Efficient and Effective Biclique Counting with Local Differential Privacy

YIZHANG HE, University of New South Wales, Australia
WENJIE ZHANG, University of New South Wales, Australia
KAI WANG, Data-Driven Management Decision Making Lab, Antai College of Economics and Management, Shanghai Jiao Tong University, China
XUEMIN LIN, Antai College of Economics and Management, Shanghai Jiao Tong University, China
YING ZHANG, University of Technology Sydney, Australia
WEI NI, University of New South Wales, Australia

Counting (p, q)-bicliques in bipartite graphs is essential for uncovering dense substructures and higher-order patterns. However, releasing biclique counts can reveal private connections of users. To address this, we study (p, q)-biclique counting under edge local differential privacy, where each vertex protects its one-hop neighbor relationships from an untrusted data curator. The `Naive` algorithm applies randomized responses to the adjacency matrix and produces an overly dense graph where (p, q)-biclique structures are disrupted, resulting in highly biased estimates. To produce unbiased estimates, we propose a one-round algorithm that enumerates all motifs with p upper and q lower vertices, which leverages motif transformation probabilities to correct for perturbation-induced bias. To improve data utility and reduce the impact of Laplace noise, we propose a novel Multi-round Common Neighbor (`MRCN`) algorithm. `MRCN` estimates the number of common neighbors for each p-tuple (or q-tuple) and applies moment-based correction to recover unbiased (p, q)-biclique counts. Notably, `MRCN` avoids explicit enumeration of all (p, q)-motifs, offering better scalability for general p and q. We further boost performance with variance-reduction techniques such as multi-center optimization and refined noisy graph construction, and enhance scalability through layer-based pruning and vertex sampling. Extensive experiments on real-world datasets confirm the effectiveness and efficiency of our approaches.

CCS Concepts: • **Networks** → **Network privacy and anonymity**; • **Information systems** → **Network data models**.

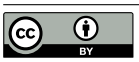
Additional Key Words and Phrases: Motif Counting, Differential Privacy

ACM Reference Format:
Yizhang He, Wenjie Zhang, Kai Wang, Xuemin Lin, Ying Zhang, and Wei Ni. 2026. Efficient and Effective Biclique Counting with Local Differential Privacy. *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 28 (February 2026), 24 pages. https://doi.org/10.1145/3786642

1 Introduction

Bipartite graphs are a natural fit for modeling relationships between two distinct sets of entities, such as user-item interactions in e-commerce [23], people-location networks in contact tracing [1], and user-page networks in social media analysis [27]. (p, q)-biclique counting has been widely

Authors' Contact Information: Yizhang He, University of New South Wales, Australia, yizhang.he@unsw.edu.au; Wenjie Zhang, University of New South Wales, Australia, wenjie.zhang@unsw.edu.au; Kai Wang, Data-Driven Management Decision Making Lab, Antai College of Economics and Management, Shanghai Jiao Tong University, China, w.kai@sjtu.edu.cn; Xuemin Lin, Antai College of Economics and Management, Shanghai Jiao Tong University, China, xuemin.lin@sjtu.edu.cn; Ying Zhang, University of Technology Sydney, Australia, ying.zhang@uts.edu.au; Wei Ni, University of New South Wales, Australia, wei.ni@ieee.org.



ACM 2836-6573/2026/2-ART28
https://doi.org/10.1145/3786642

studied [30, 45, 47] since it captures higher-order cohesive structures with p upper and q lower vertices. This enables more fine-grained pattern detection in applications such as fraud detection [40, 41, 50], dense subgraph mining [8, 9, 35, 37, 38], and link prediction [15]. However, releasing exact (p, q)-biclique counts can compromise privacy by revealing sensitive relationships, such as a user's interaction with a specific item. For example, consider the user-item networks in Fig. 1,

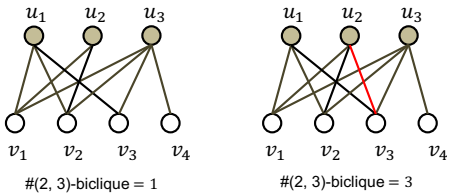


Fig. 1. An example of how $(2, 3)$-biclique counts reveal private connections.

where each edge indicates a user's purchase of an item. In the bipartite graph on the left, there are 3 users and 4 items, and one $(2, 3)$-biclique formed by users (u_1, u_3) and items (v_1, v_2, v_3). When user u_2 additionally purchases item v_3, edge (u_2, v_3) is added, resulting in the graph on the right. Suppose users u_1 and u_3 collude to infer u_2's purchases. They know all edges incident to themselves and have background knowledge of the existing purchase history of u_2, as commonly assumed in differential privacy studies [16]. If the $(2, 3)$-biclique counts of the graphs are released, the colluding users may observe a change (e.g., from 1 to 3) and infer that u_2 purchased v_3, resulting in a potential privacy breach. Thus, it is vital to estimate the number of (p, q)-bicliques in a privacy-preserving manner.

Motivated by these observations, we study the problem of privacy-preserving (p, q)-biclique counting on bipartite graphs. Specifically, given a bipartite graph G, we aim to compute the (p, q)-biclique count under *edge local differential privacy* (edge LDP) [28, 46, 49]. Edge LDP ensures robust privacy by requiring each vertex to perturb its local data before transmission to the data curator, ensuring that the presence or absence of any single edge cannot be reliably inferred from the output. This eliminates the need for a trusted curator as required in *central DP* [2, 4, 10, 11, 20, 21, 26].

Existing methods. Motif counting under edge LDP typically follows two paradigms: (1) one-round algorithms, which construct a noisy graph and estimate motifs directly from it, and (2) multi-round algorithms, which allow vertices to download noisy graphs from earlier rounds and estimate motifs locally [12, 14, 16, 18]. While He et al. [12] propose a one-round algorithm for butterfly counting ($p = q = 2$), no one-round methods exist for general (p, q)-bicliques. Classic multi-round approaches for (p, q)-biclique counting suffer from either violating edge LDP or injecting excessive noise due to high sensitivity of per-vertex motif counts [33].

Challenges. In this paper, we design accurate and efficient algorithms for counting (p, q)-bicliques under edge LDP and face the following challenges.

• *Challenge 1:* Given a bipartite graph G, randomized responses produce a much denser noisy graph G' by independently flipping edges [28], which often destroys complex structures like (p, q)-bicliques. As a result, a `Naive` algorithm that directly counts (p, q)-bicliques on G' suffers from significant bias. It is a challenge to derive unbiased estimates for (p, q)-bicliques under edge LDP.

• *Challenge 2:* Due to edge flipping in randomized responses, recovering accurate (p, q)-biclique counts requires handling the transformation of all motifs with p upper and q lower vertices. This computational task becomes prohibitively expensive for large values of p and q.

Our Approaches. To address *Challenge 1*, we extend the one-round algorithm for butterfly counting under edge LDP [12] and propose a `OneR` algorithm that provides unbiased (p, q)-biclique estimates

based on the noisy graph. Our proposed OneR algorithm analyzes how motifs transform under randomized response and corrects for the resulting bias. Specifically, OneR constructs a noisy graph and counts motifs with different numbers of edges, then uses these counts to derive unbiased estimates by accounting for expected motif transformations. While OneR works well on dense graphs, it requires enumerating all (p, q)-vertex motifs, which becomes computationally challenging for large values of p and q.

To address *Challenge 2*, we propose a multi-round algorithm, MRCN, for (p, q)-biclique counting that avoids exhaustive subgraph enumeration. Unlike classic multi-round methods that inject large Laplace noise into per-vertex biclique counts, MRCN estimates common neighbors for each p-tuple (or q-tuple) and derives unbiased biclique estimates from their higher moments. It first estimates vertex degrees, then perturbs edges via randomized response, and finally selects a central vertex per p-tuple to estimate noisy common neighbors with budget ε_2. By correcting higher moments of these estimators, MRCN obtains unbiased per-tuple biclique counts, aggregated to the total (p, q)-biclique count. This design leverages local neighborhood while avoiding the intractable global sensitivity of bicliques and reducing computation from enumerating all motifs to processing p- or q-tuples.

To enhance the data utility of MRCN, we develop MRCN++, which integrates two variance-reduction techniques: multi-center optimization and refined noisy graph construction. The multi-center strategy selects multiple upper-layer vertices to form independent common-neighbor estimators and averages them to suppress vertex-level noise. The refined noisy graph construction exploits the parallel and post-processing properties of edge LDP to create two independent noisy graphs under the same privacy budget, reducing adjacency variance and improving biclique counting accuracy.

Contributions. Here, we outline our main contributions.

• We study the (p, q)-biclique counting problem under edge LDP. We propose the OneR algorithm that corrects for motif transformations caused by randomized response. OneR produces unbiased (p, q)-biclique estimation and works well on dense bipartite graphs.

• We propose a novel multi-round common-neighbor-based algorithm, MRCN, which estimates the number of common neighbors for each p-tuple in the upper layer (or q-tuple in the lower layer). By correcting higher-order moments of these estimators, MRCN yields more accurate and unbiased estimates of the (p, q)-biclique count.

• We propose the MRCN++ algorithm with two variance-reduction techniques: multi-center optimization and refined noisy graph construction.

• We conduct extensive experiments on 13 real-world datasets to evaluate the effectiveness and efficiency of our algorithms. Results show that OneR, MRCN, and MRCN++ consistently outperform the Naive algorithm in data utility. Notably, MRCN achieves mean relative errors up to three orders of magnitude lower than Naive. Also, MRCN++ with the proposed optimizations consistently outperforms MRCN in most settings.

2 Problem Definition

In this section, we first present important definitions and notations about (p, q)-biclique and differential privacy on bipartite graphs. Then, we present the problem statement.

2.1 (p, q)-biclique Counting on Bipartite Graphs

We consider an unweighted bipartite graph $G(V = (U, L), E)$. $V = U \cup L$ denotes the set of vertices, where U and L represent the upper and lower layers, respectively. The vertices in U and L are called upper vertices and lower vertices, respectively. $E \subseteq U \times L$ denotes the set of edges. Let $n_1 = |U(G)|$, $n_2 = |L(G)|$, and $m = |E(G)|$ denote the number of upper vertices, lower vertices, and edges, respectively. The adjacency matrix $\mathcal{A}$ for G is of dimensions $n \times n$, where $\mathcal{A}[u, v] = 1$ if an edge exists between vertices u and v, and 0 otherwise. As G is bipartite, only the upper-right and

Table 1. Summary of Notations

Notation	Definition
G	a bipartite graph
$\mathcal{A}$	the adjacency matrix
$\mathcal{A}_u$	the neighbor list of the vertex u
$N(u, G)$	the neighbors of u in G
$deg(u, G)$	the degree of u in G
ε	a privacy budget
G'	a noisy bipartite graph
$\omega(X)$	the number of common neighbors among X
Δg	the global sensitivity of a function g
$\mathcal{K}_{p,q}$	number of (p, q)-bicliques in graph G
$\mathcal{K}_{p,q}(X)$	(p, q)-biclique count containing the vertex set X

lower-left blocks of $\mathcal{A}$ contain nonzero entries. The u-th row of $\mathcal{A}$ (including both "1" and "0") is the *neighbor list* of u, denoted by $\mathcal{A}_u$. In addition, the set of neighbors of a vertex u in G is denoted by $N(u, G)$, and its degree is denoted by $deg(u, G) = |N(u, G)|$. We denote by $d^1_{\max}$ and $d^2_{\max}$ the maximum degrees of vertices in the upper set $U(G)$ and lower set $L(G)$, respectively. In this paper, we consider the bicliques with p upper vertices and q lower vertices, as defined below.

Definition 1. **(p, q)-biclique.** *Given a bipartite graph G and two integers p and q, a (p, q)-biclique is a complete subgraph of G with p upper vertices and q lower vertices.*

2.2 Differential privacy on bipartite graphs

Classic DP assumes a central model, where a central data curator manages a dataset D [6]. Two datasets D and D' are neighboring datasets if they differ by one data record. For neighboring datasets, DP ensures that the outputs of a query function on these datasets are hard to distinguish. When DP under the central model is applied to graphs to protect edge privacy, the neighboring datasets are two graphs that differ by one edge. While central DP offers better data utility, the assumption that the data curator storing the entire graph can be trusted is often unattainable. Thus, we adopt ε-edge LDP, which allows each vertex to perturb its data locally before sending it to the data curator [29, 46, 49]. Under edge LDP, the neighboring datasets are two neighbor lists that differ by one bit.

Definition 2. **ε-edge local differential privacy.** *Let $G(V = (U, L), E)$ be a bipartite graph and $\varepsilon > 0$. For each vertex $u \in V(G)$, let R_u with domain $\{0, 1\}^n$ be a randomized algorithm of vertex u. R_u provides ε-edge LDP if for any two neighbor lists $\mathcal{A}_u$ and $\mathcal{A}'_u$ that differ in one bit and any $S \subseteq Range(R_u)$, where $Range(R_u)$ denotes the range (output space) of the randomized algorithm R_u:*

$$Pr(R_u(\mathcal{A}_u) \in S) \leq e^{\varepsilon} Pr(R_u(\mathcal{A}'_u) \in S)$$

Here, ε is referred to as the privacy budget.

Note that ε-edge LDP ensures that if two vertices have neighbor lists that differ by one bit, they cannot be reliably distinguished based on the outputs from the randomized algorithms. Under the post-processing property of edge LDP, any function applied to a differentially private output maintains the same privacy guarantee. This means that once a noisy graph is constructed via randomized response satisfying edge LDP, it can be safely shared with all vertices without additional privacy loss, as the noisy graph itself is a differentially private output. This follows directly from Theorem 2, which guarantees that any function of a differentially private output maintains the

same privacy guarantee. Consider t edge LDP algorithms $Z_1, Z_2, \ldots, Z_t$, whose privacy budgets are denoted by $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_t$, respectively. We have the following property.

Theorem 1. **Sequential Composition [20].** *The composite algorithm obtained by sequentially applying $Z_1, Z_2, \ldots, Z_t$ on the bipartite graph G provides $\sum_{i=1}^{t} \varepsilon_i$-differential privacy.*

Theorem 2. **Post-Processing Property [6].** *Let M be a mechanism that provides ε-differential privacy. For any function f, the composition $f \circ M$ also provides ε-differential privacy. That is, any function applied to the output of a differentially private mechanism maintains the same privacy guarantee without consuming additional privacy budget.*

Theorem 3. **Parallel Composition [6].** *Let $M_1, M_2, \ldots, M_k$ be mechanisms that provide $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k$-differential privacy, respectively, and operate on disjoint subsets of the dataset. The composition of these mechanisms provides $\max(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k)$-differential privacy.*

Randomized responses. The easiest way to provide ε-edge LDP is via randomized responses (hereafter denoted by RR), which was originally proposed as a survey technique to allow confidential answers to sensitive issues (e.g. criminal or sexual activities) [42]. Essentially, the participants are asked to answer questions truthfully with probability. This has been applied to graphs to satisfy edge local differential privacy [16, 28]. Specifically, given a privacy budget ε, each entry $x \in \{0, 1\}$ of a neighbor list is perturbed independently with probability $\frac{1}{1+e^{\varepsilon}}$:

$$RR(x) = \begin{cases} 1 - x & \text{with probability } \frac{1}{1+e^{\varepsilon}} \\ x & \text{with probability } \frac{e^{\varepsilon}}{1+e^{\varepsilon}} \end{cases}$$

Existing works on bipartite graphs [12, 13] restrict randomized response to the upper-right block. This avoids conflicts in the noisy graph where both $\mathcal{A}'[i, j]$ (perturbed by i) and $\mathcal{A}'[j, i]$ (perturbed by j) might be inconsistent. We denote the noisy graph w.r.t. a privacy budget ε as G'_{ε}.

Calibrating noise to global sensitivity. To ensure ε-edge LDP, any data transmitted from a vertex to the data curator requires noise addition. *The Laplace Mechanism* [6] is often used to draw noise proportional to the global sensitivity of the data. We formally define global sensitivity here.

Definition 3. **Global sensitivity.** *Consider a bipartite graph G. Let $\mathcal{A}_u$ be the neighbor list of vertex u. The global sensitivity of a function $f : \mathcal{A}_u \to \mathbb{R}$ is:*

$$\Delta f = max_{u,u' \in V(G)} |f(\mathcal{A}_u) - f(\mathcal{A}'_u)|$$

Here, $\mathcal{A}'_u$ is a neighbor list that differs from $\mathcal{A}_u$ in at most one entry. Δf represents the global sensitivity of f.

Definition 4. **The Laplace Mechanism.** *Given a privacy budget ε and any function f, the Laplace mechanism is defined as:*

$$\tilde{f} = f + Lap\left(\frac{\Delta f}{\varepsilon}\right)$$

where $\tilde{f}$ is the noisy version of f and $Lap(\cdot)$ is the Laplace probability density function.

Applying the Laplacian mechanism, we allow the vertices to send local graph statistics with calibrated noise to the data curator while satisfying ε-edge LDP.

Problem Statement. Given a bipartite graph G, two integers p and q, a privacy budget ε, we aim to accurately and efficiently estimate the number of (p, q)-bicliques in G under ε-edge LDP.

A naive approach. Since the randomized responses preserve ε-edge LDP, a naive solution returns the (p, q)-biclique count of the noisy graph from the randomized responses, as detailed in Algorithm

Algorithm 1: `Naive`

Input: G: a bipartite graph; p, q: integers; ε: a privacy budget;
Output: an estimate of the number of (p, q)-bicliques
`// vertex side:`
1 **foreach** $i \in U(G)$ **do**
2 **foreach** $j \in L(G)$ **do**
3 perturb $\mathcal{A}'[i, j] \leftarrow \begin{cases} 1 - \mathcal{A}[i, j], & \text{w.p. } \frac{1}{1+e^{\varepsilon}} \\ \mathcal{A}[i, j] & \text{w.p. } \frac{e^{\varepsilon}}{1+e^{\varepsilon}} \end{cases}$
4 send noisy edges to the data curator;
`// curator side:`
5 $G'_{\varepsilon} \leftarrow$ the noisy graph constructed from $\mathcal{A}'[i, j]$;
6 **return** the (p, q)-biclique count on G'_{ε};

1. Here, we use $\mathcal{A}'$ to indicate the adjacency matrix for the noisy graph G'. In Lines 1-4, each upper vertex perturbs its neighbor list and sends it to the data curator. Specifically, given a privacy budget $\varepsilon > 0$, each entry $A[i, j]$ with i and j being vertices on different layers is flipped with a probability $\frac{1}{1+e^{\varepsilon}}$. In Lines 5-6, the data curator constructs a noisy graph G'_{ε} and returns the number of (p, q)-bicliques on G'_{ε} obtained by calling the state-of-the-art biclique counting algorithm [39]. Since most neighbor lists are sparse (with more "0" than "1"), flipping them with a fixed probability $\frac{1}{1+e^{\varepsilon}}$ makes G'_{ε} much denser than G, which generally leads to over-counting.

Computational time complexity. We analyze the computational time cost and communication costs of `Naive`. During the randomized response stage, the time cost is $O\left(n_1 n_2 \cdot \frac{1}{1+e^{\varepsilon}}\right)$ due to the edge flipping probability of $\frac{1}{1+e^{\varepsilon}}$. In the counting stage, the dominant cost arises from (p, q)-biclique counting on the noisy graph G'_{ε}, which requires $O\left(\min(n_1^p, n_2^q)\right)$ time in the worst-case scenario. This is because the state-of-the-art (p, q)-biclique counting algorithm [45, 47] runs in $O\left(\min(n_1^p, n_2^q)\right)$ time. Thus, the overall time costs of `Naive` are $O\left(\frac{n_1 n_2}{1+e^{\varepsilon}} + (\min(n_1^p, n_2^q))\right)$.

Per-vertex Communication costs. The communication costs of `Naive` occur during randomized responses, when each vertex uploads the noisy edges to the data curator. Each noisy edge incurs a cost of $O(\log(n_1 n_2))$, since each edge can be represented by two vertex IDs [17]. The expected number of noisy edges sent by a vertex u is $deg(u)(1 - p_r) + (n_2 - deg(u))p_r = O(\frac{n_2}{1+e^{\varepsilon}})$. Thus, the per-vertex communication cost is $O\left(\frac{n_2}{1+e^{\varepsilon}} \cdot \log(n_1 n_2)\right) = O\left(\frac{n_2 \log(n_1 n_2)}{1+e^{\varepsilon}}\right)$.

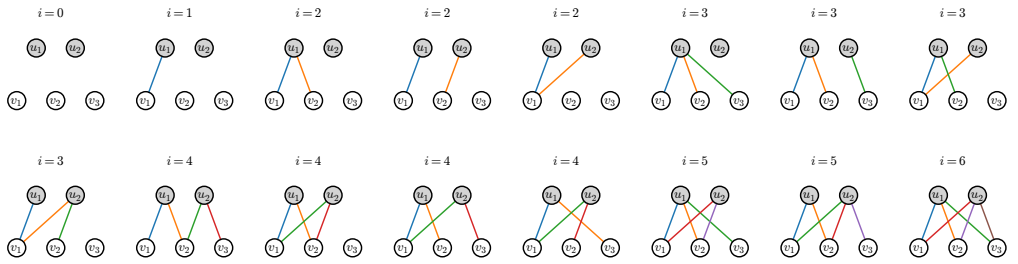


Fig. 2. All possible motifs with 2 upper and 3 lower vertices. Each motif is annotated with its edge count i.

3 One round approach

The Naive algorithm in Section 2 leads to significant bias because the noisy graph is much denser than the original graph. Recently, [12] proposes a multi-round algorithm and a non-interactive algorithm for butterfly counting under edge LDP, where the non-interactive approach achieves superior data utility as it prevents injecting substantial noise from the high global sensitivity in the classic multiple-round computing framework. The non-interactive algorithm relies on randomized responses to build a noisy graph and carefully model the transformation probabilities of all motifs with two upper and lower vertices to get unbiased butterfly count estimates. Since the butterfly is a special case of (p, q)-biclique, we extend this idea to (p, q)-biclique and propose a one-round algorithm that produces an unbiased estimate of (p, q)-biclique counts. To begin with, we revisit how motifs transform into each other with different probabilities due to randomized responses.

3.1 Motif transformation during randomized responses

To recover the (p, q)-biclique count on G, we need to investigate how the motifs with p upper vertices and q lower vertices transform into each other with different probabilities. Here, we define the transformation probability.

Definition 5. **Transformation probability [12].** *Given a privacy budget ε and a bipartite graph G, let G'_ε be a noisy graph from applying randomized response to G. Let $\mathcal{M}$ be a motif in G induced by a vertex set $V \subseteq V(G)$, and let $\mathcal{N}$ be a motif over the same vertex set V in G'_ε. The* transformation probability *from $\mathcal{M}$ to $\mathcal{N}$ is defined as the probability that $TP(\mathcal{M}, \mathcal{N}) = P(G'_\varepsilon[V] \cong \mathcal{N} \mid G[V] \cong \mathcal{M})$, where $\cong$ denotes graph isomorphism.*

As shown in Fig. 2, a $(2, 3)$-biclique can transform into 16 distinct types of motifs with different probabilities. For instance, the transformation probability of a (2, 3)-biclique to itself is $(1-p)^6$ as it requires each of the 6 edges to stay un-flipped during randomized responses ($p = 1/(1+e^\varepsilon)$). In what follows, we consider the motifs with the same number of edges together and use $\mathcal{B}_i$ to denote the motifs with i edges ($0 \le i \le p \times q$). The numbers of $\mathcal{B}_i$ on G and G' are denoted by $n_{\mathcal{B}_i}$ and $n'_{\mathcal{B}_i}$, respectively. Then, we have the following equation. $\mathbb{E}(n'_{\mathcal{B}_1}, n'_{\mathcal{B}_2}, \cdots, n'_{\mathcal{B}_{p\times q}}) = (n_{\mathcal{B}_1}, n_{\mathcal{B}_2}, \cdots, n_{\mathcal{B}_{p\times q}}) \times \mathcal{T}$ Here $\mathcal{T}$ is the transition matrix where $\mathcal{T}[i, j] = TP(\mathcal{B}_i, \mathcal{B}_j)$. In this way, the transition matrix $\mathcal{T}$ stores the transformation probabilities from $\mathcal{B}_i$ to $\mathcal{B}_j$, $\forall i, j$. Based on the above equation, we obtain the following linear system for (p, q)-biclique counting.

$$(\hat{n}_{\mathcal{B}_0}, \hat{n}_{\mathcal{B}_1}, \cdots, \hat{n}_{\mathcal{B}_{pq}}) = (n'_{\mathcal{B}_0}, n'_{\mathcal{B}_1}, \cdots, n'_{\mathcal{B}_{pq}}) \times \mathcal{T}^{-1} \tag{1}$$

Since $C_{p,q}(G) = n_{\mathcal{B}_{pq}}$, we solve for $\hat{n}_{\mathcal{B}_{pq}}$ as the unbiased estimate for $C_{p,q}(G)$ in the theorem below.

Theorem 4. *Given a bipartite graph G, a privacy budget ε, let G'_ε be the noisy graph after applying randomized response to G w.r.t. ε. Let $\mu = e^\varepsilon$. We have $\mathbb{E}(\hat{n}_{\mathcal{B}_{pq}}) = C_{p,q}(G)$ where $\hat{n}_{\mathcal{B}_{pq}} = \frac{\sum_{i=0}^{p\times q}(-1)^i \mu^i n'_{\mathcal{B}_i}}{(1-\mu)^{p\times q}}$.*

3.2 A one-round algorithm

Based on Theorem 4, we propose a one-round algorithm that returns an unbiased estimate of the number of (p, q)-bicliques. The algorithm first constructs the noisy graph G'_ε using the full privacy budget ε. It then counts the number of motifs with p upper and q lower vertices in G'_ε, and applies the formula in Theorem 4 to compute the unbiased estimate.

In practice, no general efficient algorithm exists for counting motifs with p upper and q lower vertices for arbitrary (p, q) values. Specialized counting strategies must be developed for each case. In the following, we present the one-round algorithm for the specific case of $p = 2$ as an example.

Algorithm 2: One-Round $(2, q)$-Biclique Counting under Edge LDP

Input: G: a bipartite graph with vertex sets U and L; q: target size on L side; ε: privacy budget;
Output: an estimate of the number of $(2, q)$-bicliques

```
// vertex side: apply randomized response
```

1 **foreach** $i \in U$ **do**
2 **foreach** $j \in L$ **do**
3 perturb edge presence:
4 $\mathcal{A}'[i,j] \leftarrow \begin{cases} 1-\mathcal{A}[i,j], & \text{w.p. } \frac{1}{1+e^{\varepsilon}} \\ \mathcal{A}[i,j], & \text{w.p. } \frac{e^{\varepsilon}}{1+e^{\varepsilon}} \end{cases}$
5 send noisy edges $\mathcal{A}'$ to the curator;

```
// curator side: counting motifs with two upper and q lower vertices
```

6 Construct noisy graph G'_{ε} from $\mathcal{A}'$;
7 Initialize counters $m_0, m_1, \ldots, m_{2K} \leftarrow 0$;
8 **foreach** *pair* $(u_1, u_2) \in U(G')$ **do**
9 Compute s_0, s_1, s_2 based on neighborhoods in G'_{ε};
10 **for** $c = 0$ **to** q **do**
11 **for** $b = 0$ **to** $q - c$ **do**
12 $a \leftarrow q - b - c$;
13 $i \leftarrow 2c + b$;
14 $m_i \leftarrow m_i + \binom{s_0}{a} \cdot \binom{s_1}{b} \cdot \binom{s_2}{c}$;

```
// estimate correction using inverse transform
```

15 Let $\mu \leftarrow e^{\varepsilon}$;
16 $\hat{n}_{\mathcal{B}_{2q}} \leftarrow \frac{1}{(1-\mu)^{2q}} \cdot \sum_{i=0}^{2q} (-\mu)^i \cdot m_i$;
17 **return** $\hat{n}_{\mathcal{B}_{2q}}$;

An example of the one-round algorithm when $p = 2$. Algorithm 2 presents a specialized implementation of the one-round algorithm for the case where $p = 2$ and the lower side has size q. In the counting phase (Lines 8–14), Algorithm 2 iterates over each upper vertex pair (u_1, u_2) and classifies lower vertices into s_2, s_1, and s_0, denoting the counts of vertices connected to both, exactly one, or neither vertex in the noisy graph G'_{ε}. For each triple (a, b, c) with $a + b + c = q$, the algorithm counts the number of $(2, q)$ motifs where a lower vertices connect to neither u_1 nor u_2, b to exactly one, and c to both. The count for motifs with $i = 2c + b$ edges is incremented by $\binom{s_0}{a} \cdot \binom{s_1}{b} \cdot \binom{s_2}{c}$. In this way, Algorithm 2 efficiently counts the number of motifs with two upper vertices and q lower vertices in G'_{ε}. Finally, Lines 15–16 apply the closed-form formula from Theorem 4 to recover an unbiased estimate of the number of $(2, q)$-bicliques in the original graph G.

Computational time complexity. We analyze the computational time cost of `OneR`. The time costs incurred during randomized response is $O\left(\frac{n_1 n_2}{1+e^{\varepsilon}}\right)$. Unlike `Naive`, `OneR` needs to count all motifs with p upper vertices and q lower vertices on G'_{ε}, which takes $O\left(\binom{n_1}{p} \cdot \binom{n_2}{q} \cdot pq\right)$. $\binom{n_1}{p}$ and $\binom{n_2}{q}$ denote the number of ways to choose p and q vertices from U and L, and pq accounts for the time to examine the number of edges in each motif. Thus, the time complexity of `OneR` is $O\left(\frac{n_1 n_2}{1+e^{\varepsilon}} + \binom{n_1}{p} \cdot \binom{n_2}{q} \cdot pq\right)$.

When $p = 2$, Algorithm 2 iterates over all upper vertex pairs and computes motif counts via neighborhood intersections, which takes $O(n_1^2 \cdot q^2)$ time in the counting phase. Overall, the time complexity is $O\left(\frac{n_1 n_2}{1+e^{\varepsilon}} + n_1^2 q^2\right)$ when $p = 2$.

Note that for counting *all* motifs with p upper and q lower vertices, the naive approach that enumerates all vertex sets of size p in the upper layer and q in the lower layer is highly inefficient, taking $O(\binom{n_1}{p}\binom{n_2}{q})$ time. To the best of our knowledge, no faster algorithm exists for arbitrary (p, q)-motifs.

Per-vertex Communication costs. The communication costs of `OneR` occur during randomized responses, just like `Naive`, which incurs a per-vertex communication cost of $O\left(\frac{n_2 \log(n_1 n_2)}{1+e^{\varepsilon}}\right)$.

4 A novel multi-round framework

The one-round algorithm for (p, q)-biclique counting mitigates the dense-noise problem by modeling motif transformation probabilities under randomized response to produce unbiased estimates. However, `OneR` suffers from high variance, as each (p, q)-biclique depends on about $p \times q$ independent edge flips, and it requires a separate counting procedure for each (p, q), which becomes intractable when $p > 2$. We therefore turn to multi-round algorithms, widely used in triangle counting [14, 16, 17]. In this framework, part of the privacy budget is used to construct a noisy graph via randomized response, and each vertex estimates local (p, q)- and quasi-(p, q)-bicliques based on its neighborhood overlap. To report these local counts under edge LDP, substantial Laplace noise must be added due to the large global sensitivities of noisy biclique structures.

To address the high global sensitivity of noisy (p, q)-biclique counts, we propose **Multi-Round Common Neighbor-based algorithm** (`MRCN`) for unbiased (p, q)-biclique counting under edge LDP. (1) Estimate vertex degrees via the Laplace mechanism. (2) Construct a noisy graph using randomized response. (3) For each p-vertex set, estimate its common neighbors and their higher-order moments to derive an unbiased (p, q)-biclique estimator. We first present the case $p = 2$ and later generalize to larger p.

4.1 A working example for $p = 2$

We begin with the special case of $p = 2$. To count $(2, q)$-bicliques under edge LDP, we consider all vertex pairs in $U(G)$, estimate their number of common neighbors, and derive the $(2, q)$-biclique count for each pair. We build on a recent work in common neighbor estimation under edge LDP [13] and discuss how to obtain unbiased per-vertex-pair $(2, q)$-biclique count estimators.

An unbiased estimator for common neighbors per vertex pair. The common neighbor estimator builds on the unbiased estimator for each adjacency entry $\mathcal{A}[i, j]$ in a bipartite graph G. Given privacy budget ε_1, we construct a noisy graph G'_{ε_1} by applying randomized response to each edge, flipping its presence with probability $p_r = \frac{1}{1+e^{\varepsilon_1}}$. Each noisy entry $\mathcal{A}'[i, j]$ then follows a Bernoulli distribution with parameter $1 - p_r$ if $\mathcal{A}[i, j] = 1$, and p_r if $\mathcal{A}[i, j] = 0$. Based on this, we have the following equations which link the expected value of $\mathcal{A}[i, j]'$ and $\mathcal{A}[i, j]$:

$$\mathbb{E}(\mathcal{A}'[i, j]) = \begin{cases} p_r, & \text{if } \mathcal{A}[i, j] = 0 \\ 1 - p_r & \text{if } \mathcal{A}[i, j] = 1 \end{cases}$$

This can be rearranged into: $E(\phi(i, j)) = \mathcal{A}[i, j]$. Here, $\phi(i, j) = \frac{\mathcal{A}'[i,j]-p_r}{1-2p_r}$, which is an unbiased estimator of $\mathcal{A}[i, j]$. The variance of $\phi(i, j)$ is given by

$$\mathrm{Var}(\phi(i, j)) = \mathrm{Var}(\frac{\mathcal{A}'[i, j] - p_r}{1 - 2p_r}) = \frac{p_r(1 - p_r)}{(1 - 2p_r)^2}$$

Assume the noisy graph G_{ε_1} has been constructed and downloaded by each vertex. This sharing is permitted under edge LDP's post-processing property, as the noisy graph is a differentially private output that can be safely distributed to all vertices without compromising the original edge privacy guarantees. This follows from Theorem 2, which ensures that sharing the noisy graph maintains

the same privacy guarantee. For a vertex pair (u, w) with $u, w \in U(G)$, the number of common neighbors can be locally estimated by u using the unbiased estimator $\phi(v, w)$ for each $v \in N(u, G)$. Specifically, $\sum_{v \in N(u,G)} \phi(v, w)$ provides an unbiased estimate of the common neighbors between u and w. To report this to the data curator, Laplace noise must be added based on the global sensitivity of the estimator, which is $\frac{1-p_r}{1-2p_r}$. Thus, the unbiased estimator at the curator becomes

$$f_u = \sum_{v \in N(u,G)} \phi(v, w) + \text{Lap}\left(\frac{1-p_r}{(1-2p_r)\varepsilon_2}\right).$$

The variance of f_u arises from two sources: the sum $\sum_{v \in N(u,G)} \phi(v, w)$ and the Laplace noise. The variance contributed by the first term is $\frac{p_r(1-p_r)}{(1-2p_r)^2} \deg(u)$, since each $\phi(v, w)$ is independent and there is no covariance. The variance of the Laplace noise is

$$\text{Var}\big(\text{Lap}\left(\frac{1-p_r}{(1-2p_r)\varepsilon_2}\right)\big) = 2\left(\frac{1-p_r}{(1-2p_r)\varepsilon_2}\right)^2 = \frac{2(1-p_r)^2}{(1-2p_r)^2\varepsilon_2^2}.$$

Thus, the total variance of f_u is

$$\text{Var}(f_u) = \frac{p_r(1-p_r)}{(1-2p_r)^2} \deg(u) + \frac{2(1-p_r)^2}{(1-2p_r)^2\varepsilon_2^2}.$$

Estimating $\mathbb{K}_{2,q}(u, w)$ for the vertex pair $(u.w)$. Let $\omega(u, w)$ denote the true number of common neighbors between u and w, and let f_u be the unbiased estimate of $\omega(u, w)$ obtained from u's local neighborhood. For the case $p = 2$, the estimators f_u and f_w (computed from u's and w's perspectives, respectively) are statistically independent. This independence arises because f_u depends only on edges from u's neighbors to w (i.e., $\mathcal{A}'[v, w]$ for $v \in N(u, G)$), while f_w depends only on edges from w's neighbors to u (i.e., $\mathcal{A}'[u, v']$ for $v' \in N(w, G)$). Since these edge sets are disjoint, and randomized response flips each edge independently, the estimators f_u and f_w are independent. This independence simplifies the variance calculation and is crucial for moment-based correction.

To estimate the number of $(2, q)$-bicliques involving u and w, denoted by $\mathbb{K}_{2,q}(u, w)$, we use the combinatorial formula: $\mathbb{K}_{2,q}(u, w) = \binom{\omega(u,w)}{q}$. Using $\binom{f_u}{q}$ as a plug-in estimator introduces bias, as the expectation is not preserved under nonlinear transformations such as polynomials. To address this, we introduce a *moment correction technique* that: (1) expresses $\binom{\omega(u,w)}{q}$ as a polynomial in $\omega(u, w)$, (2) derives unbiased estimators for each monomial ω^k using the higher moments of f_u, and (3) combines them to construct an unbiased estimator for $\binom{\omega(u,w)}{q}$. We first illustrate the method for $q = 2$ and then extend it to handle general $q > 2$.

(1) When $p = 2$ and $q = 2$, we count the number of butterflies in G. For a vertex pair (u, w), the butterfly count is $\mathbb{K}_{2,2}(u, w) = \binom{\omega(u,w)}{2} = \frac{\omega(u,w)^2 - \omega(u,w)}{2}$. Given the estimator f_u for $\omega(u, w)$, we aim to estimate $\omega(u, w)^2$ by leveraging the variance: $\text{Var}(f_u) = \mathbb{E}(f_u^2) - \mathbb{E}(f_u)^2$. Thus, we have $\mathbb{E}(f_u^2 - \text{Var}(f_u)) = \omega(u, w)^2$. Based on this, we obtain the following unbiased estimator for $\mathbb{K}_{2,2}(u, w)$:

$$\hat{\mathbb{K}}_{2,2}(u, w) = \frac{f_u^2 - f_u - \text{Var}(f_u)}{2}.$$

Given the expression of $\text{Var}(f_u)$ obtained earlier, we can estimate it by replacing $\deg(u)$ with its unbiased estimate $\tilde{\deg}(u) = \deg(u, G) + \text{Lap}(1/\varepsilon_0)$ from the first round. Based on the above discussion, we can obtain the following unbiased estimator for the total number of butterflies:

$$\hat{\mathbb{K}}_{2,2} = \sum_{u < w \in U(G)} \frac{f_u^2 - f_u - \hat{\text{Var}}(f_u)}{2}$$

(2) When $p = 2$ and $q > 2$, we apply the moment-based correction technique to estimate $\mathbb{K}_{2,q}(u, w)$. Specifically, we express the binomial coefficient as a degree-q polynomial:

$$\binom{\omega(u,w)}{q} = \sum_{k=0}^{q} a_k \cdot \omega(u,w)^k,$$

where the coefficients a_k depend only on q.

Each monomial ω^k is estimated using the k-th moment of f_u, corrected for the bias introduced by its variance. The Gaussian assumption for f_u is justified by the Central Limit Theorem: since f_u is the sum of many independent random variables (randomized responses from u's neighbors plus Laplace noise), it converges to a Gaussian distribution as the vertex degree increases. This assumption is further validated empirically, as shown in Figure 3, where the Q-Q plot confirms that higher-order cumulants are indeed negligible, particularly for high-degree vertices. Based on the Gaussian assumption, we compute ω^k for each k using the following recurrence [19]:

$$\widehat{\omega^k} = f_u^k - \sum_{i=1}^{\lfloor k/2 \rfloor} \binom{k}{2i} (2i-1)!! \cdot f_u^{k-2i} \cdot \hat{\sigma}^{2i},$$

where $\hat{\sigma}^2$ is the estimated variance of f_u, and $(2i-1)!!$ denotes the double factorial. These corrected moments are substituted into the binomial expansion to obtain an unbiased estimate of $\mathbb{K}_{2,q}(u, w)$.

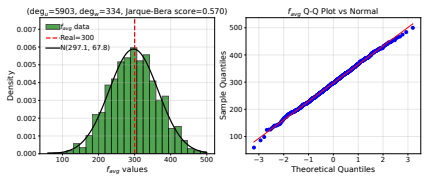


Fig. 3. Normality assumption check for the common neighbor estimators on the dataset `lrcwiki`.

Example (when $q = 3$): To estimate $\mathbb{K}_{2,3}(u, w) = \binom{\omega(u,w)}{3} = \frac{\omega^3 - 3\omega^2 + 2\omega}{6}$, we corrected moments derived from Equation : $\widehat{\omega} = f_u$, $\widehat{\omega^2} = f_u^2 - \hat{\sigma}^2$, and $\widehat{\omega^3} = f_u^3 - 3f_u \cdot \hat{\sigma}^2$. Substituting into the polynomial yields:

$$\widehat{\mathbb{K}}_{2,3}(u, w) = \frac{f_u^3 - 3f_u \cdot \hat{\sigma}^2 - 3f_u^2 + 2f_u}{6}.$$

The `MRCN` algorithm when $p = 2$. We present the `MRCN` algorithm for $(2, q)$-biclique counting under edge LDP in Algorithm 3. In *Phase 0*, each vertex adds Laplace noise to its degree. The degree estimation satisfies edge LDP through parallel composition (Theorem 3). Each vertex perturbs only its own adjacency list, and edges from u to v and v to u are treated as different secrets [16]. Since adjacency lists are disjoint subsets, the privacy cost remains ε_0 for all vertices. In *Phase 1*, a noisy bipartite graph is generated by independently perturbing each edge via randomized responses. In *Phase 2*, for each pair of upper-layer vertices (u, w), the algorithm estimates the number of common neighbors using a local estimator from u's perspective, denoted by f_u, and injects calibrated Laplace noise to preserve privacy. In *Phase 3*, `MRCN` computes the variance of f_u to construct an unbiased estimator for $\binom{\omega(u,w)}{q}$. These per-pair estimates are aggregated to yield the final unbiased estimate of the $(2, q)$-biclique count.

Algorithm 3: The MRCN algorithm ($p = 2$)

Input: G: a bipartite graph; ε: total privacy budget;
Output: Private estimate of the number of $(2, q)$-bicliques

Split ε into ε_0, ε_1, and ε_2;
// **Phase 0: Degree Estimation**
foreach $u \in V(G)$ **do**
  $\tilde{d}(u) \leftarrow deg(u, G) + \mathrm{Lap}(1/\varepsilon_0)$;
// **Phase 1: Noisy Graph Construction**
$p_r \leftarrow \frac{1}{1+e^{\varepsilon_1}}$;
foreach $i \in U(G)$ **do**
  foreach $j \in L(G)$ **do**
    $\mathcal{A}'[i,j] \leftarrow \begin{cases} 1 - \mathcal{A}[i,j], & \text{w.p. } p_r \\ \mathcal{A}[i,j], & \text{w.p. } 1 - p_r \end{cases}$;
// **Phase 2: Local Common Neighbor Estimation**
$\gamma \leftarrow \frac{1-p_r}{1-2p_r}$;
foreach $u, w \in U(G'), u < w$ **do**
  $S_1 \leftarrow 0$; $S_2 \leftarrow 0$;
  foreach $v \in N(u, G)$ **do**
    if $\mathcal{A}'[v, w] = 1$ **then**
      $S_1 \leftarrow S_1 + 1$;
    else
      $S_2 \leftarrow S_2 + 1$;
  $f_u \leftarrow \gamma \cdot S_1 - \frac{p_r}{1-2p_r} \cdot S_2 + \mathrm{Lap}\left(\frac{1-p_r}{(1-2p_r)\varepsilon_2}\right)$;
// **Phase 3: Moment Correction and Biclique Estimation**
$\hat{\mathbb{K}}_{2,q} \leftarrow 0$;
foreach $u, w \in U(G'), u < w$ **do**
  $\hat{\mathrm{Var}}(f_u) \leftarrow \frac{p_r(1-p_r)(\tilde{d}(u))}{(1-2p_r)^2} + \frac{2(1-p_r)^2}{(1-2p_r)^2\varepsilon_2^2}$;
  $\widehat{\binom{\omega(u,w)}{q}} \leftarrow$ unbiased estimate via moment-based correction;
  $\hat{\mathbb{K}}_{2,q} \leftarrow \hat{\mathbb{K}}_{2,q} + \widehat{\binom{\omega(u,w)}{q}}$;
return $\hat{\mathbb{K}}_{2,q}$;

4.2 The MRCN algorithm

In this subsection, we present the MRCN algorithm for general values of p, which relies on estimating the number of common neighbors for each p-tuple of $U(G)$. These per-tuple common neighbor estimators, along with their higher moments, are aggregated to form an unbiased estimate of the (p, q)-biclique count.

Estimating common neighbors for each p-tuple in $U(G)$. Assume that the degree estimates and the noisy graph G_{ε_1} are constructed in previous rounds. Consider each p-tuple $X = \{u_1, u_2, \cdots, u_p\}$ in $U(G)$. Without loss of generality, let u_1 be the vertex in X with the smallest estimated degree. We obtain the unbiased estimator for $\omega(X)$ in the following theorem.

Theorem 5. *Let $X = \{u_1, u_2, \ldots, u_p\} \subseteq U(G)$ be a p-tuple of vertices. Assume that $\phi(v, u_j)$ is an unbiased estimator of the true adjacency $A[v, u_j] \in \{0, 1\}$, i.e., $\mathbb{E}[\phi(v, u_j)] = A[v, u_j]$. Then, the*

estimator

$$f_1 = \sum_{v \in N(u_1)} \prod_{j>1} \phi(v, u_j)$$

is an unbiased estimator of the number of common neighbors of $u_1, u_2, \ldots, u_p$, i.e., $\omega(X)$.

Proof. For each $v \in N(u_1)$, define $S_v = \prod_{j>1} A[v, u_j]$. Thus, the number of common neighbors among X is $\omega(X) = \sum_{v \in N(u_1)} S_v$. Consider the estimator $f_1 = \sum_{v \in N(u_1)} \prod_{j>1} \phi(v, u_j)$. By independence and unbiasedness of each $\phi(v, u_j)$,

$$\mathbb{E}[f_1] = \sum_{v \in N(u_1)} \prod_{j>1} \mathbb{E}[\phi(v, u_j)] = \sum_{v \in N(u_1)} \prod_{j>1} A[v, u_j] = \omega(X).$$

This completes the proof. □

To release f_1 under edge LDP, we apply the Laplace mechanism. Let $\gamma = \frac{1-p_r}{1-2p_r}$. Since each $\phi(v, u_j)$ is bounded by γ, the global sensitivity of f_1 is bounded by $\Delta(f_1) \le \gamma^{p-1}$.

Theorem 6 (Global Sensitivity of f_1 Estimator). *Let $f_1 = \sum_{v \in N(u_1)} \prod_{j>1} \phi(v, u_j)$ be the common neighbor estimator for a p-tuple $X = \{u_1, u_2, \ldots, u_p\}$, where each $\phi(v, u_j)$ is bounded by $\gamma = \frac{1-p_r}{1-2p_r}$. Then the global sensitivity of f_1 under edge LDP is bounded by: $\Delta(f_1) \le \gamma^{p-1}$ for all $p \ge 1$.*

Proof. Since each $\phi(v, u_j)$ is bounded by γ, each term $\prod_{j>1} \phi(v, u_j)$ is bounded by γ^{p-1} (as there are $(p-1)$ factors). Under edge LDP, the global sensitivity is: $\Delta(f_1) = \max_{\mathcal{A}_u, \mathcal{A}'_u} |f_1(\mathcal{A}_u) - f_1(\mathcal{A}'_u)|$ where $\mathcal{A}_u$ and $\mathcal{A}'_u$ are neighboring neighbor lists differing by one bit. When neighbor lists differ by one bit, the maximum change in f_1 is bounded by the maximum value of any single term $\prod_{j>1} \phi(v, u_j)$, which is γ^{p-1}. □

Thus, an unbiased differentially private estimator for $\omega(X)$ is $\hat{f}_1 = \sum_{v \in N(u_1)} \prod_{j>1} \phi(v, u_j) + \mathrm{Lap}\left(\frac{\gamma^{p-1}}{\varepsilon_2}\right)$. This generalizes the per-vertex-pair common neighbor estimator from Section 4.1. To estimate the number of (p, q)-bicliques containing X, we must properly estimate $\binom{\omega(X)}{q}$. Directly substituting $\hat{f}_1$ into $\binom{\hat{f}_1}{q}$ introduces significant bias, since expectation is not preserved under nonlinear transformations. To correct this, as in the case $p = 2$, we estimate $\mathrm{Var}(\hat{f}_1)$ and apply a moment-based bias adjustment. Define $\theta = \frac{p(1-p_r)}{1-2p_r}$ and $\gamma = \frac{1-p_r}{1-2p_r}$. We carefully analyze the variance of $\hat{f}_1$ as follows.

$$\mathrm{Var}(\hat{f}_1) = \sum_{v \in N(u_1)} \mathrm{Var}\left(\prod_{j>1} \phi(v, u_j)\right) + 2 \cdot \frac{\gamma^{2p-2}}{\varepsilon_2^2}$$

We investigate the first term on the right hand side:

$$\begin{aligned}
\sum_{v \in N(u_1)} \mathrm{Var}\left(\prod_{j>1} \phi(v, u_j)\right) &= \sum_{v \in N(u_1)} \left(\prod_{j>1} \left(\mathrm{Var}(\phi(v, u_j)) + \mathcal{A}[v, u_j]^2\right) - \prod_{j>1} \mathcal{A}[v, u_j]^2\right) \\
&= \sum_{v \in N(u_1)} \left(\prod_{j>1} \left(\mathcal{A}[v, u_j]^2 + \theta\right) - \prod_{j>1} \mathcal{A}[v, u_j]^2\right) \\
&= \sum_{\substack{Y \subsetneq X \\ u_1 \notin Y}} \theta^{p-1-|Y|} \sum_{v \in N(u_1)} \prod_{u_j \in Y} \mathcal{A}[v, u_j]^2
\end{aligned}$$

Note that $\sum_{v\in N(v_1)}\prod_{u_j\in Y}\mathcal{A}[u_j,v]$ corresponds to the number of common neighbors among the vertex set $Y\cup\{v_1\}$. Under edge LDP, this can be unbiasedly estimated by $\sum_{v\in N(v_1)}\prod_{u_j\in Y}\phi(v,u_j)+\text{Lap}\left(\frac{\gamma^{|Y|}}{\varepsilon_2}\right)$. Therefore, $\text{Var}(\hat{f}_1)$ can be estimated via:

$$\sum_{\substack{Y\subseteq X\\ v_1\notin Y}}\theta^{p-1-|Y|}\left(\sum_{v\in N(u_1)}\prod_{u_j\in Y}\phi(v,u_j)+\text{Lap}\left(\frac{\gamma^{|Y|}}{\varepsilon_2}\right)\right)+2\frac{\gamma^{2p-2}}{\varepsilon_2^2}. \tag{2}$$

Using $\hat{f}_1$ computed from the neighborhood of u_1 and its estimated variance $\text{Var}(\hat{f}_1)$, we apply the same moment-correction technique as in the case $p=2$ to approximate higher moments of $\hat{f}_1$ and estimate the number of (p,q)-bicliques containing X.

Algorithm 4: The MRCN algorithm

Input: G: a bipartite graph; ε: total privacy budget; p,q: biclique parameters
Output: Private estimate of the number of (p,q)-bicliques

Split ε into ε_0, ε_1, and ε_2;
// **Phase 0: Degree Estimation**
foreach $u\in V(G)$ **do**
  $\tilde{d}(u)\leftarrow deg(u,G)+\text{Lap}(1/\varepsilon_0)$;
// **Phase 1: Noisy Graph Construction**
$p_r\leftarrow\frac{1}{1+e^{\varepsilon_1}}$, $\gamma\leftarrow\frac{1-p_r}{1-2p_r}$;
foreach $i\in U(G)$ **do**
  foreach $j\in L(G)$ **do**
    $\mathcal{A}'[i,j]\leftarrow\begin{cases}1-\mathcal{A}[i,j], & \text{with probability } p_r\\ \mathcal{A}[i,j], & \text{with probability } 1-p_r\end{cases}$;
// **Phase 2: Common Neighbor Estimation**
Initialize $\hat{\mathbb{K}}_{p,q}\leftarrow 0$;
foreach *p-tuple* $X=(x_1,x_2,\ldots,x_p)$ *from* $U(G)$ **do**
  $S\leftarrow 0$;
  $x_1\leftarrow$ the vertex with the smallest estimated degree in x_i;
  foreach $v\in N(x_1,G)$ **do**
    $prod\leftarrow 1$;
    for $j=2$ **to** p **do**
      $\phi\leftarrow\frac{\mathcal{A}'[v,x_j]-p_r}{1-2p_r}$;
      $prod\leftarrow prod\times\phi$;
    $S\leftarrow S+prod$;
  $\hat{f}_1\leftarrow S+\text{Lap}(\frac{\gamma^p}{\varepsilon_2})$;
  $\text{Var}(\hat{f}_1)\leftarrow$ estimate variance of $\hat{f}_1$ from Equation 2;
  $\widehat{\binom{\omega(X)}{q}}\leftarrow$ unbiased estimate via moment-based correction;
  $\hat{\mathbb{K}}_{p,q}\leftarrow\hat{\mathbb{K}}_{p,q}+\widehat{\binom{\omega(X)}{q}}$;
return $\hat{\mathbb{K}}_{p,q}$;

The MRCN algorithm. Algorithm 4 presents the MRCN algorithm based on the discussion above. Phase 0 estimates vertex degrees, and Phase 1 constructs a noisy graph via randomized response. In Phase 2, MRCN iterates over each p-tuple in the upper layer. For each tuple X, it selects the

vertex with the smallest estimated degree and traverses its neighbors to estimate the number of common neighbors $\omega(X)$. MRCN computes the variance of the estimator and applies a moment-based correction to obtain unbiased estimates of higher-order moments. Using these, MRCN constructs an unbiased estimator for $\binom{\omega(X)}{q}$ and aggregates the results across all p-tuples to produce the final unbiased estimate of the (p,q)-biclique count.

Computational time complexity. Randomized responses in MRCN takes $O\left(n_1 n_2 \cdot \frac{1}{1+e^{\varepsilon_1}}\right)$. Instead of enumerating all (p,q)-motifs as in OneR, MRCN processes each p-tuple by selecting a source vertex, exploring its neighbors, and computing the product of $\phi(v, x_j)$ over the remaining vertices in the tuple. This costs $O(d^1_{\max} \cdot p)$ per p-tuple, with $\binom{n_1}{p}$ tuples overall ($d^1_{\max}$ is the maximum degree on $U(G)$). Hence, the total time complexity is $O\left(\frac{n_1 n_2 + \binom{n_1}{p} d^1_{\max}}{1+e^{\varepsilon_1}}\right)$.

Per-vertex Communication costs. The dominant communication cost of MRCN lies in downloading the noisy graph. Its expected size is $m(1-p_r) + (n_1 n_2 - m)p_r = O\left(\frac{n_1 n_2}{1+e^{\varepsilon}}\right)$. As each noisy edge requires $O(\log(n_1 n_2))$ bits, the per-vertex communication cost is $O\left(\frac{n_1 n_2 \log(n_1 n_2)}{1+e^{\varepsilon}}\right)$.

4.3 Optimizations

In this section, we present several optimizations to enhance the data utility and efficiency of MRCN by introducing *multi-center optimization* and *refined noisy graph construction*.

4.3.1 Multi-center optimization. To reduce the variance of biclique estimators under edge LDP, we propose a *multi-center optimization* that aggregates estimators from all vertices in each p-tuple, rather than relying on a single center vertex as in the basic MRCN algorithm. While the single-center approach yields an unbiased estimate, constructing estimators $f_1, f_2, \ldots, f_p$ from each vertex in the p-tuple and averaging them improves both accuracy and robustness. Based on our discussion in Section 4.2, for each vertex v_i, we can compute the estimator f_i, as given by $f_i = \sum_{u \in N(v_i)} \prod_{j \neq i} \phi(u, v_j) + \text{Lap}\left(\frac{\gamma^{p-1}}{\varepsilon_2}\right)$, where $\phi(u, v_j)$ is the unbiased estimator $\mathcal{A}[u, v_j]$, and $\gamma = \frac{1-p_r}{1-2p_r}$. We then average these: $f_{\text{avg}} = \frac{1}{p}\sum_{i=1}^{p} f_i$, which remains unbiased due to the linearity of expectation.

To accurately estimate the variance of f_{avg}, we explicitly account for the covariance between each pair (f_i, f_j). Let $\text{Cov}(f_i, f_j)$ denote the covariance between f_i and f_j, which arises due to overlapping edge estimators. We model these using intermediate cross terms (e.g., f_{ij} and f_{ji}), computed as partial sums over $\phi(u, v_j)$ from v_i's neighborhood and vice versa. The variance of f_{avg} is: $\text{Var}(f_{\text{avg}}) = \frac{1}{p^2}\left(\sum_{i=1}^{p} \text{Var}(f_i) + 2\sum_{i<j} \text{Cov}(f_i, f_j)\right)$. This is used in the moment-correction technique for estimating the per-tuple biclique counts. Here, we illustrate how to compute $\text{Var}(f_{\text{avg}})$ when $p = 2$ and $p = 3$.

(1) When $p = 2$: For the 2-tuple (u_1, u_2), we construct estimators $\hat{f}_1$ and $\hat{f}_2$ based on the neighborhoods of u_1 and u_2, respectively. Specifically, $\hat{f}_1$ depends only on entries $\mathcal{A}[v_i, u_2]$ where $v_i \in N(u_1)$, and $\hat{f}_2$ depends only on entries $\mathcal{A}[v_i, u_1]$ where $v_i \in N(u_2)$. Since these entries are disjoint in the noisy graph, $\hat{f}_1$ and $\hat{f}_2$ are independent, implying $\text{Cov}(\hat{f}_1, \hat{f}_2) = 0$. Thus, the variance of the average estimator can be computed as $\text{Var}(f_{\text{avg}}) = \frac{1}{4}\left(\text{Var}(\hat{f}_1) + \text{Var}(\hat{f}_2)\right)$.

(2) When $p > 2$: For $p > 2$, the estimators $\hat{f}_i$ are no longer independent due to shared edges in overlapping neighborhoods. This correlation arises because different estimators depend on common edges. For example, in a triple (u_1, u_2, u_3), both $\hat{f}_1$ and $\hat{f}_2$ depend on edges from vertices in $N(u_1) \cap N(u_2)$ to u_3, creating shared random variables that make the estimators correlated. To handle this, we need to estimate and account for the covariances between estimators.

As an example, for $p = 3$, we systematically estimate the covariances by computing cross-terms $f_{12}, f_{21}, f_{13}, f_{31}, f_{23}, f_{32}$ that capture the pair-wise common neighbors:

$$f_{12} = \sum_{v \in N(u_1)} \phi(v, u_2) + \text{Lap}(\frac{\gamma}{\varepsilon_2}), \quad f_{21} = \sum_{v \in N(u_2)} \phi(v, u_1) + \text{Lap}(\frac{\gamma}{\varepsilon_2}),$$

$$\text{Cov}(\hat{f}_1, \hat{f}_2) \approx \text{Var}(\phi) \cdot \frac{f_{12} + f_{21}}{2}.$$

The final averaged estimator and its estimated variance are:

$$\hat{f}_{\text{avg}} = \frac{\hat{f}_1 + \hat{f}_2 + \hat{f}_3}{3}, \quad \text{Var}(\hat{f}_{\text{avg}}) = \frac{1}{9}\left(\sum_{i=1}^{3} \text{Var}(\hat{f}_i) + 2\sum_{i<j} \text{Cov}(\hat{f}_i, \hat{f}_j)\right).$$

4.3.2 Refined noisy graph construction. Edge LDP algorithms often apply randomized responses to either the upper or lower triangle of the adjacency matrix to avoid conflicts in the noisy graph. However, this halves the usable data and limits estimation accuracy. We propose the *Refined noisy graph construction*, which applies randomized responses to each vertex in G and uses the upper and the lower triangle of the noisy adjacency matrix to build two independent noisy graphs. This construction relies on the independence assumption for parallel composition. Each edge (u, v) is flipped twice—once from u's adjacency list to obtain $\mathcal{A}'[u, v]$ and once from v's list to obtain $\mathcal{A}'[v, u]$—producing independent entries in the upper and lower triangles. Under edge LDP, these are treated as distinct secrets [16], so the construction remains compliant with edge LDP. This still complies with edge LDP by parallel composition and post-processing properties [28]. The parallel composition is justified by Theorem 3, since the two randomized response mechanisms operate on disjoint data (each user's own adjacency list), ensuring that the overall privacy guarantee is preserved. This allows us to improve the unbiased edge estimator $\phi(i, j) = \frac{\mathcal{A}'[i,j] - p_r}{1-2p_r}$ by averaging over the two noisy graphs: $\phi^{\text{avg}}(i, j) = \frac{1}{2}\left(\phi^{(1)}(i, j) + \phi^{(2)}(i, j)\right)$, which remains unbiased and has half the variance: $\text{Var}(\phi^{\text{avg}}(i, j)) = \frac{1}{2}\text{Var}(\phi(i, j))$. Since many estimators for common neighbors and (p, q)-bicliques are based on $\phi(i, j)$, this directly improves estimation accuracy.

4.3.3 Expected L2 Loss Analysis. We analyze the expected L2 loss of MRCN under different optimization strategies.

(1) Variance decomposition. The total (p, q)-biclique estimator can be expressed as the sum of per-p-tuple estimators: $\hat{\mathbb{K}}_{p,q} = \sum_{X \in \binom{U(G)}{p}} \hat{B}_X$, where $\hat{B}_X$ is the biclique estimator for the p-tuple X. The variance satisfies $\text{Var}[\hat{\mathbb{K}}_{p,q}] = \sum_X \text{Var}[\hat{B}_X] + \sum_{X \neq X'} \text{Cov}(\hat{B}_X, \hat{B}_{X'})$. Since cross-covariances are non-negative, we obtain a general lower bound: $\text{Var}[\hat{\mathbb{K}}_{p,q}] \geq \sum_{X \in \binom{U(G)}{p}} \text{Var}[\hat{B}_X]$. Hence, analyzing $\text{Var}[\hat{B}_X]$ for each p-tuple gives a lower bound on the total variance.

(2) Modeling the per-tuple estimator. For each p-tuple $X = \{u_1, u_2, \ldots, u_p\}$, the estimator $\hat{B}_X$ depends on the common-neighbor estimator f_{avg}, which approximates a Gaussian variable: $f_{\text{avg}} \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = \omega(X)$ is the true number of common neighbors, and $\sigma^2 = \text{Var}(f_{\text{avg}})$ is the estimation variance.

(3) Deriving $\text{Var}(f_{\text{avg}})$ for $p = 2$. When $p = 2$, we have $f_{\text{avg}} = \frac{1}{2}(f_u + f_w)$, where f_u and f_w are independent. Thus, $\text{Var}(f_{\text{avg}}) = \frac{1}{4}(\text{Var}(f_u) + \text{Var}(f_w))$. For each $i \in \{u, w\}$, the per-vertex estimator variance is $\text{Var}(f_i) = \frac{p_r(1-p_r)}{(1-2p_r)^2}\deg(i) + \frac{2(1-p_r)^2}{(1-2p_r)^2\varepsilon_2^2}$. Combining the two gives

$$\text{Var}(f_{\text{avg}}) = \frac{p_r(1-p_r)}{4(1-2p_r)^2}(\deg(u) + \deg(w)) + \frac{(1-p_r)^2}{(1-2p_r)^2\varepsilon_2^2}.$$

This variance encapsulates both the sampling noise (via p_r) and Laplace noise (via ε_2).

(4) Applying the delta method. To estimate the true biclique count $\binom{\omega}{q}$ from f_{avg}, we apply the delta method for variance propagation. For a Gaussian variable $X \sim \mathcal{N}(\mu, \sigma^2)$ and a smooth function h,

$$\text{Var}[h(X)] \approx (h'(\mu))^2\sigma^2 + \tfrac{1}{2}(h''(\mu))^2\sigma^4.$$

We next derive $\text{Var}[\hat{B}_X]$ for $q = 2$ and $q = 3$.

Case $q = 2$ (butterflies). We estimate $\binom{\omega}{2} = \frac{\omega(\omega-1)}{2}$ using $h(x) = \frac{x(x-1)}{2}$. Then $h'(\mu) = \mu - \frac{1}{2}$ and $h''(\mu) = 1$. Substituting into the delta approximation:

$$\text{Var}[\hat{B}_X] = \tfrac{\sigma^2}{4}\left(2\sigma^2 + 1 + 4\mu(\mu-1)\right).$$

Case $q = 3$. For $\binom{\omega}{3} = \frac{\omega(\omega-1)(\omega-2)}{6}$, let $h(x) = \frac{x(x-1)(x-2)}{6}$. Then $h'(\mu) = \frac{3\mu^2-6\mu+2}{6}$ and $h''(\mu) = \mu - 1$. Applying the same procedure:

$$\text{Var}[\hat{B}_X] = \tfrac{\sigma^2}{6}\left(3\sigma^2 + 1 + 6\mu(\mu-1)(\mu-2)\right).$$

(5) Lower bounds for total variance. Substituting the per-tuple variance expressions, we obtain lower bounds for the total variance.

Butterfly counting ($p = 2, q = 2$):

$$\text{Var}[\hat{\mathbb{K}}_{2,2}] \geq \sum_{u<w\in U(G)} \tfrac{\sigma^2_{u,w}}{4}\left(2\sigma^2_{u,w} + 1 + 4\omega(u,w)(\omega(u,w)-1)\right).$$

$(p = 2, q = 3)$ biclique counting:

$$\text{Var}[\hat{\mathbb{K}}_{2,3}] \geq \sum_{u<w\in U(G)} \tfrac{\sigma^2_{u,w}}{6}\left(3\sigma^2_{u,w} + 1 + 6\omega(u,w)(\omega(u,w)-1)(\omega(u,w)-2)\right).$$

5 Experimental evaluation

In this section, we evaluate the effectiveness and efficiency of the proposed (p, q)-biclique counting algorithms under ε-edge LDP through experiments.

Table 2. Summary of Datasets

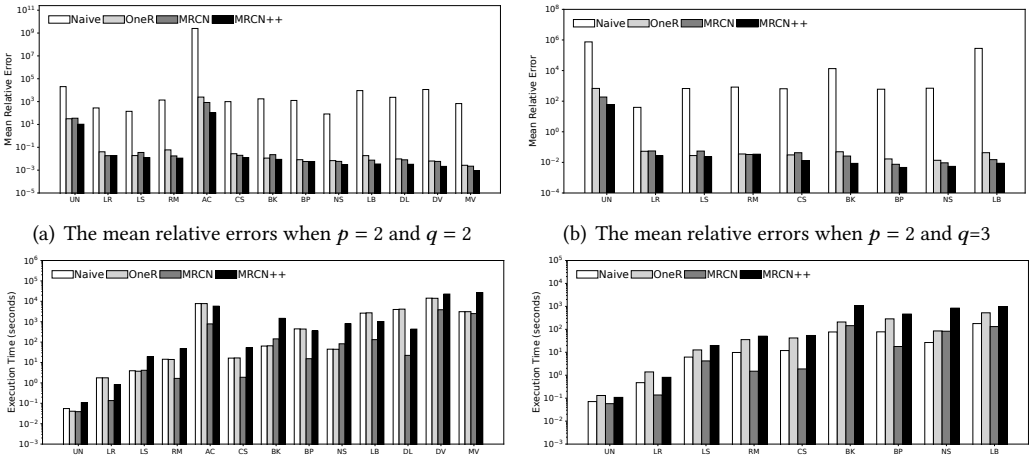
Dataset	Upper	Lower	$\lvert E\rvert$	$\lvert U\rvert$	$\lvert L\rvert$	Density
Unicode (UN)	Country	Language	1.26K	255	615	0.008
lrcwiki (LR)	User	Article	18.9K	209	8.74K	0.010
filmtrust (LS)	User	Film	35.5K	1.5K	2.07K	0.011
rmwiki (RM)	User	Article	57.9K	1.22K	8.09K	0.006
Condmat (AC)	Author	Paper	58.6K	16.7K	22.0K	0.0002
csbwiki (CS)	User	Article	62.1K	1.33K	8.28K	0.006
bag-kos (BK)	Document	Word	353K	3.43K	6.90K	0.015
bpywiki (BP)	User	Article	399.7K	1.32K	57.9K	0.005
NIPS (NS)	Document	Word	746.3K	1.50K	12.4K	0.040
jester (JS)	User	Joke	1.73M	52	50.7K	0.656
lastfm band (LB)	User	Band	898K	38.7M	16.7M	1.4e-9
Discogs (DL)	Label	Style	1.06M	243.8K	0.38K	0.011
DiggVotes (DV)	User	Story	3.01M	139.4K	3.55K	0.006

Datasets. We use 13 datasets from KONECT (http://konect.cc/). Table 2 shows the statistics of the datasets. $|U|$ and $|L|$ are the number of vertices in the upper and lower layers. $|E|$ is the number of edges in the graph.

Algorithms. We evaluate the following (p, q)-biclique counting algorithms under edge LDP. (1) `Naive`: the algorithm directly returning the (p, q)-biclique count on the noisy graph; (2) `OneR`:

the one-round algorithm that computes unbiased estimates by counting (p, q)-vertex motifs in the noisy graph; (3) MRCN: a common neighbor estimator based multi-round algorithm; (4) MRCN+: MRCN with multi-center aggregation; (5) MRCN++: MRCN with refined noisy graph construction and multi-center aggregation; All algorithms are implemented in C++. The experiments are run on a Linux server with an Intel Xeon 6342 processor and 512GB of memory.

Parameter settings. By default, the privacy budget ε is set to 1. We allow it to vary from 0.4 to 1.6 with increments of 0.2. For MRCN algorithms, we allocate the privacy budget as follows: $\varepsilon_0 = 0.05\varepsilon$, $\varepsilon_1 = 0.6\varepsilon$, and $\varepsilon_2 = 0.35\varepsilon$. When $p > 2$, we apply vertex sampling to improve the efficiency of p-tuple enumeration. Let τ denote the number of p-tuples sampled, which is set to 10^6 by default.



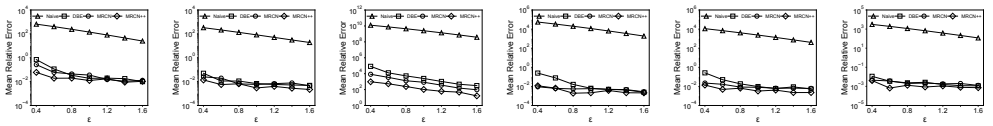
(a) The mean relative errors when $p = 2$ and $q = 2$

(b) The mean relative errors when $p = 2$ and q=3

(c) The computational time cost when $p = 2$ and $q = 2$

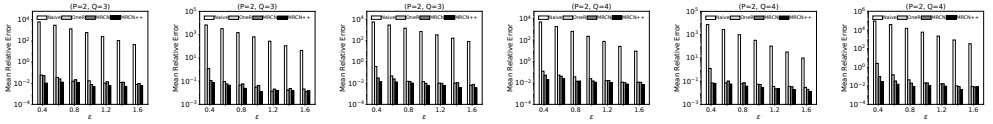
(d) The computational time cost when $p = 2$ and $q = 3$

Fig. 4. Evaluate the performance of all algorithms on different datasets ($\varepsilon = 1$)



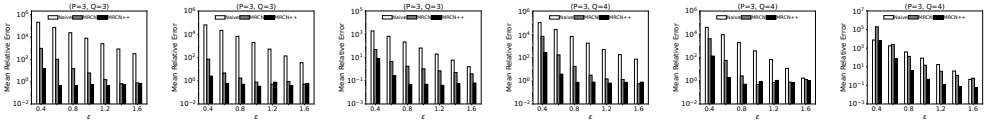
(a) LS (b) NS (c) AC (d) DV (e) DL (f) MV

Fig. 5. Evaluate the effect of ε on all algorithms when $p = 2$ and $q = 2$ (butterfly counting).



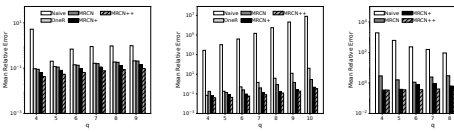
(a) BP ($q = 3$) (b) CS ($q = 3$) (c) NS ($q = 3$) (d) BP ($q = 4$) (e) CS ($q = 4$) (f) NS ($q = 4$)

Fig. 6. Evaluate the effect of ε on all algorithms when $p = 2$ and $q = 3, 4$.



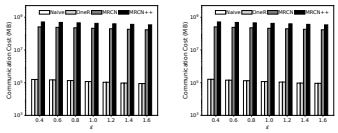
(a) LR ($q = 3$) (b) CS ($q = 3$) (c) LS ($q = 3$) (d) LR ($q = 4$) (e) CS ($q = 4$) (f) LS ($q = 4$)

Fig. 7. Evaluate the effect of ε for $(p = 3, q = 3)$ and $(p = 3, q = 4)$ on 3 datasets.



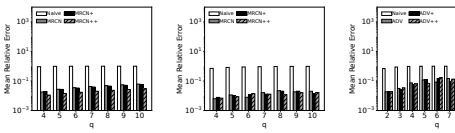
(a) LR ($p = 2$) (b) LS ($p = 2$) (c) LR ($p = 3$) (d) LS ($p = 3$)

Fig. 8. Evaluate the performance of the algorithms with larger values of p and q ($p \in \{2, 3\}$, $q \in \{4, 5, \ldots, 10\}$, $\varepsilon = 1$).



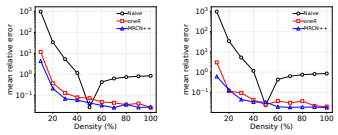
(a) LS ($q = 2$) (b) LS ($q = 3$)

Fig. 9. Evaluate the communication costs on LS ($p = 2, q = 2, 3$).



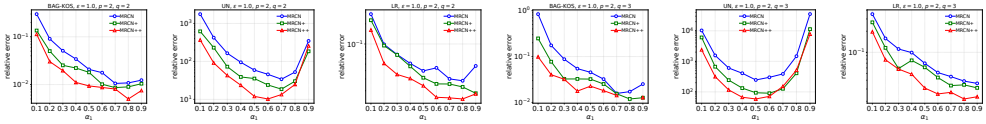
(a) $p = 3, \varepsilon = 1$ (b) $p = 3, \varepsilon = 2$ (c) $p = 4, \varepsilon = 1$ (d) $p = 4, \varepsilon = 2$

Fig. 10. Evaluate the performance of algorithms on JS dataset ($n_1 = 52$, $n_2 = 50,693$, density = 0.656).



(a) 100×200 (b) 150×300

Fig. 11. Evaluate the effect of graph density ($p = 2, q = 3$).



(a) BK ($q = 2$) (b) UN ($q = 2$) (c) LR ($q = 2$) (d) BK ($q = 3$) (e) UN ($q = 3$) (f) LR ($q = 3$)

Fig. 12. Evaluate the effect of privacy budget allocation when $p = 2, q = 2, 3$.

Evaluate the performance of (p, q)-biclique counting algorithms under edge LDP across datasets. In Fig. 4(a) and Fig. 4(b), we report the performances of Naive, OneR, MRCN, and MRCN++ across different datasets when $\varepsilon = 1$, $p = 2$, and $q = 2, 3$. First, OneR, MRCN, and MRCN++ significantly outperform Naive across all datasets. This is because these algorithms produce unbiased estimates of (p, q)-bicliques, which addresses the large bias issue in Naive. On most datasets, MRCN and MRCN++ further outperform OneR. This is because the novel multi-round algorithms MRCN and MRCN++ estimate the number of common neighbors for each p-tuple in the upper layer or q-tuple in the lower layer of G, and leverage robust common neighbor estimators and their higher moments to derive unbiased per-tuple biclique counts. Also, MRCN++ consistently improves the data utility of MRCN. This is because MRCN++ adopts multi-center optimization and refined noisy graph

construction to reduce the variance of the common neighbor estimators and resulting per-tuple biclique estimators.

Evaluate the computational time costs across datasets. In Fig. 4(c) and Fig. 4(d), we report the computational time costs of Naive, OneR, MRCN and MRCN++ when $(p = 2, q = 2)$ and $(p = 2, q = 2)$, $\varepsilon = 1$. Naive and OneR incur higher time costs on sparse graphs like AC and LB, due to the combinatorial cost of counting (p, q)-bicliques on the noisy graph. In contrast, MRCN and MRCN++ avoid full motif enumeration and leverage local common neighbor estimations with cost proportional to $\binom{n_1}{p} \cdot d_{\max}^{q-1}$, which is significantly lower when the graph is sparse. OneR generally incurs higher time costs than Naive. Although both methods apply randomized response with the full privacy budget, their counting phase differ: Naive counts only the (p, q)-bicliques on the noisy graph, whereas OneR must enumerate all motifs with p upper and q lower vertices. This becomes more pronounced when $q = 3$, as the number of possible $(2, 3)$-vertex motifs significantly exceeds that of $(2, 2)$-vertex motifs. MRCN++ incurs higher runtime than MRCN because it estimates common neighbors from multiple source vertices and constructs two noisy graphs from the perturbed adjacency matrix.

Evaluate the effect of the privacy budget ε. In Fig. 5, Fig. 6, and Fig. 7, we report the mean relative errors of Naive, OneR, MRCN, and MRCN++ across datasets, as ε varies between 0.4 and 1.6. Note that we provide an efficient implementation of the OneR algorithm only for the case $p = 2$, as extending it to general $p > 2$ is challenging due to the intractable number of distinct (p, q)-motif types. Overall, the data utility of all algorithms improves as the privacy budget increases. Specifically, MRCN, MRCN++, and OneR consistently outperform Naive by orders of magnitude across various (p, q) settings, as they rely on provably unbiased estimators rather than directly counting bicliques on the noisy graph. As shown in Fig. 5 and Fig. 6, when $p = 2$, MRCN and MRCN++ generally yield lower mean relative errors than OneR, with the advantage becoming more pronounced on sparse graphs. OneR results in high variance because each (p, q)-biclique depends on roughly $p \times q$ independent Bernoulli variables. MRCN and MRCN++ address the issue of OneR by allowing the vertices to download the noisy graph and estimate the number of common neighbors for each vertex pair (u, w) locally, which involves $deg(u)$ or $deg(w)$ random variables. In Fig. 7 ($p = 3$), both MRCN and MRCN++ significantly outperform the Naive algorithm. We omit OneR as it requires enumerating all motifs with p upper and q lower vertices, which is infeasible when $p > 2$.

Evaluate the performance of all algorithms for larger values of q. In Fig. 8, we test MRCN, MRCN+, and MRCN++ on larger q values ranging from 4 to 10 across two datasets (lrcwiki, filmtrust) for $p = 2$ and $p = 3$ with $\varepsilon = 1$. In Fig. 10, we further evaluate our algorithms on the jester dataset, a dense real-world bipartite graph, testing different values of $p \in \{3, 4\}$ and $\varepsilon \in \{1, 2\}$ with q ranging from 4 to 10. Note that we do not include OneR when $pgeq3$ due to its long running time from enumerating all motifs with p upper and q lower vertices on the noisy graph. First, we observe that the MRCN algorithms and OneR consistently outperform Naive on larger q values. This is because Naive introduces significant bias from directly querying the noisy graph. Second, when $p = 2$, the MRCN algorithms generally outperform OneR because OneR results in high variance as each (p, q)-biclique depends on roughly $p \times q$ independent Bernoulli variables, while MRCN algorithms leverage local common neighbor estimation with fewer random variables and incorporate variance-reduction techniques like multi-center optimization and refined noisy graph construction. Also, we can see that MRCN+ and MRCN++ progressively achieve better performance across both experiments. This shows that our proposed multi-center optimization and refined noisy graph construction techniques are effective on larger q values as well, even on dense real-world graphs.

Evaluate the communication costs of all algorithms. In Fig. 9, we report the per-vertex communication costs (in MB) of each algorithm as ε varies on the filmtrust dataset. Fig. 9 shows results for $(p = 2, q = 2)$ and $(p = 2, q = 3)$. We observe that Naive and OneR incur nearly identical

communication costs. This is because both algorithms use the randomized responses with identical flip probabilities. In contrast, MRCN and MRCN++ incur higher communication costs because the multi-round protocol requires: (1) uploading noisy edges to the curator, (2) downloading noisy edges for all vertices, and (3) uploading the common-neighbor estimators to the data curator. MRCN++ incurs the highest communication costs from its refined noisy graph construction and multi-center optimization, sending more noisy edges and local estimators.

Evaluate the effect of privacy budget allocation. In Fig. 12, we evaluate how different privacy budget allocations affect the performance of MRCN, MRCN+, and MRCN++ when $\varepsilon = 1$. Without loss of generality, we set $p = 2$ and $q = 2, 3$. We use α_0, α_1, and α_2 to represent the fraction of ε allocated to degree estimated, randomized responses, and the Laplace mechanism. Specifically, we fix $\alpha_0 = 0.05$ and vary α_1 from 0.1 to 0.9, with $\alpha_2 = 1 - \alpha_0 - \alpha_1$. Across all datasets, we observe that MRCN++ consistently achieves the lowest relative error, followed by MRCN+ and MRCN. The optimal allocation of α_1 varies by dataset: smaller datasets (to, co, unicode) perform best with $\alpha_1 \approx 0.5$–0.6, while larger datasets (lrcwiki, csbwiki, filmtrust) benefit from higher allocations of $\alpha_1 \approx 0.7$–0.8. This is because larger graphs require more privacy budget for constructing accurate noisy graphs, as the number of edges scales with graph size. The default allocation of $\alpha_1 = 0.6$ provides competitive performance across all datasets.

Evaluate the effect of graph density. Fig. 11 illustrates how graph density affects algorithm performance on synthetic bipartite graphs generated using the Erdős–Rényi model. In this model, each potential edge (u, v) between n_1 and n_2 vertices is included independently with probability equal to the target density, ranging from 10% to 100%. We evaluate two graph sizes (100×200 and 150×300) with parameters $p = 2$, $q = 3$, and privacy budget $\varepsilon = 1$. For each density, five independent graph instances are generated, and each algorithm is executed 10 times per instance to account for randomness. Across all configurations, MRCN++ consistently achieves the lowest relative error. The performance gap increases with graph density: while Naive exhibits exponential growth in error, MRCN++ remains stable and accurate. These results demonstrate that our multi-round algorithm MRCN++, with all optimizations applied, is robust to variations in graph density and outperforms Naive and OneR across the full spectrum from sparse to dense bipartite graphs.

6 Related Work

We review the related works on (p, q)-biclique counting on bipartite graphs and motif counting under edge LDP.

(p, q)-biclique counting. Yang et al. [45] first systematically study (p, q)-biclique enumeration and counting and propose the BCList and BCList++ algorithms. BCList adopts a branch-and-bound framework with pruning but has exponential complexity in $p + q$. BCList++ reduces this to exponential in either p or q by anchoring the search to a single partition, incorporating a cost model, memory reuse, and parallelism for scalability. To avoid full enumeration, Ye et al. [47] propose EPivoter, which encodes bicliques via edge-pivoting, and two sampling-based methods, ZigZag and ZigZag++, using h-zigzag dynamic programming. To further accelerate counting, Qiu et al. [30] develop GBC, which uses hierarchical truncated bitmaps for efficient set intersections, a hybrid DFS-BFS search for better GPU parallelism, and a composite load-balancing scheme for workload distribution. The special case of (p, q)-biclique with $p = q = 2$ is known as a butterfly, and butterfly counting has been widely studied [22, 31, 32, 36, 39–41, 43].

Motif counting under edge-local differential privacy. Motif counting under CDP has been widely studied [3, 21, 25, 44, 48]. These works assume full access to the graph and are not applicable under edge LDP [7, 16–18, 24, 34]. Edge LDP [28] is a widely accepted privacy model for graph mining. In this setting, [16] proposes one-round and two-round triangle counting algorithms, further improved by [17] for better communication efficiency and accuracy. [7] offers a detailed

analysis of these techniques, while [14] introduces a vertex-centric algorithm that enhances data utility. Localized variants [24, 34] allow each vertex to report its 2-hop neighborhood but require a stronger LDP model. Beyond triangle motifs, 4-cycle counting has been studied under the shuffle model [18]. Beyond graph structures, LDP has been applied to other domains such as trajectory synthesis [5], demonstrating the broad applicability of local differential privacy. On bipartite graphs, He et al. [12] propose one-round and multi-round butterfly counting algorithms under edge LDP. The multi-round method extends triangle counting by estimating per-vertex butterfly counts, but suffers from high Laplace noise due to motif complexity. Since butterflies are $(2, 2)$-bicliques and general (p, q)-bicliques are more complex, we design a new multi-round algorithm that avoids per-vertex counts to improve accuracy under edge LDP. Also, [13] studies common neighbor counting on bipartite graphs. We build upon the multi-round estimator [13] for (p, q)-biclique counting, adapting it to share privacy budget allocation and weighting parameters across all p-tuples. A recent work by Sun et al. [33] proposes k-star LDP for (p, q)-biclique counting, but differs substantially from our approach: (1) [33] uses k-star adjacency lists rather than traditional neighbor lists in edge LDP, and (2) their edge LDP baseline lacks Laplace noise, violating edge LDP requirements. Therefore, we do not compare to [33] as it uses a different privacy model.

7 Conclusion

In this paper, we tackle (p, q)-biclique counting in bipartite graphs under edge LDP. The `Naive` algorithm directly counts (p, q)-bicliques on noisy graphs and results in significant bias. To address this, we introduce the `OneR` algorithm, which employs motif transformation probabilities to provide unbiased estimates. For enhanced generalizability and data utility, we introduce the `MRCN` algorithm, which refines common neighbor estimates for p-tuples in the upper layer or q-tuples in the lower layer, utilizing higher moments to achieve superior accuracy. Additionally, we propose refined noisy graph construction and multiple-center optimization to enhance accuracy further, as well as vertex sampling to improve efficiency. Extensive experiments on real-world datasets validate the effectiveness and efficiency of our proposed algorithms.

8 Acknowledgments

Kai Wang is the corresponding author. Kai Wang is supported by NSFC 62302294 and U2241211. Wenjie Zhang is supported by ARC DP230101445 and FT210100303. Xuemin Lin is supported by NSFC U2241211. Ying Zhang is supported by ARC LP210301046 and DP230101445. We would like to thank the anonymous reviewers for their valuable and constructive feedback.

References

[1] Xiaoshuang Chen, Kai Wang, Xuemin Lin, Wenjie Zhang, Lu Qin, and Ying Zhang. 2021. Efficiently answering reachability and path queries on temporal bipartite graphs. *Proceedings of the VLDB Endowment* (2021).

[2] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*. 123–138.

[3] Xiaofeng Ding, Shujun Sheng, Huajian Zhou, Xiaodong Zhang, Zhifeng Bao, Pan Zhou, and Hai Jin. 2021. Differentially private triangle counting in large graphs. *IEEE Transactions on Knowledge and Data Engineering* 34, 11 (2021), 5278–5292.

[4] Xiaofeng Ding, Xiaodong Zhang, Zhifeng Bao, and Hai Jin. 2018. Privacy-preserving triangle counting in large graphs. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1283–1292.

[5] Yuntao Du, Yujia Hu, Zhikun Zhang, Ziquan Fang, Lu Chen, Baihua Zheng, and Yunjun Gao. 2023. Ldptrace: Locally differentially private trajectory synthesis. *arXiv preprint arXiv:2302.06180* (2023).

[6] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[7] Talya Eden, Quanquan C Liu, Sofya Raskhodnikova, and Adam Smith. 2023. Triangle Counting with Local Edge Differential Privacy. *arXiv preprint arXiv:2305.02263* (2023).

[8] Xinyi Gao, Guanhua Ye, Tong Chen, Wentao Zhang, Junliang Yu, and Hongzhi Yin. 2025. Rethinking and accelerating graph condensation: A training-free approach with class partition. In *Proceedings of the ACM on Web Conference 2025*. 4359–4373.

[9] Xinyi Gao, Junliang Yu, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin. 2025. Graph condensation: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[10] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 169–178.

[11] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2009. Boosting the accuracy of differentially-private histograms through consistency. *arXiv preprint arXiv:0904.0942* (2009).

[12] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, Wei Ni, and Ying Zhang. 2024. Butterfly Counting over Bipartite Graphs with Local Differential Privacy. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2351–2364.

[13] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2024. Common Neighborhood Estimation over Bipartite Graphs under Local Differential Privacy. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–26.

[14] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Wei Ni. 2025. Robust Privacy-Preserving Triangle Counting under Edge Local Differential Privacy. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[15] Zan Huang, Xin Li, and Hsinchun Chen. 2005. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*. 141–142.

[16] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2021. Locally Differentially Private Analysis of Graph Statistics.. In *USENIX Security Symposium*. 983–1000.

[17] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-Efficient Triangle Counting under Local Differential Privacy. In *31st USENIX Security Symposium (USENIX Security 22)*. 537–554.

[18] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1505–1519.

[19] Leon Isserlis. 1918. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika* 12, 1/2 (1918), 134–139.

[20] Honglu Jiang, Jian Pei, Dongxiao Yu, Jiguo Yu, Bei Gong, and Xiuzhen Cheng. 2021. Applications of differential privacy in social network analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 108–127.

[21] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.

[22] Rundong Li, Pinghui Wang, Peng Jia, Xiangliang Zhang, Junzhou Zhao, Jing Tao, Ye Yuan, and Xiaohong Guan. 2021. Approximately counting butterflies in large bipartite graph streams. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2021), 5621–5635.

[23] Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. 2020. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1677–1688.

[24] Yuhan Liu, Suyun Zhao, Yixuan Liu, Dan Zhao, Hong Chen, and Cuiping Li. 2022. Collecting triangle counts with edge relationship local differential privacy. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2008–2020.

[25] Tianzi Lv, Huanzhou Li, Zhangguo Tang, Fangzhou Fu, Jian Cao, and Jian Zhang. 2021. Publishing Triangle Counting Histogram in Social Networks Based on Differential Privacy. *Security and Communication Networks* 2021 (2021), 1–16.

[26] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[27] Clare M O'Connor, Jill U Adams, and Jennifer Fairman. 2010. Essentials of cell biology. *Cambridge, MA: NPG Education* 1 (2010), 54.

[28] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 425–438.

[29] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 425–438.

[30] Linshan Qiu, Zhonggen Li, Xiangyu Ke, Lu Chen, and Yunjun Gao. 2024. Accelerating Biclique Counting on GPU. *arXiv preprint arXiv:2403.07858* (2024).

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

[32] Aida Sheshbolouki and M Tamer Özsu. 2022. sGrapp: Butterfly approximation in streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 4 (2022), 1–43.
[33] Henan Sun, Zhengyu Wu, Rong-Hua Li, Guoren Wang, and Zening Li. 2024. K-stars LDP: A Novel Framework for (p, q)-clique Enumeration under Local Differential Privacy. *arXiv preprint arXiv:2403.01788* (2024).
[34] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui Wang, and Ting Yu. 2019. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 703–717.
[35] Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. Paths-over-graph: Knowledge graph empowered large language model reasoning. In *Proceedings of the ACM on Web Conference 2025*. 3505–3522.
[36] Jia Wang, Ada Wai-Chee Fu, and James Cheng. 2014. Rectangle counting in large bipartite graphs. In *2014 IEEE International Congress on Big Data*. IEEE, 17–24.
[37] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-Trained Graph Transformer. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2227–2240.
[38] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural attributed community search at billion scale. *Proceedings of the ACM on Management of Data* 1, 4 (2024), 1–25.
[39] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2019. Vertex Priority Based Butterfly Counting for Large-Scale Bipartite Networks. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1139–1152.
[40] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient bitruss decomposition for large-scale bipartite graphs. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 661–672.
[41] Zhibin Wang, Longbin Lai, Yixue Liu, Bing Shui, Chen Tian, and Sheng Zhong. 2023. I/O-Efficient Butterfly Counting at Scale. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.
[42] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.
[43] Qingyu Xu, Feng Zhang, Zhiming Yao, Lv Lu, Xiaoyong Du, Dong Deng, and Bingsheng He. 2022. Efficient load-balanced butterfly counting on GPU. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2450–2462.
[44] Wen Xu, Zhetao Li, Haolin Liu, Yunjun Gao, Xiaofei Liao, and Kenli Li. 2024. Differentially private weighted graphs publication under continuous monitoring. *IEEE Transactions on Mobile Computing* (2024).
[45] 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.
[46] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. 2020. LF-GDPR: A framework for estimating graph metrics with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4905–4920.
[47] Xiaowei Ye, Rong-Hua Li, Qiangqiang Dai, Hongchao Qin, and Guoren Wang. 2023. Efficient Biclique Counting in Large Bipartite Graphs. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.
[48] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 731–745.
[49] Yuxuan Zhang, Jianghong Wei, Xiaojian Zhang, Xuexian Hu, and Wenfen Liu. 2018. A two-phase algorithm for generating synthetic graph under local differential privacy. In *Proceedings of the 8th International Conference on Communication and Network Security*. 84–89.
[50] Gengda Zhao, Dong Wen, Xiaoyang Wang, Kai Wang, and Xuemin Lin. 2025. Preserving K-Connectivity in Dynamic Graphs. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 156–169.

Received July 2025; revised October 2025; accepted November 2025