A Relational Model of Data for Large Shared Data Banks

Ben Trovato
Institute for Clarity in Documentation
Dublin, Ireland
trovato@corporation.com

ABSTRACT

write abstract

PVLDB Reference Format:
Ben Trovato. A Relational Model of Data for Large Shared Data Banks. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

Contributions. Our key contributions are as follows:

- We present the first approximation algorithm with formal accuracy guarantees for the (p, q)-biclique counting problem.
- We propose an innovative edge-oriented technique for refining the sampling space, strategically balancing the computational load between the two stages of the algorithm to optimize overall runtime.
- We introduce a novel sampling method that not only ensures accuracy but also allows for dynamic refinement of the sampling process, adapting to the graph's structure for improved efficiency.

These contributions significantly advance the field of biclique counting, offering a practical and efficient approach for large-scale bipartite graph analysis.

1 INTRODUCTION

1.1 Exact Biclique Counting

1.2 Approximation Algorithms

2 PRELIMINARIES

In this paper, we focus on a large unweighted and undirected bipartite graph $G = (U, V, E)$, where U and V are sets of vertices, and E is the set of undirected edges. For each vertex $u \in U$ (or V), its neighbors are denoted by $N(u) = \{v \mid e(u, v) \in E\}$. The common neighbors of a vertex set S are denoted as $N(S) = \bigcap_{u \in S} N(u)$. Let $d(u)$ be the degree of a vertex u, i.e., $d(u) = |N(u)|$.

For simplicity, we assume without loss of generality that the vertices on each side of the bipartite graph G' are arranged in a particular order. Specifically, we order the vertices as $u_1 \prec u_2 \prec \cdots \prec u_{n_1}$ for U and $v_1 \prec v_2 \prec \cdots \prec v_{n_2}$ for V. Additionally, we assume that $p \leq q$, with p representing the number of vertices from the U side and q from the V side. The same methods can be applied when $p > q$.

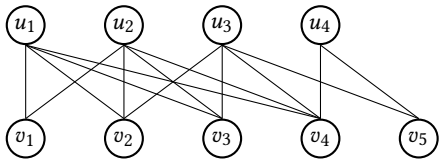

Figure 1: An example graph

Please note that, for convenience, the examples provided in this paper consider the natural ordering of the vertices.

Definition 2.1 (Biclique). A biclique $g = (U', V', E')$ in the bipartite graph $G = (U, V, E)$ is a pair of vertex subsets $U' \subseteq U$ and $V' \subseteq V$ such that every vertex $u \in U'$ is adjacent to every vertex $v \in V'$. i.e., $E' = \{(u, v) \mid u \in U', v \in V'\} \subseteq E$.

A biclique B is called a (p, q)-biclique if $|U'| = p$ and $|V'| = q$.

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

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

change () to {}

Consider the graph shown in Figure 1, there are five (2,3)-bicliques $\mathcal{B}_{(2,3)} = \{\{(u_1, u_2), (v_1, v_2, v_3)\}, \{(u_1, u_2), (v_1, v_2, v_4)\}, \{(u_1, u_2), (v_1, v_3, v_4)\}, \{(u_1, u_2), (v_2, v_3, v_4)\}\}, \{(u_2, u_3), (v_2, v_3, v_4)\}\}$.

Table 1: Frequently used notations

Notation	Meaning
$\mathcal{B}_{p,q}$	Set of all (p, q)-bicliques in G
$cnt_{p,q}(G)$	Total number of (p, q)-bicliques in G, i.e., $\lvert\mathcal{B}_{(p,q)}\rvert$
$\widehat{cnt}_{p,q}(G)$	Estimated total number of (p, q)-bicliques in G
$\Delta(G)$	Set of all (p, q)-zstars in G
$\mathcal{S}_{p,q}(G)$	sampling space of $\mathcal{B}_{p,q}$ satisfying $\mathcal{S}_{p,q}(G) \supseteq \mathcal{B}_{p,q}$
$\mathbb{S}_{p,q}(G)$	BC-Shadow

3 OUR APPROACH

add intro here

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Algorithm 1: BCFramework($G, p, q, \epsilon, \delta$)

Input: A bipartite graph $G = (U, V, E)$, integers p, q and, accuracy parameters ϵ, δ

Output: an estimation $\widehat{cnt}_{p,q}(G)$ for $cnt_{p,q}(G)$

1 Arrange U and V in some order

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

2 $\mathbb{S} \leftarrow \{(\emptyset, \emptyset, U, V)\}$ **while** *the construction stopping condition is not satisfied* **do**

3 Choose a sample subspace (R_U, R_V, S_U, S_V) and remove it from $\mathbb{S}$

// Refine the subspace (R_U, R_V, S_U, S_V) by partitioning it

4 **for** *each* $e(u, v) \in S_U \cup S_V$ **do**

5 Add $(R_U \cup u, R_V \cup v, N_{>u}(v) \cap S_U, N_{>v}(u) \cap S_V)$

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

6 $|\mathcal{S}_{p,q}(G)| = \sum_{(R_U,R_V,S_U,S_V)\in\mathbb{S}} |\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U, S_V)|$

$s \leftarrow 0, t \leftarrow 0$

7 **while** *the sampling stopping condition is not satisfied* **do**

8 Sample a subspace (R_U, R_V, S_U, S_V) from $\mathcal{S}$ with probability $\frac{|\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U,S_V)|}{|\mathcal{S}_{(p,q)}(G)|}$

9 Sample an element P u.a.r. from $\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U, S_V)$

10 **if** *P forms a (p, q)-biclique in G* **then**

11 $s \leftarrow s + 1$

12 $t \leftarrow t + 1$

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

Our framework for estimating the (p, q)-biclique count is presented in Algorithm 1 which operates in two stages.

In Stage I (Lines 1–6), we construct a sample space $\mathcal{S}_{(p,q)}(G)$ that meets the following conditions:

- $\mathcal{S}_{p,q}(G)$ is a superset of the set of all (p, q)-bicliques in G, denoted as $\mathcal{B}_{p,q}$, i.e.$\mathcal{B}_{p,q} \subseteq \mathcal{S}_{p,q}(G)$.
- Total count of sample elements in $\mathcal{S}_{p,q}(G)$ can be computed efficiently.
- We can efficiently sample elements uniformly at random (u.a.r.) from $\mathcal{S}_{(p,q)}(G)$.

Each element of the sample space $\mathcal{S}_{(p,q)}(G)$ is a pair of vertex sets (U_p, V_q), where $U_p \subseteq U$ and $V_q \subseteq V$, such that $|U_p| = p$ and $|V_q| = q$ with some additional constraints. In Stage II (Lines 7–12), 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. Specifically, if t samples are drawn from $\mathcal{S}_{p,q}(G)$, and for each sample i, $X_i = 1$ if the sample forms a (p, q)-biclique in G, and $X_i = 0$ otherwise, then the estimate of $cnt_{p,q}(G)$ is given by:

$$cnt_{p,q}(G) = |\mathcal{S}_{p,q}(G)| \cdot \frac{1}{t} \sum_{i=1}^{t} X_i \tag{1}$$

The core idea involves defining the (p, q)-biclique density of the sample space $\mathcal{S}_{p,q}(G)$ as:

$$\mu_{\mathcal{S}p,q} = \frac{cnt_{p,q}(G)}{|\mathcal{S}_{p,q}(G)|} \tag{2}$$

This value lies between 0 and 1. Since $cnt_{p,q}(G)$ is fixed, the (p, q)-biclique density depends only on the chosen sample space $\mathcal{S}_{p,q}(G)$. Different sample spaces will yield different values of $\mu_{\mathcal{S}_{p,q}(G)}$. Therefore, $cnt_{p,q}(G)$ can be expressed as:

$$cnt_{p,q}(G) = |\mathcal{S}_{p,q}(G)| \cdot \mu_{\mathcal{S}_{(p,q)}(G)} \tag{3}$$

The task then becomes estimating $\mu_{\mathcal{S}_{p,q}(G)}$, under the assumption that $|\mathcal{S}_{p,q}(G)|$ can be efficiently determined. Algorithm 1 estimates $\mu_{\mathcal{S}_{(p,q)}(G)}$ using the empirical mean $\hat{\mu}$ from t random samples, calculated as:

$$\hat{\mu} = \frac{1}{t} \sum_{i=1}^{t} X_i \tag{4}$$

Thus, the estimate for $cnt_{p,q}(G)$ becomes:

$$\widehat{cnt}_{p,q}(G) = |\mathcal{S}_{(p,q)}(G)| \cdot \hat{\mu} \tag{5}$$

This provides an unbiased estimate for $cnt_{p,q}(G)$.

3.1 BC-Shadow

The core idea of our algorithm, is to sample two vertex sets with from U and V with cardinalities $|p|$ and $|q|$ respectively, and then verify whether these sets form a biclique. A straightforward approach to sample a (p, q)-biclique would be to select all possible subsets of p vertices from U and q vertices from V, which leads to an unfeasible large sample space for large graphs, i.e., $\binom{|U|}{p} \cdot \binom{|V|}{q}$. Such an approach is computationally impractical due to the excessive memory requirements.To mitigate this, we need to reduce the sample space by eliminating vertex sets that do not form a biclique. In this section, we introduce a novel edge oriented sample space refinement technique, termed *BC-Shadow*, which is inspired by the notion of the shadow that was originally introduced for k-clique counting [2], was formally defined in [**?**].

Definition 3.1 (BC-Shadow). Let $G = (U, V, E)$ be a bipartite graph, and let p and q be two integers. The **BC-Shadow** $\mathbb{S}$ is a quadruple (R_U, R_V, S_U, S_V) representing a sampling space, with the following properties:

- $R_U \subseteq U$ and $R_V \subseteq V$, such that $R_U \cup R_V$ forms a biclique in G,
- $S_U \subseteq U \setminus R_U$ and $S_V \subseteq V \setminus R_V$, where $S_U = N(R_V)$, and $S_V = N(R_U)$
- For every (p, q)-biclique $B_U \cup B_V$ in G, there exists a unique subspace $(R_U, R_V, S_U, S_V) \in \mathbb{S}_{p,q}(G)$ such that: $R_U \subseteq B_U$, $R_V \subseteq B_V$, $B_U \setminus R_U \subseteq S_U$, and$B_U \setminus R_V \subseteq S_V$.

Note that, for counting the number of (p, q)-bicliques in G, it is enough to store the cardinalities $|R_U|$ and $|R_V|$ within $\mathbb{S}_{p,q}(G)$, rather than explicitly storing the sets R_U and R_V. However, if the goal is to sample and report the bicliques, then it is necessary to store the sets R_U and R_V explicitly.

From the definition of the BC-Shadow, the count of (p, q)-bicliques in G can be computed as:

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

In agorithm **??**, $\mathbb{S}_{p,q}(G)$ is initialized as $\{(\emptyset, \emptyset, U, V)\}$ and continously refined until stopping condition is met (Lines1–6). Which means that while R_U and R_V expand S_U and S_V shrink which result in increasing the bicilique denisity in the sampling space.

give an example

Lemma 3.2. *Lines 1–6 of Algorithm* **??** *correctly construct a valid BC-Shadow according to Definition 3.1.*

Proof. We prove by induction that after each iteration, the set $\mathbb{S}$ satisfies the conditions of Definition 3.1. Initially, $\mathbb{S} = \{(\emptyset, \emptyset, U, V)\}$, which satisfies the conditions: since $R_U = \emptyset$ and $R_V = \emptyset$, $R_U \times R_V$ trivially satisfies the biclique condition (no edges exist to violate it); every vertex in $S_U = U$ is adjacent to all vertices in $R_V = \emptyset$ (vacuously true), and similarly for S_V and R_U; and every (p, q)-biclique in G is contained within $R_U \cup S_U = U$ and $R_V \cup S_V = V$. For the inductive step, assume $\mathbb{S}$ satisfies Definition 3.1 before refinement. During refinement, for each edge $(u, v) \in E$ with $u \in S_U$ and $v \in S_V$, we create a new subspace (R'_U, R'_V, S'_U, S'_V) where $R'_U = R_U \cup \{u\}$ and $R'_V = R_V \cup \{v\}$; since u is adjacent to v, and by the adjacency properties of S_U and S_V, $R'_U \times R'_V$ forms a biclique. The updated shadow sets S'_U and S'_V consist of vertices adjacent to all of R'_V and R'_U, satisfying the second condition. The uniqueness condition holds because any (p, q)-biclique extending from (R_U, R_V) will be captured in exactly one of the new subspaces based on the ordering of vertices, ensuring no biclique is missed or duplicated. Therefore, after each iteration, $\mathbb{S}$ remains a valid BC-Shadow, and Lines 1–6 correctly construct it. □

3.2 A Novel Sampling Structure

After completing the BC-Shadow construction, the sampling space is significantly reduced. To count the bicliques, we can sample a subspace (R_U, R_V, S_U, S_V) and then sample an element P from it to check P forms a biclique. A simple approach might be to choose two vertex sets from U and V with sizes $p - |R_U|$ and $q - |R_V|$ respectively, but this naive method can still result in a large sampling space, i.e.,$\binom{|S_U|}{p-|R_U|} \cdot \binom{|S_V|}{q-|R_V|}$. To further refine the process and make sampling more efficient, we need to impose additional constraints on P. In this section, we introduce a novel sampling structure, called zstar [3], designed to efficiently sample elements from subspaces $(S_U, S_V, R_U, R_V) \in \mathbb{S}_{p,q}(G)$

For the convenience and improve the presentation of the general idea we introduce the concepts considering G as the processing graph rather than the subspace (S_U, S_V, R_U, R_V). Note that when applying for the proposed techniques to the subspace (S_U, S_V, R_U, R_V) we need to consider the subgraph $G'(U', V', E')$ induced by R_U, R_V, p as p' where $p' = p - |R_U|$ and q as q' where $q' = q - |R_V|$.

For convenience, we assume without loss of generality that the vertices on each side of the bipartite graph G' are arranged in some order. Specifically, we have $u_1 \prec u_2 \prec \cdots \prec u_{n_1}$ and $v_1 \prec v_2 \prec \cdots \prec v_{n_2}$.We also assume that the $p \le q$ because the same methods can be used when $p < q$. please note that for convenience, given examples in this paper does not consider the natural is this correct? ordering.

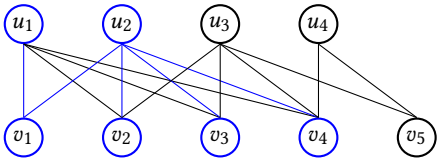

Figure 2: An example for $Z_{(2,4)}$

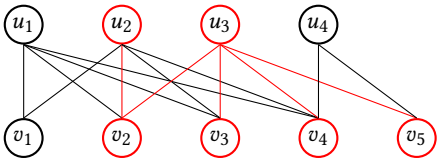

Figure 3: Another for $Z_{(2,4)}$

Definition 3.3. Given a bipartite graph $G(U, V, E)$ and integers p, q, h , a h-zstar in G,is defined as an ordered vertex set

$$Z = \{u_{i_1}, v_{j_1}, u_{i_2}, v_{j_2}, \ldots, u_{i_h}, v_{j_h}, \ldots, v_{j_{h+(q-p)}}\}$$

that satisfies the following conditions:

(1) The number of vertices from the U side is h, and the number of vertices from the V side is $h+(q-p)$. i.e., $|Z| = 2h+q-p$
(2) The vertices are ordered such that $i_1 < i_2 < \cdots < i_h$ and $j_1 < j_2 < \cdots < j_{h+(q-p)}$.

Additionally, the length of a (p, q)-zstar is defined as $2h - 1$.

edit examples Figures 2 and 3 shows two 2-zstars in the example graph G.Note that the 2-zstar in figure 2 forms a (2,3)-bicique while figure 3 does not form a (2,3)-biclique. In 2-zstar shown in the figure 2, The primary purpose of designing zstar is to establish a correspondence between a biclique and a zstar such that every biclique has **exactly one** corresponding zstar. Note that every zstar may not have a corresponding biclique.

let $\Delta(G)$ be the set of h-zstars in G. We define a mapping $f : \mathcal{B}_{p,q} \to \Delta$ such that $f(b) = z$, where $B \in \mathcal{B}_{p,q}$ is a biclique and $Z \in \Delta$ is a h-zstar . This mapping f ensures that:

$$\forall B \in \mathcal{B}_{p,q}, \exists Z \in \Delta \text{ such that } f(B) = Z,$$

but

$$\exists Z \in \Delta \text{ such that } \nexists B \in \mathcal{B}_{p,q} \text{ with } f(B) = Z.$$

Lemma 3.4. *given a (p, q)-biclique in G with $p \le q$, it contains exactly one h-zstar if $p = h$*

Proof. The size of (p,q)-biclique is $p + q$ and by definition the size of h-zstar is $2h + q - p$ .Hence a (p, q)-biclique has exactly one h-zstar . □

let $\mathcal{T}$ is the set of uniform h-zstars sampled form Δ .

Lemma 3.5. *let Z be a h-zstar uniformly sampled from Δ. The probability of Z forms a (p,q)-biclique in G is $1/|\Delta|$*

Proof. As given in 3.4 a (p,q)-biclique contains exactly one h-zstar ,hence the lemma holds. □

Let $X(Z)$ be a binary random variable indicating whether a given h-zstar Z forms a biclique, such that $X(Z) \in \{0,1\}$. For any Z, we define:

$$X(Z) = \begin{cases} 1 & \text{if } Z \text{ forms a biclique} \\ 0 & \text{otherwise} \end{cases}$$

Based on Lemma 3.5, we can construct an unbiased estimator for the count of bicliques by uniformly sampling Z from Δ.

Theorem 3.6. *Consider $\mathcal{T}$, a set of h-zstars sampled uniformly from Δ. The unbiased estimator for the count of (p,q)-bicliques in graph G, denoted as $\widehat{cnt}_{(p,q)}$, is given by:*

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

Proof. We proceed as follows:

(1) By definition, we can establish that $\sum_{Z \in \mathcal{T}} X(Z) = \widehat{cnt}_{(p,q)}$
(2) This leads us to $\mathbb{E}[X(Z)] = \frac{\widehat{cnt}_{(p,q)}}{|\Delta|}$.
(3) Utilizing these results, we can demonstrate:

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

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

Counting and Sampling zstars: Reminding that we assume that $p \le q$ and $h = p$, To maximize the utility of zstars, it is essential to efficiently count the number of h-zstars in a graph G. We propose an efficient dynamic programming approach for counting h-zstars , inspired by [3] . As defined in 3.3, If every zstar originates from a node u , then the total count of zstars can be determined by summing the zstars that start from each u in the set U.Let $dp[i][(u,v)]$ represent the number of zstars with length i starting from (u,v), where $i \in [2, 2h-1]$. Then the number of zstars with length $2h-1$ starting from u can be represented as $\sum_{v \in N(u)} dp[2h-1][e(u,v)]$. Then the total zstars in G is $\sum_{e \in E} dp[2h-1][e(u,v)]$.

We noticed that $dp[i][e(u,v)]$ can also be computed using a dynamic programming algorithm. The core concept of our DP approach is that each zstar with the length of i beginning from u is built upon the zstar of length $i-1$ starting from node v when i > 2.Therefore, we have below recursion equation:

$$\begin{cases} dp[i][e(u,v)] \leftarrow \sum_{u' \in N_{>u}(v)} dp[i-1][e(v,u')], \\ dp[i-1][e(v,u')] \leftarrow \sum_{v' \in N_{>v}(u')} dp[i-2][e(u',v')], \\ dp[2][e(v,u)] \leftarrow \binom{|N_{>u}(v)|}{q-p} \end{cases} \tag{6}$$

Algorithm 2: zstarDP(G,p,q)

Input: A bipartite graph $G(U,V,E)$ and integers p and q
Output: Dynamic programming table dp

```
for each e(v,u) ∈ E do
    Let N_{>v}(u) ← the set of neighbors of u with vertex
      rank higher than the rank of v
    dp[2][e(v,u)] ← C(|N_{>v}(u)|, q−p+1)
for i = 3 to 2h − 1 do
    for each (v,u) ∈ E do
        for each (u′,v) where u′ < u do
            dp[i+1][(u′,v)] ←
              dp[i][(u′,v)] + dp[i−1][e(v,u)]
    if i ≠ 2h − 1 then
        for each e(u′,v) ∈ E do
            for each e(v′,u′) where v′ < v do
                dp[i+1][e(v′,u′)] ←
                  dp[i+1][e(v′,u′)] + dp[i][e(u′,v)]
    i ← i + 2
return dp
;
```

Note that notations $dp[i][e(u,v)]$ and $dp[i][e(v,u)]$ in equation (6) are different to each other. Algorithm 2 outlines the steps for counting h-zstars. The algorithm takes as input the graph G and two integers, p and q, and produces a dynamic programming table that records the zstar counts for each edge in G, with zstar lengths ranging from 2 to $2h-1$.

The initial time complexity of Algorithm 2 is $O(h \cdot d_{max} \cdot |E|)$, where d_{max} is the maximum degree of the input graph G and $h = \min(p,q)$. This is inefficient for large graphs since d_{max} could be equal to number of vertices in either side, making the algorithm impractical for large graphs. To address this, we apply a differential-interval updating technique that reduces the time complexity to $O(h \cdot |E|)$.

Consider the dynamic programming table at level i, denoted as $dp[i]$, represented as a sequence $\{a_1, a_2, \ldots, a_n\}$, where $a_j = dp[i][e_j]$ corresponds to an edge e_j. We transform this into a difference sequence $\{b_1, b_2, \ldots, b_n\}$, where $b_1 = a_1$ and $b_j = a_j - a_{j-1}$ for $j = 2$ to n. Each a_j is reconstructed by computing the prefix sum: $a_j = \sum_{k=1}^{j} b_k$.

In the algorithm, updates occur when i is odd (for edges $e(v,u')$ where $u' > u$) and when i is even (for edges $e(u,v')$ where $v' > v$). These updates increment a_j over specific intervals. By using the difference sequence $\{b_1, b_2, \ldots, b_n\}$, we can efficiently apply these interval updates by modifying b_l += w and b_{r+1} −= w for the range $[l,r]$, where w is the value we need to add. After applying all updates, the original $dp[i]$ is reconstructed by computing prefix sums.

This technique reduces the per-update time from $O(k)$, where k is the number of edges in the interval, to $O(1)$, leading to a total time complexity of $O(h \cdot |E|)$.

sampling a zstar: Since we have efficient method to count the zstars in G ,now we need to efficiently sample a zstar uniformly at random from $\Delta(G)$, ensuring that each zstar has an equal probability $\frac{1}{|\Delta(G)|}$ of being selected. To achieve this, we utilize the DP table $dp[i][e(u,v)]$, which records the number of partial zstars of length i ending at edge $e(u,v)$.

We present an efficient method to uniformly sample a h-zstar from the bipartite graph G, leveraging the dynamic programming counts obtained earlier. The central challenge in uniformly generating a h-zstar is to establish 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 in G is given by:

$$|\Delta(G)| = \sum_{e(u,v)\in E} dp[2h-1][e(u,v)], \tag{7}$$

where $dp[2h-1][e(u,v)]$ denotes the number of zstars of length $2h-1$ 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 a zstar Z as:

$$P(e(u,v)) = \frac{dp[2h-1][e(u,v)]}{\sum_{e(u,v)\in E} dp[2h-1][e(u,v)]} \tag{8}$$

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

$$P(e(v,u') \mid e(u,v)) = \frac{dp[2h-2][e(v,u')]}{dp[2h-1][e(u,v)]}. \tag{9}$$

This recursive method continues for each subsequent edge in the zstar, decrementing i from $2h-1$ down to 2. By this stage, we have sampled p vertices from the U side and $p-1$ vertices from the V side. To complete the h-zstar, we need to select the remaining $q-p+1$ vertices from V.

Given the last edge $e(u,v)$ selected in the recursive procedure, we proceed by choosing $q-p+1$ additional vertices from the set $N_{>v}(u)$. To maintain uniformity, each vertex v' in $N_{>v}(u)$ is selected with equal probability:

$$P(v') = \frac{1}{|N_{>v}(u)|}. \tag{10}$$

By uniformly sampling these vertices, we ensure that all possible combinations are equally likely, thus preserving the uniformity of the overall sampling process. We summarise these steps in algorithm 3.

Algorithm 3: Sampling a Z-Star

Input: A bipartite graph $G = (U, V, E)$, integers p, q
Output: A uniformly sampled zstar Z

Set the distribution $\mathcal{D}$ over edges where $p(e(u,v)) = \frac{dp[2h-1][e(u,v)]}{\sum_{e(u,v)\in G} dp[2h-1][e(u,v)]}$;
Sample an edge $e(u_0, v_0)$ according to $\mathcal{D}$
Initialize $Z \leftarrow \{u_0, v_0\}$, $k \leftarrow 0$
// Recursive construction of zstar
for $i = 2h-2$ ***down to*** 2 ***step*** -1 **do**
  if *i is even* **then**
    // Sampling edge from V to U
    $E' \leftarrow \{e(v_k, u') \mid u' \in N_{>u_k}(v_k)\}$
    Set the distribution $\mathcal{D}$ over E' where:
    $$P(e(v_k, u')) = \frac{dp[i][e(v_k, u')]}{dp[i+1][e(u_k, v_k)]}$$
    Sample an edge $e(v_k, u_{k+1})$ according to $\mathcal{D}$
    Append u_{k+1} to Z
  else
    // Sampling edge from U to V
    $E' \leftarrow \{e(u_{k+1}, v') \mid v' \in N_{>v_k}(u_{k+1})\}$
    Set the distribution $\mathcal{D}$ over E' where:
    $$P(e(u_{k+1}, v')) = \frac{dp[i][e(u_{k+1}, v')]}{dp[i+1][e(v_k, u_k)]}$$
    Sample an edge $e(u_{k+1}, v_{k+1})$ according to $\mathcal{D}$
    Append v_{k+1} to Z
    $k \leftarrow k+1$
// Add remaining vertices from V
$V_{\text{rem}} \leftarrow \{v' \in N_{>v_k}(u_{k+1})\}$
Uniformly select $q-p+1$ vertices from V_{rem} without replacement;
Append the selected vertices to Z
return Z

3.3 Sampling Stopping Condition

writting as a generalised version, should it be written specifically for the sampling structure.

In this subsection, we focus on the accuracy of our estimation, which is determined by **Stage II** of our algorithm.

The existing algorithm [3] requires the user to specify the number of samples t to be drawn uniformly at random from $\mathcal{S}_{p,q}(G)$. As this value is set manually,It cannot provide any formal guarantees on the accuracy of the estimates. To address this issue, we propose a different strategy for determining when to stop the sampling process. Specifically, we fix the required numbers of successful samples s (i.e., samples that correspond to actual(p,q)-bicliques), and continue sampling from $\mathcal{S}_{p,q}(G)$ until s successful samples have been collected.We then estimate the biclique count $cnt_{p,q}(G)$ using $\widehat{cnt}_{p,q}(G) = |\mathcal{S}_{p,q}(G)| \cdot \frac{s}{t}$, where t is the total number of samples drawn.

Based on the stopping rule theorem [1], the estimation accuracy is guaranteed when the number of successful samples s satisfies $s \geq \gamma$, where

$$\gamma = 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2},$$

where e is Euler's number. We will use the parameter γ for the quantity $1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$ throughout the remainder of the paper.

Algorithm 4: BicliqueEstimation($\mathcal{S}_{p,q}(G), p, q, \epsilon, \delta$)

Input: $\mathcal{S}_{p,q}(G)$, integers p, q, error parameters $\epsilon \in (0,1)$ and $\delta \in (0,1)$
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 A u.a.r. from $\mathcal{S}_{p,q}(G)$;
  if *A forms a (p,q)-biclique in G* **then**
    $s \leftarrow s + 1$;
  $t \leftarrow t + 1$;
return $\widehat{cnt}_{p,q}(G) \leftarrow |\mathcal{S}_{p,q}(G)| \cdot \frac{s}{t}$;

Theorem 3.7. *Let $\widehat{cnt}_{p,q}(G)$ be the estimate returned by Algorithm 4. Then, $P\left(|\widehat{cnt}_{p,q}(G) - cnt_{p,q}(G)| > \epsilon \cdot cnt_{p,q}(G)\right) \le \delta$.and the expected number of samples $E[t]$ satisfies $\frac{\gamma \cdot |\mathcal{S}_{p,q}(G)|}{cnt_{p,q}(G)} \le E[t] < \frac{(\gamma+1)\cdot|\mathcal{S}_{p,q}(G)|}{cnt_{p,q}(G)}$.*

Proof. Since each sample from $\mathcal{S}_{p,q}(G)$ is drawn uniformly at random, and $\frac{s}{t}$ is the proportion of successful samples, $\frac{s}{t}$ is an unbiased estimator of the biclique density $\mu_{\mathcal{S}_{p,q}(G)} = \frac{cnt_{p,q}(G)}{|\mathcal{S}_{p,q}(G)|}$.

By applying the stopping rule theorem (e.g., see [1]), we have that the estimator $\widehat{cnt}_{p,q}(G) = |\mathcal{S}_{p,q}(G)| \cdot \frac{s}{t}$ satisfies the desired accuracy guarantee when $s \ge \gamma$.

The bounds on $\mathbb{E}[t]$ follow from standard properties of negative binomial sampling, where the expected number of trials to achieve s successes in Bernoulli trials with success probability $\mu_{\mathcal{S}_{p,q}(G)}$ is $E[t] = \frac{s}{\mu_{\mathcal{S}_{p,q}(G)}}$. Substituting $\mu_{\mathcal{S}_{p,q}(G)} = \frac{cnt_{p,q}(G)}{|\mathcal{S}_{p,q}(G)|}$ and $s \ge \gamma$ yields the desired bounds.

□

The accuracy guarantee remains valid regardless of the specific sample space $\mathcal{S}_{p,q}(G)$, provided it includes all possible (p,q)-bicliques in G. However, different constructions of $\mathcal{S}_{p,q}(G)$ will influence the biclique density $\mu_{\mathcal{S}_{p,q}(G)}$, which in turn impacts the expected running time of the algorithm. In the following subsection, we will 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** (sample space construction) and **Stage II** (sampling and estimation).

3.4 Balancing the Running Time of the Two Stages

If we spend minimal time refining the sample space in Stage I, the sample space $S_{p,q}(G)$ remains large and contains many elements that do not form bicliques. This results in a low biclique density $\mu_{S_{p,q}(G)}$, causing Stage II to require a large number of samples to achieve the desired estimation accuracy. Conversely, if we overinvest time in refining $S_{p,q}(G)$, Stage I becomes time-consuming without proportionate benefits, as the gains in increasing $\mu_{S_{p,q}(G)}$ diminish with each refinement.

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.

3.4.1 Estimating the Expected Running Time of Stage II. The expected running time of Stage II depends on:

- **The number of samples required**: Determined by the desired estimation accuracy and the biclique density of the sample space.

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

Input: Bipartite Graph $G(U,V,E)$ integers p, q and error parameters ϵ, δ
Output: Shadow $\mathbb{S}$

1 Arrange U and V in some order
2 $\widetilde{cnt}_{p,q}(G) \leftarrow 1$; $|\mathcal{S}_{p,q}(G)| \leftarrow |\mathcal{P}_{(p,q)}(G)|$; $\tilde{T}_{sample} \leftarrow \infty$
3 $\mathbb{S} \leftarrow \{(\emptyset, \emptyset, U, V, \widetilde{cnt}_{p,q}(G)/|\mathcal{S}_{p,q}(G)|)\}$
4 **while** *ElapsedTime()* $< \gamma \cdot \frac{|\mathcal{S}_{p,q}(G)|}{\widetilde{cnt}_{(p,q)}(G)} \cdot \tilde{T}_{sample}$ **do**
5 $(R_U, R_V, S_U, S_V, \ddot{\mu}) \leftarrow \arg\min_{(R'_U, R'_V, S'_U, S'_V, \ddot{\mu}') \in \mathbb{S}} \ddot{\mu}'$;
6 Remove $(R_U, R_V, S_U, S_V, \ddot{\mu})$ from $\mathbb{S}$;
  $\widetilde{cnt}_{(p,q)}(G) \leftarrow \widetilde{cnt}_{(p,q)}(G) - |\mathcal{P}_{(p,q)}(S_U, S_V)| \cdot \ddot{\mu}$;
7 $|\mathcal{S}_{p,q}(G))| \leftarrow |\mathcal{S}_{p,q}(G)| - \mathcal{P}_{(p,q)}(S_U, S_V)$;
8 **if** $R_U = \emptyset$ **then**
9 $n_{sample} \leftarrow 0$; $T_{total} \leftarrow 0$
10 **for** ***each*** $u \in S_U$ **do**
11 **for** ***each*** $v \in N_u(S_V)$ **do**
12 $(R'_U, R'_V, S'_U, S'_V, \ddot{\mu}) \leftarrow$ $(R_U \cup u, R_V \cup v, N_v(S_U), N_u(S_V))$;
13 $\ddot{\mu}' \leftarrow$ auxiliary $(p - |R'_U|, q - |R'_V|)$-biclique density in (S'_U, S'_V) from $\hat{n}$ samples;
14 Add $(R'_U, R'_V, S'_U, S'_V, \ddot{\mu}')$ to $\mathbb{S}$;
15 $\widetilde{cnt}_{(p,q)}(G) \leftarrow$ $\widetilde{cnt}_{(p,q)}(G) + |\mathcal{P}_{(p-|R'_U|, q-|R'_V|)}(S'_U, S'_V)| \cdot \ddot{\mu}'$;
16 $|\mathcal{S}_{p,q}(G)| \leftarrow$ $|\mathcal{S}_{p,q}(G)| + |\mathcal{P}_{(p-|R'_U|, q-|R'_V|)}(S'_U, S'_V)|$;
17 $S_V \leftarrow S_V \backslash v$;
18 **if** $S_U = \emptyset$ **then**
19 $n_{sample} \leftarrow n_{sample} + \hat{n}$
20 $T_{total} \leftarrow T_{total}$ + running time of Line 13
21 $S_U \leftarrow S_U \backslash u$;
22 **return** $\mathbb{S}$

- **The average time per sample** T_{sample}: Time taken to draw a sample from $S_{p,q}(G)$ and verify whether it forms a biclique.

Following from the stopping rule theorem, the expected number of samples t taken when the sampling algorithm terminates is approximately:

$$t \approx \frac{\gamma}{\mu_{S_{p,q}(G)}}$$

where $\gamma = 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$, ϵ and δ are the relative error and confidence parameters, and e is Euler's number.

Therefore, the expected running time of Stage II is:

$$\text{Expected Running Time of Stage II} \approx \frac{\gamma}{\mu_{S_{p,q}(G)}} \times T_{\text{sample}} \quad (8)$$

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

3.4.2 Effect of Refinement on Biclique Density. We observe that each refinement of a sample subspace of the BC-Shadow $S_{p,q}(G)$ reduces the size of the corresponding sample space and, consequently, increases the biclique density $\mu_{S_{p,q}(G)}$. This is formalized in the following lemma.

Lemma 3.8. *Assuming that the same ordering of vertices is used in defining $P_{p-|R_U|,q-|R_V|}(S_U, S_V)$ for different subspaces of $S_{p,q}(G)$, and that vertices are processed in the same order during refinement, each refinement of a sample subspace of $S_{p,q}(G)$ results in a smaller sample space and an increased biclique density.*

Proof. Suppose a subspace (R_U, R_V, S_U, S_V) is refined into $\{(R_{U_i}, R_{V_i}, S_{U_i}, S_{V_i})\}_{i=1}^{l}$ by the refinement process. We need to show that:

$$\bigcup_{i=1}^{l} S_{p,q}(R_{U_i}, R_{V_i}, S_{U_i}, S_{V_i}) \subseteq S_{p,q}(R_U, R_V, S_U, S_V)$$

This means that the refined sample spaces are subsets of the original sample space. Additionally, since the sample spaces do not overlap due to the ordering, we have:

$$S_{p,q}(R_{U_i}, R_{V_i}, S_{U_i}, S_{V_i}) \cap S_{p,q}(R_{U_j}, R_{V_j}, S_{U_j}, S_{V_j}) = \emptyset \quad \text{for } i \neq j$$

Thus, each refinement reduces the size of the sample space, and since the number of bicliques remains the same or decreases less rapidly, the biclique density $\mu_{S_{p,q}(G)}$ increases. □

3.4.3 Balancing Strategy. If we stop Stage I immediately after initializing $S_{p,q}(G)$ as $\{(\emptyset, \emptyset, U, V)\}$, Stage I will be very efficient, but Stage II will require a long time due to the low biclique density. Conversely, extensive refinement in Stage I will increase $\mu_{S_{p,q}(G)}$ but also significantly increase the time spent in Stage I.

To determine when to stop refining and proceed to Stage II, we need to estimate the expected running time of Stage II without actually executing it. We can compute γ since ϵ and δ are input parameters, and T_{sample} can be estimated by sampling a few elements from $S_{p,q}(G)$ and measuring the average time. However, estimating $\mu_{S_{p,q}(G)}$ is challenging since it is the very quantity we aim to estimate.To address this challenge, we propose computing an auxiliary estimation $\tilde{\mu}$ for $\mu_{S_{p,q}(G)}$. Unlike the final estimate, $\tilde{\mu}$ does not require strict theoretical accuracy guarantees, as its sole purpose is to influence the running time, rather than the accuracy of the final biclique count $cnt_{p,q}(G)$. 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 $\tilde{\mu}$ accurately and efficiently.

3.4.4 Construction Stopping Condition. Our construction stopping condition is:

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

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

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

- Sampling a small number of elements uniformly at random from $P_{p',q'}(S_U, S_V)$, 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 $\text{cnt}_{p',q'}(S_U, S_V) \approx |P_{p',q'}(S_U, S_V)| \times \mu'$.

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

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

$$|S_{p,q}(G)| = \sum_{(R_U,R_V,S_U,S_V) \in S} |P_{p',q'}(S_U, S_V)|$$

The auxiliary estimation of the biclique density is then:

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

3.4.6 The Pseudocode of Shadow Construction. Based on the above discussions, the pseudocode of our shadow construction algorithm for biclique counting is presented in Algorithm 5. The algorithm takes a bipartite graph $G(U, V, E)$, integers p, q, and error parameters ϵ, δ as input, and outputs a shadow $\mathbb{S}$.

The initialization phase begins by arranging the vertices in U and V in some order to facilitate efficient sampling and refinement (Line 1). The initial count $\widetilde{cnt}_{p,q}(G)$ is set to 1, and the size of the sample space $|S_{p,q}(G)|$ is initialized to $|\mathcal{P}_{p,q}(G)|$ (Lines 2–3). The value of $\tilde{T}_{\text{sample}}$ is initially set to infinity, and the shadow $\mathbb{S}$ is initialized with the full vertex sets U and V, along with the initial estimation of μ (Line 4).

In the refinement loop (Lines 5–24), the algorithm continues refining the shadow as long as the elapsed time is less than the estimated running time of Stage II (Line 5). At each iteration, the subspace $(R_U, R_V, S_U, S_V, \mu)$ with the smallest μ is selected for refinement (Line 6). The algorithm updates $\widetilde{cnt}_{p,q}(G)$ and the size of the sample space $|S_{p,q}(G)|$ by removing the contribution of the selected subspace (Lines 7–8). For each $u \in S_U$ and $v \in N_u(S_V)$, new subspaces are created by extending R_U and R_V, and auxiliary information is computed to estimate the biclique density μ' in the new subspaces using a small sample size (Lines 11–13). These new subspaces are added back to the shadow $\mathbb{S}$, and the values of $\widetilde{cnt}_{p,q}(G)$ and $|S_{p,q}(G)|$ are updated accordingly (Lines 14–15). The processed vertices are removed from S_U and S_V (Lines 16–17), and after processing the initial subspaces, the algorithm estimates $\tilde{T}_{\text{sample}}$ based on the total sampling time and number of samples (Lines 21–22). This approach ensures efficient refinement of the subspaces while maintaining computational feasibility during the sampling process.

3.5 Estimating Bicliques

Algorithm 6: BicliqueEstimation2($\mathbb{S}, p, q, \epsilon, \delta$)

Input: Shadow $\mathbb{S}$ integers p, q, error parameters ϵ, δ
Output: $\widetilde{cnt}_{(p,q)}(G)$

1 $s \leftarrow 0$; $t \leftarrow 0$
2 $\gamma = 1 + \frac{4(1+\epsilon)(e-2)\ln(2/\delta)}{\epsilon^2}$
3 **while** $s < \gamma$ **do**
4 Sample a subspace $(R_U, R_V, S_U, S_V, \ddot{\mu}) \in \mathbb{S}$ with probability $\frac{|\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U, S_V)|}{|\mathcal{S}_{p,q}(G)|}$
5 Sample an element P uniformly at random from $\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U, S_V)$
6 **if** *P forms a biclique in G* **then**
7 $s \leftarrow s + 1$
8 $t \leftarrow t + 1$
9 $\widehat{cnt}_{p,q}(G) \leftarrow |\mathcal{S}_{p,q}(G)| \cdot \frac{s}{t}$
10 **return** $\widehat{cnt}_{p,q}(G)$

The final algorithm for estimating the number of (p, q)-bicliques is shown in Algorithm 6.The algorithm proceeds by repeatedly sampling elements from the overall sample space until the number of successful samples s reaches the threshold γ (Lines 3 8). In each iteration, a subspace $(R_U, R_V, S_U, S_V, \ddot{\mu})$ is selected from the shadow $\mathbb{S}$ with probability proportional to the size of its corresponding sampling space $\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U, S_V)$ relative to the total sample space size $|\mathcal{S}_{p,q}(G)|$ (Line 4). This weighted sampling ensures that every potential (p, q)-biclique in the total sample space has an equal probability of being selected, maintaining uniformity across the entire sample space.

Once a subspace is selected, an element P is sampled uniformly at random from $\mathcal{P}_{(p-|R_U|,q-|R_V|)}(S_U, S_V)$ (Line 5). This element represents a $p - |R_U|$-zstar.We then check whether the sampled element P forms a biclique in G (Line 6). If it does, we increment the successful sample counter s (Line 7). Regardless of whether P forms a biclique, we increment the total sample counter t (Line 8). This process continues until the number of successful samples s reaches the threshold γ (Line 3), ensuring that sufficient evidence has been gathered to achieve the desired estimation accuracy.After obtaining the required number of successful samples, we compute the estimate $\widehat{cnt}_{p,q}(G)$ of the total number of (p, q)-bicliques (Line 9). This estimate is calculated by scaling the ratio of successful samples to total samples by the size of the overall sample space $|\mathcal{S}_{p,q}(G)|$.Finally, the algorithm returns this estimate $\widehat{cnt}_{p,q}(G)$ (Line 10), which, due to the properties of the sampling method and the stopping condition, satisfies the accuracy and confidence guarantees specified by the error parameters ϵ and δ.

ACKNOWLEDGMENTS

This work was supported by the [...] Research Fund of [...] (Number [...]). Additional funding was provided by [...] and [...]. We also thank [...] for contributing [...].

REFERENCES

[1] 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. https://doi.org/10.1109/SFCS.1995.492471
[2] Shweta Jain and C. Seshadhri. 2016. A Fast and Provable Method for Estimating Clique Counts Using Tur\'an's Theorem. *arXiv* (Nov. 2016). https://doi.org/10.48550/arXiv.1611.05561 arXiv:1611.05561
[3] 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. https://doi.org/10.1145/3588932