novel sampling structure, called *zstar*, which imposes structural constraints on vertex selection. By enforcing these constraints, zstar significantly reduces the sample space, thereby improving the efficiency of the estimation process.

Definition 2 (zstar). *Given a bipartite graph G and integers p, q, h s.t. $p \le q$, an h-zstar in G is defined as an ordered vertex sequence $Z = (u_{i_1}, v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_h}, v_{j_h}, v_{j_{h+1}}, \ldots, v_{j_{h+q-p}})$ such that*

(1) *it contains h vertices from U and $h+q-p$ vertices from V, i.e., $|Z| = 2h+q-p$;*
(2) *its vertices are ordered with $i_1 < i_2 < \cdots < i_h$ and $j_1 < j_2 < \cdots < j_h < j_{h+1} < \cdots < j_{h+q-p}$;*
(3) *$(u_{i_1}, v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_h})$ forms a path in G, while $(u_{i_h}, v_{j_h}, v_{j_{h+1}}, \ldots, v_{j_{h+q-p}})$ forms a star in G.*

Additionally, the length of an h-zstar is defined as $2h-1$.

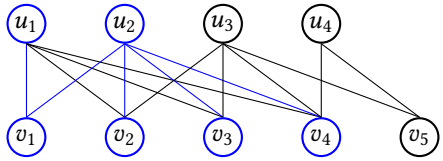


Fig. 2. An example for 2-zstar ($q - p = 2$)

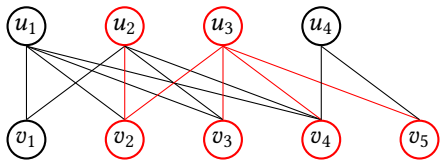


Fig. 3. Another example for 2-zstar ($q - p = 2$)

For brevity, we may refer to an h-zstar simply as a zstar when the parameter h is not particularly relevant. Figures 2 and 3 show two 2-zstars in the example graph G for $q - p = 2$. The 2-zstar in Figure 2 forms a $(2, 4)$-biclique while the one in Figure 3 does not.

Our primary goal in designing zstar is to establish a one-to-one correspondence from bicliques to zstars such that every biclique corresponds to exactly one zstar, as proved in the lemma below.

Lemma 1. *Every (p, q)-biclique in G with $p \leq q$ contains exactly one h-zstar for $h = p$.*

Proof. Let's consider an arbitrary (p, q)-biclique; without loss of generality, assume it is $(\{u_1, u_2, \ldots, u_p\}, \{v_1, v_2, \ldots, v_q\})$. Then, these vertices form a unique zstar $(u_1, v_1, u_2, v_2, \ldots, u_p, v_p, v_{p+1}, \ldots, v_q)$, while any other ordering of these vertices is not a zstar. □

Moreover, each zstar is contained in at most one biclique, as both have $p + q$ vertices. Let $X(Z)$ be the number of (p, q)-bicliques containing an h-zstar Z for $h = p$. Then, we have $X(Z) \in \{0, 1\}$ and

$$X(Z) = \begin{cases} 1, & \text{if } Z \text{ forms a } (p, q)\text{-biclique} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Lemma 2. *Let $\Delta(G)$ be the set of all h-zstars in G for $h = p$, and Z be a random h-zstar uniformly sampled from $\Delta(G)$. The probability that Z forms a (p, q)-biclique in G is*

$$\Pr(Z \textit{ forms a } (p, q)\textit{-biclique in } G) = \frac{cnt_{p,q}(G)}{|\Delta(G)|}$$

Proof. As each (p, q)-biclique in G corresponds to exactly one h-zstar for $h = p$ and each h-zstar is contained in at most one (p, q)-biclique, the total number of h-zstars of $\Delta(G)$ that form (p, q)-bicliques is equal to $cnt_{p,q}(G)$. Then, the lemma follows from the fact that Z is sampled uniformly at random from $\Delta(G)$. □

Based on Lemma 2, we construct an unbiased estimator for the count of (p, q)-bicliques by uniformly sampling Z from $\Delta(G)$.

Theorem 2. *Let $\mathcal{T}$ be a set of zstars uniformly sampled from $\Delta(G)$. Then,*

$$\widehat{cnt}_{p,q}(G) = \frac{|\Delta(G)| \sum_{Z \in \mathcal{T}} X(Z)}{|\mathcal{T}|}$$

is an unbiased estimator for the count of (p, q)-bicliques in G.

Proof. For a random zstar Z sampled from $\Delta(G)$, we have

$$\mathbb{E}[X(Z)] = \Pr(Z \text{ forms a } (p,q)\text{-biclique in } G) = cnt_{p,q}(G)/|\Delta(G)|$$

By the linearity of expectation, we get:

$$\mathbb{E}\left[\widehat{cnt}_{p,q}(G)\right] = \mathbb{E}\left[\frac{|\Delta(G)|}{|\mathcal{T}|}\sum_{Z\in\mathcal{T}} X(Z)\right] = \frac{|\Delta(G)|}{|\mathcal{T}|}\sum_{Z\in\mathcal{T}}\mathbb{E}[X(Z)] = cnt_{p,q}(G),$$

proving that $\widehat{cnt}_{p,q}(G)$ is an unbiased estimator for the number of (p,q)-bicliques in G. □

Following the above discussions, we can define the sample space $\mathcal{S}_{p,q}(G)$ as $\Delta(G)$, the set of all h-zstars in G for $h = p$, and invoke Algorithm 1 to obtain an accurate estimation of $cnt_{p,q}(G)$.

3.1.1 Comparing with ZigZag. A sampling structure similar to but distinct from zstar, called zigzag, was proposed in [34].

Definition 3 (zigzag). *Given a bipartite graph G and an integer h, an h-*zigzag *in G is an ordered path $(u_{i_1}, v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_h}, v_{j_h})$ such that $i_1 < i_2 < \cdots < i_h$ and $j_1 < j_2 < \cdots < j_h$.*

We remark that when $p = q$, our zstar coincides with zigzag. However, when $p \neq q$, the two structures are distinct: a zstar is not a zigzag, and vice versa. In particular, an h-zstar contains $2h + q - p$ vertices while an h-zigzag contains $2h$ vertices. For example, for the graph in Figure 1 and $p = q$, the vertex sequence $(u_1, v_1, u_2, v_2, u_3, v_3)$ is both a 3-zstar and a 3-zigzag. When $q - p = 2$, neither of the two 2-zstars shown in Figures 2 and 3 is a 2-zigzag; conversely, the sequence (u_1, v_1, u_2, v_2) forms a 2-zigzag, but not a 2-zstar.

Each (p,q)-biclique in G with $p \leq q$ contains $\binom{q}{p}$ h-zigzags for $h = p$ (in contrast to our Lemma 1), and a zigzag may be contained in multiple (p,q)-bicliques (in contrast to Equation (2)). Specifically, for an h-zigzag Z with $h = p$, let $Z_U = Z \cap U$, the number of (p,q)-bicliques in G containing Z is

$$c_{p,q}(Z) = \begin{cases} \binom{|\Lambda(Z_U)|-p}{q-p}, & \text{if } Z \text{ forms a biclique} \\ 0, & \text{otherwise} \end{cases}$$

Let $\mathcal{H}$ be the set of all h-zigzags in G for $h = p$, and $\mathcal{T}$ be a set of zigzags uniformly sampled from $\mathcal{H}$. It is shown in [34] that the following quantity provides an unbiased estimator for $cnt_{p,q}(G)$:

$$\frac{|\mathcal{H}|}{\binom{q}{p}} \cdot \frac{\sum_{Z\in\mathcal{T}} c_{p,q}(Z)}{|\mathcal{T}|}$$

However, the sample space $\mathcal{S}_{p,q}(G) = \mathcal{H}$ is incompatible with Algorithm 1. That is, invoking Algorithm 1 with sample space $\mathcal{H}$ cannot guarantee the estimation accuracy. This is because the random variables $c_{p,q}(Z)$ take large values that are greater than 1. Note that, a fixed number of samples is taken from $\mathcal{H}$ in [34].

3.1.2 Counting and Sampling zstars. As discussed earlier, our sample space $\Delta(G)$ consists of all h-zstars in G for $h = p$. However, explicitly storing these h-zstars is computationally infeasible due to their sheer number. Consequently, we do not store the sample space directly. Instead, to efficiently sample a zstar as required in Line 3 of Algorithm 1, we compute and store the counts of zstars of various lengths and originating from different edges. Let $dp[i][e(u,v)]$ represent the number of zstars of length i starting from $e(u,v)$, where $i \in [2, 2h-1]$. Note that, for the zstar defined in Definition 2, the first two vertices u_{i_1} and v_{j_1} should be connected by an edge, and we say that the zstar starts from the edge $e(u_{i_1}, v_{j_1})$. Then, the total number of h-zstars in G for $h = p$ is $\sum_{e(u,v)\in E} dp[2p-1][e(u,v)]$. To compute $dp[i][e(u,v)]$, we need $dp[i-1][e(v,u')]$, for $u' \in N(v)$. Thus, we revise Definition 2 to also include zstars where the first vertex is from V; that is, the

Algorithm 2: Count-zstar(G, p, q)

Input: A bipartite graph G and integers p and q satisfying $p \leq q$
Output: Dynamic programming table $dp[\cdot][\cdot]$

```
for each edge e(v,u) ∈ E do
    N_{>v}(u) ← the set of u's neighbors that rank higher than v;
    dp[2][e(v,u)] ← C(|N_{>v}(u)|, q−p+1);
for i ← 3 to 2p − 1 do
    if i%2 = 1 then
        for each edge e(u,v) ∈ E do
            dp[i][e(u,v)] ← Σ_{u'∈N_{>u}(v)} dp[i−1][e(v,u')];
    else
        for each edge e(v,u) ∈ E do
            dp[i][e(v,u)] ← Σ_{v'∈N_{>v}(u)} dp[i−1][e(u,v')];
return dp;
```

subsequence $(v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_h}, v_{j_h}, v_{j_{h+1}}, \ldots, v_{j_{h+q-p}})$ satisfying the conditions in Definition 2 is a zstar of length $2h-2$ starting from $e(v_{j_1}, u_{i_2})$. $dp[i](e(u,v)]$ and $dp[i][e(v,u)]$ values can be computed by dynamic programming with the following recursion equations:

$$\begin{cases} dp[i][e(u,v)] = \sum_{u' \in N_{>u}(v)} dp[i-1][e(v,u')], & \text{if } i > 2 \\ dp[i][e(v,u)] = \sum_{v' \in N_{>v}(u)} dp[i-1][e(u,v')], & \text{if } i > 2 \\ dp[i][e(v,u)] = \binom{|N_{>v}(u)|}{q-p+1}, & \text{if } i = 2 \end{cases} \tag{3}$$

where $N_{>v}(u)$ is the set of u's neighbors that rank higher than v. We emphasize that $dp[i][e(u,v)]$ and $dp[i][e(v,u)]$ are distinct and should not be conflated. In particular, $dp[i][e(u,v)]$ is defined only for odd values of i, whereas $dp[i][e(v,u)]$ is defined exclusively for even values of i. Algorithm 2 outlines the steps for counting zstars following Equation (3). It takes as input a bipartite graph G and two integers p and q, and produces a dynamic programming table that records the zstar counts starting from each edge in G, with zstar lengths ranging from 2 to $2p-1$.

Now, we present an efficient method to uniformly sample an h-zstar Z for $h = p$ from the bipartite graph G, leveraging the dynamic programming counts obtained above. The central challenge lies in establishing the correct probability distribution over all possible zstars in G. By utilizing the counts computed in the dynamic programming table, we can construct this probability distribution effectively. Recall that, the total number of h-zstars for $h = p$ in G is given by: $|\Delta(G)| = \sum_{e(u,v) \in E} dp[2p-1][e(u,v)]$, where $dp[2p-1][e(u,v)]$ is the number of zstars of length $2p-1$ starting from edge $e(u,v)$. This total count allows us to define the probability of an edge $e(u,v)$ being the starting edge of Z as:

$$\Pr(e(u,v)) = \frac{dp[2p-1][e(u,v)]}{\sum_{e(u',v') \in E} dp[2p-1][e(u',v')]}$$

Once the starting edge $e(u_{i_1}, v_{j_1})$ is selected, the subsequent edges in the zstar Z can be determined recursively. Specifically, for the next edge $e(v_{j_1}, u')$ where $u' \in N_{>u_{i_1}}(v_{j_1})$ (the neighbors of v_{j_1} with a higher rank than u_{i_1}), the conditional probability of choosing $e(v_{j_1}, u')$ given that $e(u_{i_1}, v_{j_1})$ is the current edge is:

$$\Pr\left(e(v_{j_1}, u') \mid e(u_{i_1}, v_{j_1})\right) = \frac{dp[2p-2][e(v_{j_1}, u')]}{dp[2p-1][e(u_{i_1}, v_{j_1})]}.$$

Algorithm 3: Sample-zstar(G, p, q, dp)

Input: A bipartite graph G, two integers p and q satisfying $p \le q$, and dynamic programming table dp
Output: A uniformly sampled zstar Z

1 Set the distribution $\mathcal{D}$ over all edges of G where $\Pr(e(u,v)) = \frac{dp[2p-1][e(u,v)]}{\sum_{e(u',v')\in E} dp[2p-1][e(u',v')]}$;
2 $e(u_{i_1}, v_{j_1}) \leftarrow$ sample an edge according to $\mathcal{D}$;
3 $k \leftarrow 1$;
// Recursively construct zstar
4 **while** $k < p$ **do**
// Sample an edge from V to U
5 $E' \leftarrow \{e(v_{j_k}, u') \mid u' \in N_{>u_{i_k}}(v_{j_k})\}$;
6 Set the distribution $\mathcal{D}$ over E' where $\Pr(e(v_{j_k}, u')) = \frac{dp[2p-2k][e(v_{j_k},u')]}{dp[2p-2k+1][e(u_{i_k},v_{j_k})]}$;
7 Sample an edge $e(v_{j_k}, u_{i_{k+1}})$ according to $\mathcal{D}$;
8 $k \leftarrow k + 1$;
9 **if** $k < p$ **then**
// Sample an edge from U to V;
10 $E' \leftarrow \{e(u_{i_k}, v') \mid v' \in N_{>v_{j_{k-1}}}(u_{i_k})\}$;
11 Set the distribution $\mathcal{D}$ over E' where $\Pr(e(u_{i_k}, v')) = \frac{dp[2p-2k+1][e(u_{i_k},v')]}{dp[2p-2k+2][e(v_{j_{k-1}},u_{i_k})]}$;
12 Sample an edge $e(u_{i_k}, v_{j_k})$ according to $\mathcal{D}$;
// Add remaining vertices from V
13 Uniformly sample $q - p + 1$ vertices $\{v_{j_p}, v_{j_{p+1}}, \ldots, v_{j_q}\}$ from $N_{>v_{j_{p-1}}}(u_{i_p})$ without replacement;
14 **return** $Z = (u_{i_1}, v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_p}, v_{j_p}, v_{j_{p+1}}, \ldots, v_{j_q})$;

Recall from Equation (3) that $dp[2p-1][e(u,v)] = \sum_{u'\in N_{>u}(v)} dp[2p-2][e(v,u')]$. This recursive method continues for each subsequent edge in the zstar, decrementing the length from $2p-1$ down to 2. After this, we have sampled a path $(u_{i_1}, v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_p})$. To complete the zstar, we need to sample the remaining $q-p+1$ vertices from V. Given the last edge $e(v_{j_{p-1}}, u_{i_p})$ in the sampled path, we proceed by sampling $q-p+1$ additional vertices $\{v_{j_p}, v_{j_{p+1}}, \ldots, v_{j_q}\}$ from the set $N_{>v_{j_{p-1}}}(u_{i_p})$. To maintain uniformity, each vertex $v' \in N_{>v_{j_{p-1}}}(u_{i_p})$ is sampled with equal probability, where $\Pr(v') = \frac{1}{|N_{>v_{j_{p-1}}}(u_{i_p})|}$. We summarize these steps in Algorithm 3.

Theorem 3. *The time complexity of Algorithm 3 is $O(|E| + p \cdot d_{\max})$.*

Proof. Setting the initial distribution over the edges of G in Line 1 requires $O(|E|)$ time. The recursive construction of the zstar runs for $2p-3$ iterations, where each iteration processes up to $d_{\max}$ neighbors. These recursion steps take $O(p \cdot d_{\max})$ time in total. Since selecting the remaining $q-p+1$ vertices does not dominate the time complexity, the overall time complexity is $O(|E| + p \cdot d_{\max})$. □

3.2 Refining the Sample Space

Although the sample space defined in Section 3.1 is much smaller than the naive sample space $\{(B_U, B_V) \mid B_U \subseteq U, |B_U| = p, B_V \subseteq V, |B_V| = q\}$ discussed in Section 2.1, it is still very large for large bipartite graphs. As not all zstars correspond to bicliques, we propose to further reduce the sample space size by removing unpromising zstars in this subsection. To this end, we introduce

a novel edge-oriented sample space refinement technique, termed *BC-Shadow*, inspired by the shadow concept originally introduced for k-clique counting [13] and formally defined in [6].

Definition 4 (BC-Shadow). *Given a bipartite graph $G = (U, V, E)$ and two integers p and q, the BC-Shadow, denoted $\mathbb{S}_{p,q}(G)$, is a set of quadruples (R_U, R_V, S_U, S_V) representing a sample space, with the following properties:*

(1) *For each $(R_U, R_V, S_U, S_V) \in \mathbb{S}_{p,q}(G)$, $R_U \subseteq U$ and $R_V \subseteq V$ such that (R_U, R_V) forms a biclique in G;*
(2) *For each $(R_U, R_V, S_U, S_V) \in \mathbb{S}_{p,q}(G)$, $S_U \subseteq U \setminus R_U$ and $S_V \subseteq V \setminus R_V$ such that each vertex of S_U is fully adjacent to R_V (i.e., $S_U \subseteq \Lambda(R_V)$) and each vertex of S_V is fully adjacent to R_U;*
(3) *For every (p, q)-biclique (B_U, B_V) in G, there exists a unique quadruple $(R_U, R_V, S_U, S_V) \in \mathbb{S}_{p,q}(G)$ such that $R_U \subseteq B_U$, $R_V \subseteq B_V$, $B_U \setminus R_U \subseteq S_U$, and $B_V \setminus R_V \subseteq S_V$.*

Each quadruple (R_U, R_V, S_U, S_V) of the BC-Shadow $\mathbb{S}_{p,q}(G)$ represents a disjoint ***sample subspace*** of $\mathcal{S}_{p,q}(G)$, and the union of (R_U, R_V) and each $(p - |R_U|, q - |R_V|)$-biclique in the subgraph $G[S_U, S_V]$ of G induced by vertices (S_U, S_V) produces a unique (p, q)-biclique of G. Let $cnt_{p-|R_U|,q-|R_V|}(S_U, S_V)$ be the number of $(p - |R_U|, q - |R_V|)$-bicliques in $G[S_U, S_V]$. Then,

$$cnt_{p,q}(G) = \sum_{(R_U,R_V,S_U,S_V)\in\mathbb{S}_{p,q}(G)} cnt_{p-|R_U|,q-|R_V|}(S_U, S_V)$$

The BC-Shadow $\mathbb{S}_{p,q}(G)$ is a compact representation of $\mathcal{S}_{p,q}(G)$. Assuming that $|R_U| = |R_V|$ for each subspace which is the case in this paper, extending each h-zstar defined in (S_U, S_V) with vertices of R_U and R_V results in an $(h + |R_U|)$-zstar. Let $\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)$ be the set of all h-zstars for $h = p - |R_U|$ in $G[S_U, S_V]$. Without loss of generality, we also use $\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)$ to denote the results of extending these zstars by R_U and R_V. Then, the sample space $\mathcal{S}_{p,q}(G)$ defined by $\mathbb{S}_{p,q}(G)$ is

$$\bigcup_{(R_U,R_V,S_U,S_V)\in\mathbb{S}_{p,q}(G)} \mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)$$

which satisfies the following conditions:

- $\mathcal{S}_{p,q}(G)$ is a superset of the set of all (p, q)-bicliques in G, i.e., $\mathcal{B}_{p,q}(G) \subseteq \mathcal{S}_{p,q}(G)$;
- The cardinality of $\mathcal{S}_{p,q}(G)$ can be computed efficiently from the cardinalities of the sample subspaces $\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)$;
- We can efficiently sample a zstar uniformly at random (u.a.r.) from $\mathcal{S}_{p,q}(G)$.

Specifically, to sample a zstar u.a.r. from $\mathcal{S}_{p,q}(G)$, we first sample a subspace (R_U, R_V, S_U, S_V) from $\mathcal{S}_{p,q}(G)$ with probability proportional to $|\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)|$. Then, we sample a zstar u.a.r. from $\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)$ using the techniques described in Section 3.1. Note that, the value of $|\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)|$ is also computed via the method outlined in Section 3.1.

The BC-Shadow of the sample space defined in Section 3.1 consists of a single subspace $(\emptyset, \emptyset, U, V)$. Replacing this with multiple subspaces reduces the size of the sample space, as each subspace introduces additional constraints (specifically, Property (2) in Definition 4). To illustrate, consider the subspace $(\{u_{i_1}\}, \{v_{j_1}\}, S_U, S_V)$. Any h-zstar for $h = p - 1$ defined over (S_U, S_V) — for example, $(u_{i_2}, v_{j_2}, \ldots, u_{i_p}, v_{j_p}, v_{j_{p+1}}, \ldots, v_{j_q})$ — can be extended by prepending (u_{i_1}, v_{j_1}) to form a valid h-zstar for $h = p$ defined in G. In addition to satisfying the constraints specified in Definition 2, this extended zstar must satisfy the following two additional constraints:

- $\{v_{j_2}, \ldots, v_{j_q}\} \subseteq N_{>v_{j_1}}(u_{i_1})$ and
- $\{u_{i_2}, \ldots, u_{i_p}\} \subseteq N_{>u_{i_1}}(v_{j_1})$.

Moreover, these constraints become increasingly restrictive as $|R_U|$ and $|R_V|$ grow larger. However, larger values of $|R_U|$ and $|R_V|$ also lead to increased sample space construction time.

Algorithm 4: BCFramework($G, p, q, \epsilon, \delta$)

Input: A bipartite graph G, two integers p and q, and error parameters $\epsilon \in (0,1)$ and $\delta \in (0,1)$
Output: An estimation $\widehat{cnt}_{p,q}(G)$ of $cnt_{p,q}(G)$

`// Stage-I: construct a sample space` $\mathcal{S}_{p,q}(G)$, `represented by a BC-Shadow` $\mathbb{S}_{p,q}(G)$

1 Arrange vertices of U and V in some order;
2 $\mathbb{S}_{p,q}(G) \leftarrow \{(\emptyset, \emptyset, U, V)\}$;
3 **while** *the construction stopping condition is not satisfied* **do**
4 Choose a sample subspace (R_U, R_V, S_U, S_V) and remove it from $\mathbb{S}_{p,q}(G)$;
`// Refine the subspace` (R_U, R_V, S_U, S_V) `by partitioning it`
5 **for each** *edge* $e(u,v) \in G[S_U, S_V]$ **do**
6 Add $(R_U \cup u, R_V \cup v, N_{>u}(v) \cap S_U, N_{>v}(u) \cap S_V)$ to $\mathbb{S}_{p,q}(G)$;

`// Stage-II: sample from` $\mathbb{S}_{p,q}(G)$ `to get` $\widehat{cnt}_{p,q}(G)$

7 $|\mathcal{S}_{p,q}(G)| \leftarrow \sum_{(R_U,R_V,S_U,S_V)\in\mathbb{S}_{p,q}(G)} |\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)|$;
8 $s \leftarrow 0$; $t \leftarrow 0$; $\gamma \leftarrow 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$;
9 **while** $s < \gamma$ **do**
10 Sample a subspace (R_U, R_V, S_U, S_V) from $\mathbb{S}_{p,q}(G)$ with probability $\frac{|\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U,S_V)|}{|\mathcal{S}_{p,q}(G)|}$;
11 Sample a zstar Z from $\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)$ u.a.r. by invoking Algorithm 3;
12 **if** *Z forms a biclique in G* **then** $s \leftarrow s + 1$;
13 $t \leftarrow t + 1$;
14 **return** $\widehat{cnt}_{p,q}(G) \leftarrow |\mathcal{S}_{p,q}(G)| \cdot \frac{s}{t}$;

Following the above discussions, the general framework for estimating the (p,q)-biclique count based on the BC-Shadow is presented in Algorithm 4, which operates in two stages. In Stage-I (Lines 1–6), the BC-Shadow is first initialized as $\{(\emptyset, \emptyset, U, V)\}$ and then continuously refined until a stopping condition is met. During this refinement process, R_U and R_V expand while S_U and S_V shrink, leading to an increase in biclique density within the sample space. Details of Stage-I will be described in Section 3.3. In Stage-II (Lines 7—14), random samples are drawn from the sample space $\mathcal{S}_{p,q}(G)$, and the (p,q)-biclique count $cnt_{p,q}(G)$ is estimated as $|\mathcal{S}_{p,q}(G)|$ multiplied by the fraction of samples that form (p,q)-bicliques. This stage resembles Algorithm 1, but incorporates additional details specifically tailored to the BC-Shadow. As a result, the accuracy of the estimated biclique count follows from Theorem 1.

Theorem 4. *With probability at least $1-\delta$, the biclique count estimated by Algorithm 4 has a relative error at most ϵ.*

3.3 Balancing the Running Time of the Two Stages

Recall that, our framework in Algorithm 4 for estimating the (p,q)-biclique count consists of two stages: Stage-I constructs the sample space $\mathcal{S}_{p,q}(G)$, represented by a BC-Shadow $\mathbb{S}_{p,q}(G)$, while Stage-II samples zstars from $\mathcal{S}_{p,q}(G)$ and estimates $cnt_{p,q}(G)$ based on the stopping rule theorem (Theorem 1). Different sample spaces $\mathcal{S}_{p,q}(G)$ yield different biclique densities $\mu_{\mathcal{S}_{p,q}(G)}$, which in turn affect the running time of Stage-II of the algorithm. In this subsection, we explore methods for constructing $\mathcal{S}_{p,q}(G)$ that aim to optimize the overall performance of the algorithm by balancing the computational workload between Stage-I and Stage-II.

3.3.1 Balancing Strategy. As shown in Algorithm 4, our sample space construction algorithm first initializes the BC-Shadow as $\{(\emptyset, \emptyset, U, V)\}$ and then iteratively refines it. If we spend minimal time refining the sampling space (e.g., stop Stage-I immediately after the initialization), then Stage-I will be very efficient. However, the sample space remains large and contains many elements that do not form bicliques, resulting in a low $\mu_{\mathcal{S}_{p,q}(G)}$ and causing Stage-II to require a large number of samples to achieve the desired estimation accuracy. Conversely, if we over-invest time in refining $\mathcal{S}_{p,q}(G)$, Stage-I becomes time-consuming and may dominate the total running time. To address this, we propose a strategy that dynamically balances the running time between the two stages by determining an optimal point at which to stop refining the sample space and begin sampling. This balance ensures that neither stage disproportionately dominates the total running time, optimizing the algorithm's overall efficiency.

To determine when to stop sample space refinement and proceed to Stage-II, we propose to estimate the running time of Stage-II for the current sample space without actually executing it. We observe that the running time of Stage-II depends on:

- **The number of samples required**, which depends on the desired estimation accuracy and the biclique density of the sample space.
- **The average time per sample** T_{sample}, which is the time taken to draw a sample from $\mathcal{S}_{p,q}(G)$ and verify whether it forms a biclique.

Following the stopping rule theorem [9] (Theorem 1), the expected number of samples t drawn before termination is approximately

$$t \approx \frac{\gamma}{\mu_{\mathcal{S}_{p,q}(G)}}$$

where $\gamma = 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$. Consequently, the expected running time of Stage-II is given by:

$$\text{Expected Running Time of Stage-II} \approx \frac{\gamma}{\mu_{\mathcal{S}_{p,q}(G)}} \times T_{\text{sample}} \tag{4}$$

This equation indicates that as $\mu_{\mathcal{S}_{p,q}(G)}$ increases, the expected running time of Stage II decreases, and vice versa.

Among the quantities used in Equation (4), we can compute γ since ϵ and δ are given as input parameters, and T_{sample} can be estimated by sampling a few elements from $\mathcal{S}_{p,q}(G)$ and measuring the average sampling time. However, estimating $\mu_{\mathcal{S}_{p,q}(G)}$ is challenging since it is the very quantity we aim to estimate. To address this challenge, we propose to compute an auxiliary estimation $\tilde{\mu}$ for $\mu_{\mathcal{S}_{p,q}(G)}$. Unlike the final estimate, $\tilde{\mu}$ does not require strict theoretical accuracy guarantees, as its sole purpose is to estimate the running time, rather than affecting the accuracy of the final biclique count estimation. However, in order to compute $\tilde{\mu}$ effectively, certain conditions must be met. Specifically, we require a sampling structure where each biclique in the graph corresponds uniquely to a sampling structure, ensuring that $\tilde{\mu}$ remains within the range $[0, 1]$. Our novel sampling structure facilitates this, enabling us to maintain this condition and thereby compute $\tilde{\mu}$ efficiently.

3.3.2 Construction Stopping Condition. Based on the above discussions, our sample space construction stopping condition is:

$$\text{Elapsed Time in Stage-I} \geq \gamma \times \frac{T_{\text{sample}}}{\tilde{\mu}}$$

where T_{sample} is the estimated average time per sample, and $\tilde{\mu}$ is the auxiliary estimation of the biclique density.

3.3.3 Computing the Auxiliary Estimation $\tilde{\mu}$. As the sample space $\mathcal{S}_{p,q}(G)$ changes dynamically during refinement, we need to continuously estimate $\tilde{\mu}$. For each subspace $(R_U, R_V, S_U, S_V) \in \mathbb{S}_{p,q}(G)$, we estimate the biclique density μ' within that subspace by:

- Sampling a small number of zstars uniformly at random from $\mathcal{P}_{p',q'}(S_U, S_V)$, where $p' = p - |R_U|$ and $q' = q - |R_V|$.
- Calculating μ' as the proportion of these samples that form bicliques in G.
- Estimating the biclique count in the subspace as

$$cnt_{p',q'}(S_U, S_V) \approx |\mathcal{P}_{p',q'}(S_U, S_V)| \times \mu'.$$

We store μ' along with the subspace and update $\widetilde{cnt}_{p,q}(G)$, which is used for estimating $\tilde{\mu}$, and the sample space size accordingly:

$$\widetilde{cnt}_{p,q}(G) = \sum_{(R_U,R_V,S_U,S_V,\mu') \in \mathbb{S}_{p,q}(G)} |\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)| \times \mu'$$

$$|\mathcal{S}_{p,q}(G)| = \sum_{(R_U,R_V,S_U,S_V,\mu') \in \mathbb{S}_{p,q}(G)} |\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)|$$

The auxiliary estimation of the biclique density is then:

$$\tilde{\mu} = \frac{\widetilde{cnt}_{p,q}(G)}{|\mathcal{S}_{p,q}(G)|}$$

3.3.4 The Pseudocode of BC-Shadow Construction. Based on the above discussions, the pseudocode of our BC-Shadow construction algorithm is presented in Algorithm 5. It takes a bipartite graph $G = (U, V, E)$, two integers p and q, and error parameters $\epsilon \in (0, 1)$ and $\delta \in (0, 1)$ as input, and outputs a BC-Shadow $\mathbb{S}_{p,q}(G)$ that compactly represents the sample space $\mathcal{S}_{p,q}(G)$.

The initialization phase begins by arranging the vertices of U and V in some order (Line 1). The initial count $\widetilde{cnt}_{p,q}(G)$ is set to 1, the size of the sample space $|\mathcal{S}_{p,q}(G)|$ is initialized to $|\mathcal{P}_{p,q}(G)|$, and the value of T_{sample} is initially set to infinity (Line 2). The BC-Shadow $\mathbb{S}_{p,q}(G)$ is initialized with the full vertex sets U and V, along with the initial estimation μ' (Line 3).

In the refinement loop (Lines 4—21), the algorithm continues refining the BC-Shadow as long as the elapsed time is less than the estimated running time of Stage-II (Line 4). At each iteration, the subspace $(R_U, R_V, S_U, S_V, \mu)$ with the smallest μ is selected for refinement (Line 5–6). The algorithm updates $\widetilde{cnt}_{p,q}(G)$ and the size of the sample space $|\mathcal{S}_{p,q}(G)|$ by removing the contribution of the selected subspace (Lines 7–8). For each edge (u, v) in the subgraph $G[S_U, S_V]$ of G induced by vertices (S_U, S_V), a new subspace is created by moving u and v to R_U and R_V, respectively (Line 11). Let (R'_U, R'_V, S'_U, S'_V) be the newly created subspace; note that, $S'_U = N_{>u}(v) \cap S_U$ and $S'_V = N_{>v}(u) \cap S_V$. The dynamic programming table dp is computed for the subspace by invoking Algorithm 2 (Line 12); note that this also computes $|\mathcal{P}_{p-|R'_U|,q-|R'_V|}(S'_U, S'_V)|$. Then, a small number $\hat{n}$ of samples is drawn from $\mathcal{P}_{p-|R'_U|,q-|R'_V|}(S'_U, S'_V)$ by invoking Algorithm 3 (Line 13), which is used for estimating the biclique density μ' in the new subspace (Line 14); we set $\hat{n} = \frac{|\mathcal{P}_{p-|R'_U|,q-|R'_V|}(S'_U,S'_V)|}{(p-|R'_U|)\cdot(q-|R'_V|)}$ in our implementation. Then, the subspace (R'_U, R'_V, S'_U, S'_V) is added to the BC-Shadow $\mathbb{S}_{p,q}(G)$ (Line 15), and the values of $\widetilde{cnt}_{p,q}(G)$ and $|\mathcal{S}_{p,q}(G)|$ are updated accordingly (Lines 16–17). Also, during the refinement of the initial subspace $(\emptyset, \emptyset, U, V)$, the algorithm estimates T_{sample} based on the total sampling time and total number of samples (Lines 9 and 18—21). This approach ensures efficient refinement of the subspaces while maintaining computational feasibility during the sampling process.

Algorithm 5: BC-ShadowConstruction($G, p, q, \epsilon, \gamma$)

Input: A bipartite graph G, two integers p and q, and error parameters $\epsilon \in (0,1)$ and $\delta \in (0,1)$
Output: A BC-Shadow $\mathbb{S}_{p,q}(G)$

// This algorithm replaces Lines 1–6 of Algorithm 4
1 Arrange vertices of U and V in some order;;
2 $\widetilde{cnt}_{p,q}(G) \leftarrow 1$; $|\mathcal{S}_{p,q}(G)| \leftarrow |\mathcal{P}_{p,q}(G)|$; $T_{\text{sample}} \leftarrow \infty$;
3 $\mathbb{S}_{p,q}(G) \leftarrow \{(\emptyset, \emptyset, U, V, \mu' = \widetilde{cnt}_{p,q}(G)/|\mathcal{S}_{p,q}(G)|)\}$;
4 **while** *ElapsedTime()* $< \gamma \cdot \frac{|\mathcal{S}_{p,q}(G)|}{\widetilde{cnt}_{p,q}(G)} \cdot T_{sample}$ **do**
5 $(R_U, R_V, S_U, S_V, \ddot{\mu}) \leftarrow \arg\min_{(R'_U, R'_V, S'_U, S'_V, \ddot{\mu}') \in \mathbb{S}_{p,q}(G)} \ddot{\mu}'$;
6 Remove $(R_U, R_V, S_U, S_V, \ddot{\mu})$ from $\mathbb{S}_{p,q}(G)$;
7 $\widetilde{cnt}_{p,q}(G) \leftarrow \widetilde{cnt}_{p,q}(G) - |\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)| \cdot \ddot{\mu}$;
8 $|\mathcal{S}_{p,q}(G)| \leftarrow |\mathcal{S}_{p,q}(G)| - |\mathcal{P}_{p-|R_U|,q-|R_V|}(S_U, S_V)|$;
9 **if** $R_U = \emptyset$ **then** $n_{sample} \leftarrow 0$; $T_{total} \leftarrow 0$;
10 **for each** *edge* $e(u,v) \in G[S_U, S_V]$ **do**
11 $(R'_U, R'_V, S'_U, S'_V) \leftarrow (R_U \cup u, R_V \cup v, N_{>u}(v) \cap S_U, N_{>v}(u) \cap S_V)$;
12 Compute the dynamic programming table dp for the subspace (R'_U, R'_V, S'_U, S'_V) by invoking Algorithm 2;
13 Sample $\hat{n}$ zstars from $\mathcal{P}_{p-|R'_U|,q-|R'_V|}(S'_U, S'_V)$ by invoking Algorithm 3;
14 $\ddot{\mu}' \leftarrow$ the fraction of the sampled $\hat{n}$ zstars that are bicliques;
15 Add $(R'_U, R'_V, S'_U, S'_V, \ddot{\mu}')$ to $\mathbb{S}_{p,q}(G)$;
16 $\widetilde{cnt}_{p,q}(G) \leftarrow \widetilde{cnt}_{p,q}(G) + |\mathcal{P}_{p-|R'_U|,q-|R'_V|}(S'_U, S'_V)| \cdot \ddot{\mu}'$;
17 $|\mathcal{S}_{p,q}(G)| \leftarrow |\mathcal{S}_{p,q}(G)| + |\mathcal{P}_{p-|R'_U|,q-|R'_V|}(S'_U, S'_V)|$;
18 **if** $R_U = \emptyset$ **then**
19 $n_{sample} \leftarrow n_{sample} + \hat{n}$;
20 $T_{total} \leftarrow T_{total}$ + running time of Line 13;
21 **if** $R_U = \emptyset$ **then** $T_{sample} \leftarrow \frac{T_{total}}{n_{sample}}$;
22 **return** $\mathbb{S}_{p,q}(G)$;

3.4 Vertex Ordering

As stated at the beginning of this section, we assume that the vertices on each side of the bipartite graph G are arranged in a particular order. Vertex ordering is conducted in Line 1 of Algorithms 4 and 5. There are various ways to order the vertices in U and V. Through experimental evaluation, we find that sorting them in increasing order by degree yields the best performance in our algorithm. Specifically, we order the vertices such that $d(u_1) \leq d(u_2) \leq \cdots \leq d(u_{|U|})$ for $u_i \in U$, and $d(v_1) \leq d(v_2) \leq \cdots \leq d(v_{|V|})$ for $v_j \in V$, where $d(u)$ and $d(v)$ denote the degree of vertices u and v, respectively. The main intuition is that in each of the refined subspaces (R_U, R_V, S_U, S_V), at least one of S_U and S_V will be small. Let (u,v) be the last edge used during the refinement process to obtain the subspace. If $d(u)$ is small, then S_V is small because $S_V \subseteq N(u)$. Similarly, if $d(v)$ is small, then S_U is small. Otherwise, both $d(u)$ and $d(v)$ are large, then both S_U and S_V are likely to be small because $S_U \subseteq N_{>u}(v)$ and $S_V \subseteq N_{>v}(u)$ and there are not many vertices with degree larger than $d(u)$ or $d(v)$. We refer to our algorithm (i.e., Algorithm 4 with Lines 1–6 replaced by Algorithm 5) with this vertex ordering as BC$^+$.

Table 2. Dataset statistics

Dataset	$\|U\|$	$\|V\|$	$\|E\|$
Android (rec-amz-Apps-for-Android)	1323884	61275	2638172
Books (rec-amz-Books)	8026324	2330066	22507155
Github (aff-github-user2project)	56519	120867	440237
Grocery (rec-amz-Grocery-Gourmet-Food)	768438	166049	1297156
Health (rec-amz-Health-Personal-Care)	1851132	252331	2982326
Stackexch (ia-stackexch-user-marks-post-und)	545196	96680	1302040
Vinyl (rec-amz-CDs-and-Vinyl)	1578597	486360	3749004
Wikiquote (ia-wikiquote-user-edits)	21607	92924	238714
Yelp (rec-yelp-user-business)	45982	11538	229906
bookcrossing (bookcrossing-full-rating)	105278	340523	1149739

4 Experiments

In this section, we evaluate the estimation accuracy and efficiency of our algorithm by comparing the following algorithms:

- **ZigZag** [34]: an existing approximation algorithm based on the zigzag sampling structure with $t = 50$K samples.
- **Z-Shadow** [5]: an existing approximation algorithm based on shadow with $t = 10$M samples.
- **BC$^+$**: our approximation algorithm with degree increasing vertex order, and error parameters $\epsilon = 0.05$ and $\delta = 0.05$.
- **Exact**: a state-of-the-art exact counting algorithm introduced in [34].

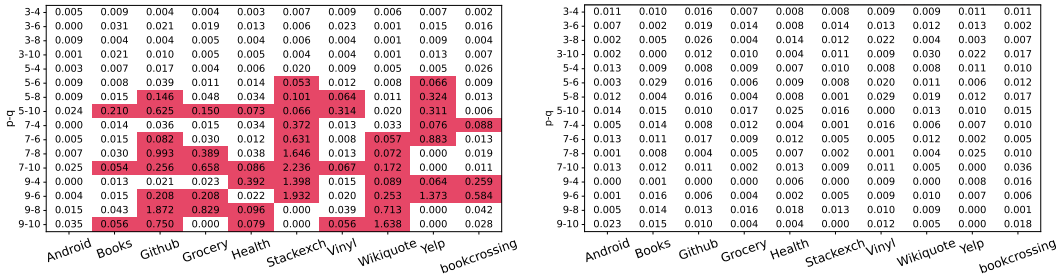
Note that, both ZigZag and Z-Shadow use a fixed number of samples and do not provide any guarantees on estimation accuracy. In contrast, our algorithm BC$^+$ takes accuracy parameters ϵ and δ as input, and dynamically determines the number of samples needed to meet the specified accuracy requirements. Additionally, we evaluate several variants of our algorithm to assess the effectiveness of our techniques. All algorithms are implemented in C++ and executed in a single-thread mode. We conduct the experiments on a machine equipped with an Intel Core i7-8700 CPU, 128GB main memory, and running Ubuntu 18.04.

We evaluate the algorithms on 10 real datasets, sourced from http://konect.cc. Table 2 provides statistics of the datasets, with shorthand names for ease of reference. Our experiments are conducted across various (p, q) configurations, with both p and q ranging from 3 to 10, reflecting the needs of real-world applications. Note that when $q > p$, we swap the two sides (U and V) of the input graph G to obtain the results.

4.1 Compare BC$^+$ with the Existing Algorithms

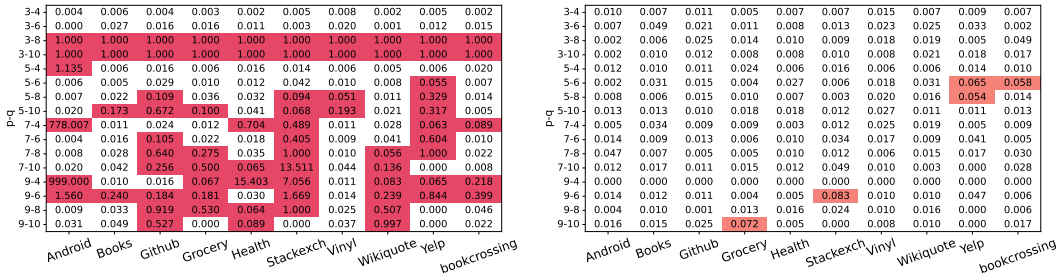
In this subsection, we compare our algorithm BC$^+$ against the existing algorithms regarding estimation stability and quality, and running time. Each testing is assigned a time limit of 4 hours.

4.1.1 Relative Standard Deviation Analysis. To evaluate the stability of the estimates of ZigZag and BC$^+$, we run each testing 10 times and measure the relative standard deviation (RSD) of the estimated biclique counts. Let $\widehat{cnt}^{(i)}$ be the estimated count of the i-th run, and $\widehat{cnt}_{avg} = \frac{1}{10}\sum_{i=1}^{10}\widehat{cnt}^{(i)}$. The RSD is defined as $\frac{\sigma}{\widehat{cnt}_{avg}}$, where $\sigma^2 = \frac{1}{9}\sum_{i=1}^{10}(\widehat{cnt}^{(i)} - \widehat{cnt}_{avg})^2$ is the sample variance. RSD quantifies the variability of the estimates and allows us to assess the consistency of each method. Figure 4(a) shows the RSD values of ZigZag for each dataset and (p, q) pair, with values exceeding 0.05 highlighted.



(a) RSD of ZigZag (b) RSD of BC$^+$

Fig. 4. Relative standard deviation of ZigZag and BC$^+$



(a) RE of ZigZag (b) RE of BC$^+$

Fig. 5. Relative error (RE) of ZigZag and BC$^+$

For small configurations (e.g., (3-4), (3-6), (5-4)), RSD remains consistently low, indicating stable estimates—though not necessarily accurate ones (discussed next). However, as (p, q) increases, RSD values become high. For instance, in Stackexch (7-10) and Github (9-8), RSD spikes to 2.236 and 1.872, respectively. These results reveal that ZigZag struggles to maintain consistency on large bicliques, likely due to insufficient sample size of 50K.

Figure 4(b) shows the RSD values of BC$^+$, demonstrating that it consistently maintains RSD values below 0.05 in every instance. This indicates that the adaptive sampling strategy employed by BC$^+$effectively controls the variance of estimated biclique counts, ensuring stable and reliable results across all datasets and (p, q) configurations. The consistently low RSD values imply that even a single run of our algorithm BC$^+$ is sufficient to obtain highly accurate estimates, eliminating the need for multiple repeated executions to reduce variance, making BC$^+$more efficient and practical for (p, q)-biclique counting.

4.1.2 Relative Error Analysis. We now assess the accuracy of the estimations by analyzing the relative error (RE). For an estimate $\widehat{cnt}^{(i)}$, its relative error is calculated as $\text{RE}^{(i)} = \frac{|\widehat{cnt}^{(i)} - cnt_{p,q}(G)|}{cnt_{p,q}(G)}$, where $cnt_{p,q}(G)$ is obtained using the Exact algorithm; we remark that obtaining the exact count requires significantly more computation time and may not be practical, as will be discussed in Section 4.1.3. As each testing is run 10 times, we report the RE as the average of the 10 runs, i.e., $\text{RE} = \frac{1}{10}\sum_{i=1}^{10} \text{RE}^{(i)}$. Figure 5(a) presents the RE values of ZigZag, where values exceeding 0.05 are highlighted in red. The results indicate that the RE is significantly high in multiple instances,

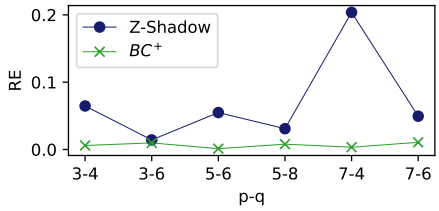
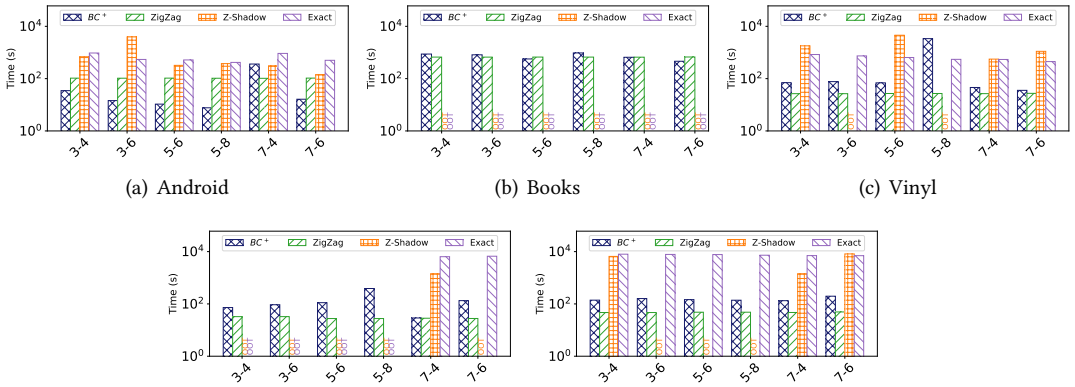


Fig. 6. Relative error of Z-Shadow and BC^+ on Android



(a) Android (b) Books (c) Vinyl

(d) Wikiquote (e) bookcrossing

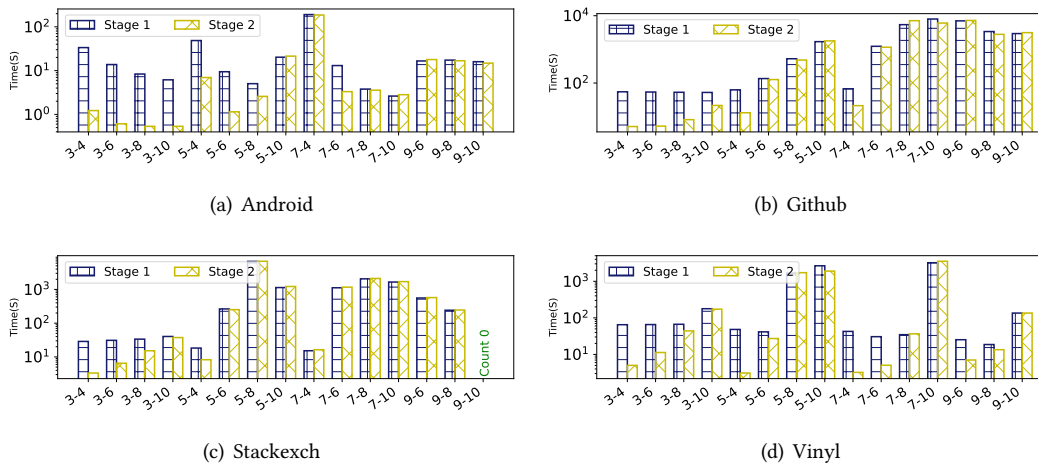
Fig. 7. Running time comparison with the existing algorithms

demonstrating the inaccuracy of ZigZag in providing biclique count estimations. For example, for the Android dataset, the RE for (9-4) and (7-4) is notably high, reflecting the method's inability to generate accurate estimates for these configurations. Comparing RE with RSD further highlights the weaknesses of ZigZag. In several cases, the algorithm maintains low RSD but exhibits high RE, indicating that while estimates are statistically stable, they are not accurate. This discrepancy is particularly evident in configurations such as (3-8) and (3-10), where RE remains high across all datasets despite low variance, making the results unreliable for practical use.

BC^+ (Fig. 5(b)) consistently achieves RE below 0.05 across nearly all datasets and configurations. Even in challenging settings such as Android (9-4) and (7-4), where ZigZag fails, BC^+ remains accurate. Note that while BC^+achieves low RE in most cases, there are a few instances where the RE surpasses 0.05, such as in (5-6) of Yelp and (9-6) of Stackexch, probably due to numerical instability issues. However, this remains within an acceptable margin.

Figure 6 shows the relative error (RE) of Z-Shadow for the Android dataset, which is the only dataset on which Z-Shadow was able to complete execution within the specified time limit for most of the tested (p, q) values, as explained in Section 4.1.3. It is clear that Z-Shadow causes fluctuations in the RE, while BC^+ consistently maintains the RE at a stable level.

4.1.3 Running Time Comparison with the Existing Algorithms. In this experiment, we evaluate the running time of BC^+ against existing algorithms, ZigZag, Z-Shadow, and Exact across six datasets, as summarized in Fig. 7. A time limit of four hours is imposed; any algorithm that fails to complete



(a) Android (b) Github

(c) Stackexch (d) Vinyl

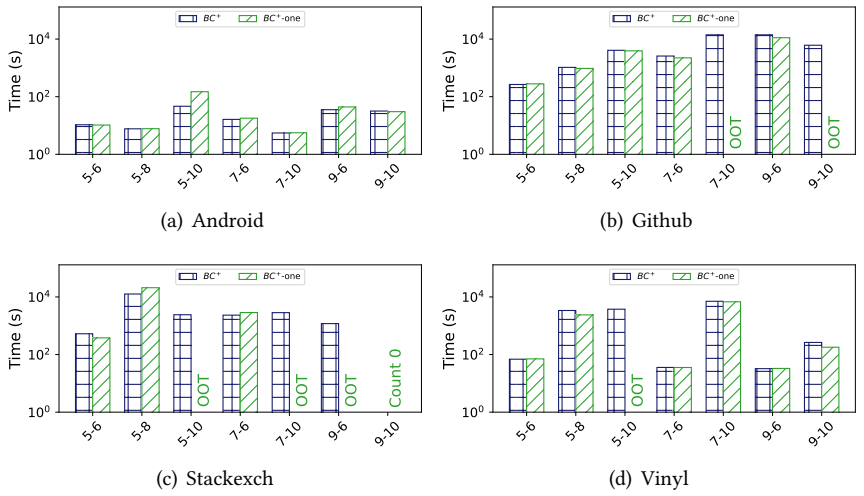
Fig. 8. Running time of the two stages of BC$^+$

within this time limit is marked as "OOT" (Out of Time). As shown, BC$^+$ consistently outperforms Exact across the majority of (p, q) configurations, regardless of graph size or density. This trend is particularly evident in the Books dataset, where Exact fails to complete any configuration within the time limit. Among all datasets, Z-Shadow completes all six (p, q) configurations only on the Android dataset within the four-hour limit (Fig. 7(a)). Even in cases where it does finish, Z-Shadow exhibits unstable performance, most notably on Android for $(7, 4)$, with large fluctuations in relative error (RE) (Fig. 6). This instability stems from its sampling mechanism, which selects p and ℓ vertices without considering structural constraints, leading to a high number of invalid samples and a low success rate, ultimately increasing the RE. ZigZag, which also uses a fixed number of samples (t), is evaluated with its default setting of 50K. On Android, BC$^+$ consistently outperforms ZigZag across all tested (p, q) configurations. In Books, both algorithms exhibit comparable runtimes. In some datasets such as Vinyl, ZigZag runs slightly faster than BC$^+$, likely due to its simple sample space construction and smaller fixed sample size. However, as discussed in earlier experiments, ZigZag's results are often unstable and lack accuracy in many of the tested scenarios. Despite its occasional speed advantage, its fixed-sample approach leads to unreliable performance across diverse graph structures.

4.2 Evaluate Our Techniques

In this subsection, we evaluate the effectiveness of our different techniques by comparing BC$^+$ with its variants. Note that, we set a time limit of 24 hours in these evaluations. As a result, we only show the results for a subset of the (p, q) values.

4.2.1 Running Time of the Two Stages of BC$^+$. Recall from Section 3.2 that BC$^+$ has two stages: Stage-I for sample space refinement, and Stage-II for zstar-based count estimation. Figure 8 presents the running time comparison between these two stages; note that, bars representing (p, q) values with zero biclique counts are omitted and indicated as *count 0*. The results demonstrate that the running time for the two stages are generally balanced across most (p, q) configurations. However, for small (p, q) such as (3,4) and (3,6), Stage-I occasionally exhibits longer running time due to the initial BC-shadow refinement. As can be seen from Algorithm 5, the initial BC-shadow will



(a) Android (b) Github

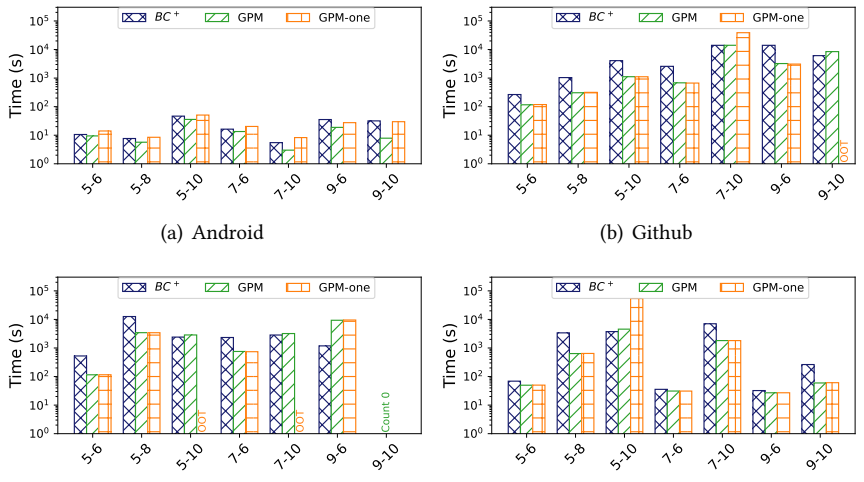
(c) Stackexch (d) Vinyl

Fig. 9. BC$^+$ with one refinement

be refined at least once. When the biclique size increases, the running time distribution between Stage I and II becomes more balanced. For instance, configurations such as (7,8) and (9,10) across all datasets reflect balanced running time between these stages, validating the efficiency of BC$^+$'s approach.

4.2.2 BC$^+$ with Only One Refinement. In this testing, we evaluate BC$^+$ against its variant that only refines the sample space once, denoted BC$^+$-one; that is, BC$^+$-one executes the while loop of Line 4 of Algorithm 5 exactly once. Everything else of BC$^+$-one is the same as BC$^+$. Their running time is shown in Figure 9. As illustrated, BC$^+$-one has a similar running time as BC$^+$ in many cases, indicating that a single refinement of the sample space is sufficient in these cases. However, there are also several cases (e.g., Github with (7,10) and (9,10), Stackexch with (5,10), (7,10), (9,6), and Vinyl with (5,10)) that BC$^+$-one runs significantly slower than BC$^+$; specifically, BC$^+$-one cannot finish within the time limit of 24 hours, as indicated by "OOT". Since BC$^+$ always performs the same number or more refinement steps than BC$^+$-one, its Stage-I is not faster. Consequently, the slower overall running time of BC$^+$-one is due to the greater number of samples taken during Stage-II. For example, for Android with (5,10), BC$^+$ draws 0.67M samples, while BC$^+$-one draws 2.98M samples. These results highlight the practical benefit of the continuous refinement and the balancing strategy employed by BC$^+$. By iteratively refining the sampling space, BC$^+$ is able to maintain a higher biclique density, leading to fewer required samples and faster convergence overall especially in challenging scenarios with large (p, q) values.

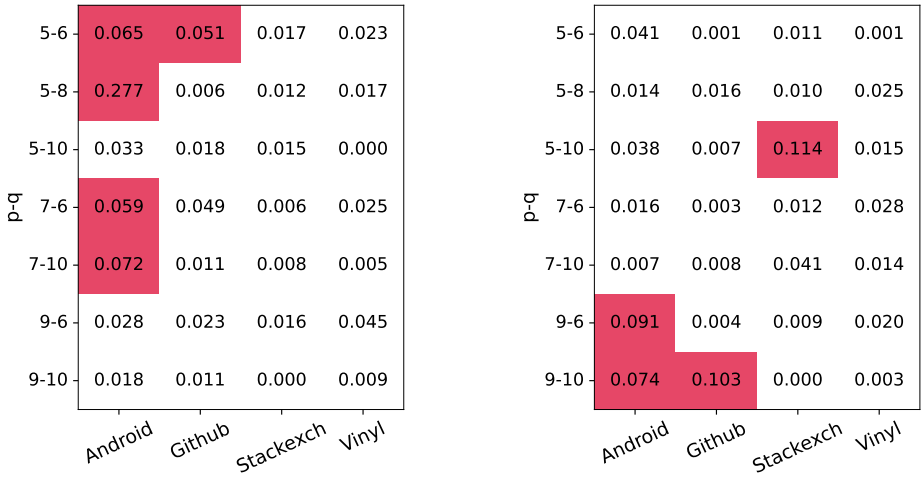
4.2.3 BC$^+$ with the Sampling Stopping Condition of ScaleGPM. As discussed at the end of Section 2.1, ScaleGPM [2] recently introduced an alternative sampling stopping condition by approximating the distribution of the empirical mean $\mu := \frac{1}{n}\sum_{i=1}^{n} X_i$ as a normal distribution. In this experiment, we evaluate a variant of BC$^+$ that adopts ScaleGPM's sampling stopping condition, which we refer to as GPM. Both BC$^+$ and GPM employ the same underlying techniques; the only difference lies in how the sampling process is terminated in Stage-II. The running time of BC$^+$ and GPM under the same accuracy parameters ($\epsilon = 0.05$ and $\delta = 0.05$) is shown in Figure 10. Overall, the two approaches perform comparably across datasets and (p, q) configurations, with GPM running slightly faster



(a) Android (b) Github

(c) Stackexch (d) Vinyl

Fig. 10. Running time of GPM and GPM-one

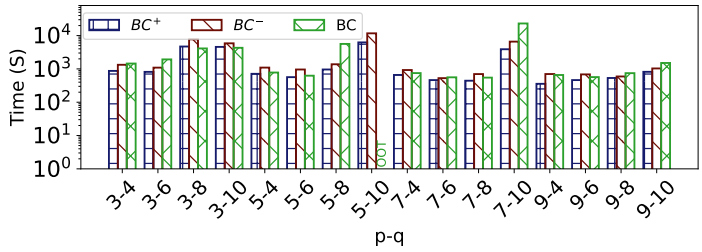


(a) Relative standard deviation (b) Relative error

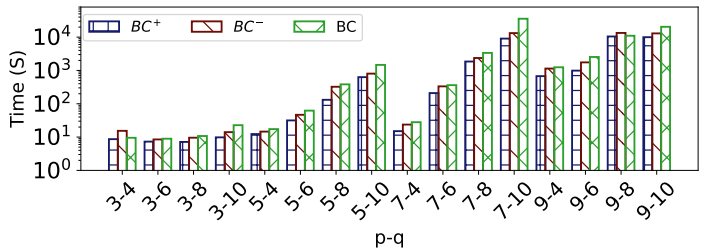
Fig. 11. RSD and RE of GPM

due to requiring fewer samples. However, since both share the same Stage-I, the performance gain of GPM over BC$^+$ is limited. Moreover, GPM exhibits higher RSD and RE, as shown in Figure 11. Notably, GPM is not guaranteed to run faster than BC$^+$ in all cases; for example, on Stackexch with (9,6), GPM is actually much slower.

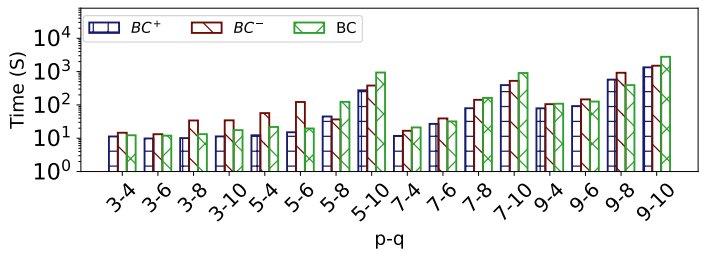
We further note that GPM benefits from our sample space refinement and balancing strategies. To assess their impact, we also evaluate a variant of GPM that performs only one refinement step (denoted GPM-one), analogous to BC$^+$-one introduced in Section 4.2.2. The running time of



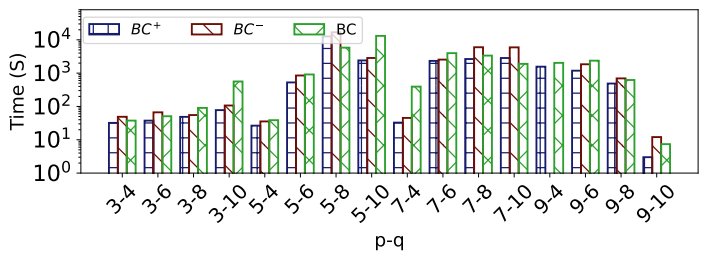
(a) Books



(b) Grocery



(c) Health



(d) Stackech

Fig. 12. Running time of other variants

GPM-one is also presented in Figure 10. Results show that GPM-one may run significantly slower than GPM and, in some cases, may even fail to complete within the time limit.

4.2.4 Other Variants of BC⁺. In this experiment, we explore another approach, termed vertex-oriented refinement (in Algorithm 5), which complements the edge-oriented method. Instead of refining based on edges, this approach adds a single vertex from S_U (or S_V) into R_U (or R_V) at each step. Consider the initial shadow where $R_U \cup R_V = \emptyset$, $S_U = U$, and $S_V = V$. In the first refinement step, a vertex $u \in S_U$ is added to R_U while keeping R_V empty, resulting in $S_U = N^2(u)$, i.e., $\bigcup_{v \in N(u)} N_{>u}(v)$ and $S_V = N(u)$. A primary limitation of the vertex-oriented approach is its potential to increase $|S_U|$, leading to a larger shadow size and consequently a more extensive sample space. To evaluate the impact of this approach, we implemented it as a variant, and refer to it as BC⁻, which also uses degree increasing vertex order.

Besides this, we also implement a variant of BC⁺ that uses degree decreasing vertex order, denoted BC.

Figure 12 shows the running time of the different variants of our algorithm. The BC⁺ variant consistently outperforms the other two variants (BC and BC⁻) in nearly every scenario, with exceptions noted for the Stackexch dataset specifically at biclique sizes (5-8) and (7-10). These exceptions may be attributed to instances where the two-hop neighbors of each node $u \in U$ are fewer, thus potentially making vertex-oriented refinements more optimized than oriented strategies in these two cases. It is also clearly evident that BC⁺ maintains an advantage over the BC⁻ variant due to its implementation of non-decreasing degree ordering, which effectively reduces computational overhead and enhances performance across most datasets and (p, q) configurations. Overall, the BC⁺ is the best-performing variant, as its optimized approaches results in consistently lower running times.

5 Conclusion

In this paper, we introduced **BC-Shadow**, the first approximation framework for (p, q)-biclique counting in bipartite graphs that provides formal accuracy guarantees. Our approach combines three key innovations: a novel sampling abstraction (*zstar*) with one-to-one correspondence to bicliques, an edge-oriented shadow refinement strategy that increases sampling efficiency, and a principled stopping rule based on theoretical bounds. Together, these components enable BC-Shadow to achieve accurate, unbiased estimation with provable guarantees, even in massive graphs. We conducted extensive experiments across diverse real-world datasets, demonstrating that BC-Shadow consistently outperforms state-of-the-art algorithms in both accuracy and runtime. Unlike prior work, BC-Shadow remains robust across varying graph densities and biclique configurations, while avoiding manual tuning of sample size or heuristics. Our work opens new directions for scalable motif estimation in large graphs.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. The work is supported by the Australian Research Council Fundings of DP220103731.

References

[1] Aman Abidi, Rui Zhou, Lu Chen, and Chengfei Liu. 2020. Pivot-based Maximal Biclique Enumeration. *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)* (2020), 3558–3564.

[2] Anna Arpaci-Dusseau, Zixiang Zhou, and Xuhao Chen. 2024. Accurate and Fast Approximate Graph Pattern Mining at Scale. arXiv:2405.03488 [cs.PF]

[3] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[4] Stephen P. Borgatti and Martin G. Everett. 1997. Network analysis of 2-mode data. *Social Networks* 19, 3 (1997), 243–269.

[5] Bole Chang, Linxin Xie, Wei Li, Meng Qin, and Jianfeng Hou. 2025. Z-Shadow: An Efficient Method for Estimating Bicliques in Massive Graphs Using Füredi's Theorem. In *Proceedings of the 28th International Conference on Extending Database Technology (EDBT)*. OpenProceedings, 755–768. doi:10.48786/edbt.2025.61

[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. doi:10.14778/3681954.3682032

[7] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2021. Efficient Exact Algorithms for Maximum Balanced Biclique Search in Bipartite Graphs. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 248–260. https://doi.org/10.1145/3448016.3459241

[8] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2022. Efficient maximal biclique enumeration for large sparse bipartite graphs. *Proc. VLDB Endow.* 15, 8 (April 2022), 1559–1571. doi:10.14778/3529337.3529341

[9] P. Dagum, R. Karp, M. Luby, and S. Ross. 1995. An optimal algorithm for Monte Carlo estimation. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*. 142–149. doi:10.1109/SFCS.1995.492471

[10] Maximilien Danisch, Oana Balalau, and Mauro Sozio. 2018. Listing k-cliques in Sparse Real-World Graphs. In *Proceedings of the 2018 World Wide Web Conference (WWW)*. 589–598.

[11] Pinghui Wang et al. 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.

[12] Zoltán Füredi. 1996. An Upper Bound on Zarankiewicz' Problem. *Comb. Probab. Comput.* 5 (1996), 29–33.

[13] Shweta Jain and C. Seshadhri. 2016. A Fast and Provable Method for Estimating Clique Counts Using Tur\'an's Theorem. *arXiv* (Nov. 2016).

[14] Shweta Jain and Comandur Seshadhri. 2020. The Power of Pivoting for Exact Clique Counting. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM)*. 268–276.

[15] Shweta Jain and Comandur Seshadhri. 2020. Provably and Efficiently Approximating Near-cliques using the Turán Shadow: PEANUTS. In *Proceedings of the 2020 World Wide Web Conference (WWW)*. 1966–1976.

[16] Madhav Jha, Comandur Seshadhri, and Ali Pinar. 2015. Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*. 495–505.

[17] 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.

[18] Rong-Hua Li, Sen Gao, Lu Qin, Guoren Wang, Weihua Yang, and Jeffrey Xu Yu. 2020. Ordering heuristics for k-clique listing. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2536–2548.

[19] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2020. Maximum biclique search at billion scale. *Proc. VLDB Endow.* 13, 9 (May 2020), 1359–1372. doi:10.14778/3397230.3397234

[20] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2022. Maximum and top-k diversified biclique search at scale. *The VLDB Journal* 31, 6 (2022), 1365–1389. doi:10.1007/s00778-021-00681-6

[21] Samuel Manoharan and Sathesh Ammaiappan. 2020. Patient Diet Recommendation System Using K Clique and Deep Learning Classifiers. In *Conference Proceedings*. 121–130.

[22] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos E. Tsourakakis, and Shen Chen Xu. 2015. Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 815–824.

[23] Zhe Pan, Xu Li, Shuibing He, Xuechen Zhang, Rui Wang, Yunjun Gao, Gang Chen, and Xian-He Sun. 2024. AMBEA: Aggressive Maximal Biclique Enumeration in Large Bipartite Graph Computing. *IEEE Trans. Comput.* 73, 12 (2024), 2664–2677.

[24] Ryan Rossi, Nesreen Ahmed, and Eunyee Koh. 2018. Higher-order Network Representation Learning. *Companion Proceedings of the WWW Conference* (2018), 3–4.

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

[26] Ahmet Erdem Sariyuce and Ali Pinar. 2018. Peeling Bipartite Networks for Dense Subgraph Discovery. In *WSDM*. 504–512.

[27] Phonexay Vilakone, Doo-Soon Park, Khamphaphone Xinchang, and Fei Hao. 2018. An Efficient Movie Recommendation Algorithm Based on Improved K-Clique. *Human-centric Computing and Information Sciences* 8, 1 (2018), 161.

[28] Kaixin Wang, Kaiqiang Yu, and Cheng Long. 2024. Efficient k-Clique Listing: An Edge-Oriented Branching Strategy. *Proc. ACM Manag. Data* 2, 1 (2024), Article 7.

[29] Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Shunyang Li. 2022. Discovering Hierarchy of Bipartite Graphs with Cohesive Subgraphs. *ICDE* (2022), 2291–2305.

[30] Jianye Yang, Yun Peng, and Wenjie Zhang. 2021. (p, q)-Biclique Counting and Enumeration for Large Sparse Bipartite Graphs. *Proceedings of the VLDB Endowment* 15, 2 (2021), 141–153.
[31] Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. 2022. Scalable and Effective Bipartite Network Embedding. *Proceedings of the 2022 International Conference on Management of Data (SIGMOD)* (2022), 1977–1991.
[32] Kai Yao, Lijun Chang, and Jeffrey Xu Yu. 2022. Identifying similar-bicliques in bipartite graphs. *Proc. VLDB Endow.* 15, 11 (July 2022), 3085–3097.
[33] Jianxiong Ye, Zhaonian Zou, Dandan Liu, Bin Yang, and Xudong Liu. 2025. BCviz: A Linear-Space Index for Mining and Visualizing Cohesive Bipartite Subgraphs. *Proc. ACM Manag. Data* 3, 1, Article 16 (Feb. 2025), 27 pages. doi:10.1145/3709666
[34] 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 (May 2023), 1–26. doi:10.1145/3588932
[35] Xiaowei Ye, Rong-Hua Li, Qiangqiang Dai, 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.
[36] Hao Yin, Austin R. Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. *Physical Review E* 97, 5 (2018), 052306.
[37] Yiwei Zhao, Zi Chen, Chen Chen, Xiaoyang Wang, Xuemin Lin, and Wenjie Zhang. 2022. Finding the Maximum k-Balanced Biclique on Weighted Bipartite Graphs. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–14.

Received April 2025; revised July 2025; accepted August 2025