Most Similar Biclique Search at Scale

Deming Chu
University of New South Wale
deming.chu@unsw.edu.au

Zhizhi Gao
Guangzhou University
zhizhigao@e.gzhu.edu.cn

Fan Zhang*
Guangzhou University
zhangf@gzhu.edu.cn

Wenjie Zhang
University of New South Wale
wenjie.zhang@unsw.edu.au

Xuemin Lin
Shanghai Jiao Tong University
xuemin.lin@sjtu.edu.cn

Zhihong Tian
Guangzhou University
tianzhihong@gzhu.edu.cn

ABSTRACT

The biclique is a fundamental model of bipartite cohesive subgraphs. To analyze a bipartite graph, many existing works seek the maximum biclique, that is, the biclique with the largest number of edges. However, our finding is that the *most similar biclique* (i.e., the biclique whose vertices are the most similar to each other) can be a good alternative for understanding the network. Using the model, we can detect meaningful communities with high similarity and avoid unnecessary searches based on vertex similarity. In particular, we aim to find (i) *local most similar biclique*: the biclique that contains a query node q and the similarity between vertices is the highest, and (ii) *global most similar biclique*: the biclique with the highest similarity between vertices.

Despite the NP-hardness of the problems, this paper presents two efficient algorithms, *Mosib* and *Mosib-GloApp*. Specifically, our *Mosib* is an exact algorithm for the most similar biclique search. The algorithm incorporates three novel graph reduction rules that can reduce the size of the bipartite graph while preserving the most similar biclique, as well as two similarity-first search rules that can prioritize the bicliques with high similarity in the search. These techniques can significantly improve the practical efficiency of the algorithm. Meanwhile, our *Mosib-GloApp* is an approximate algorithm that adopts a novel MinHash-based dividing method, and it can further improve the efficiency of the global most similar biclique search. We experimentally evaluate our algorithms on real-world networks, and show that the most similar biclique models can find meaningful results while being computed efficiently.

PVLDB Reference Format:
Deming Chu, Zhizhi Gao, Fan Zhang, Wenjie Zhang, Xuemin Lin, and Zhihong Tian. Most Similar Biclique Search at Scale. PVLDB, 18(4): 1022 - 1034, 2024.
doi:10.14778/3717755.3717763

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/nedchu/mosib-release.

* Fan Zhang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 4 ISSN 2150-8097.
doi:10.14778/3717755.3717763

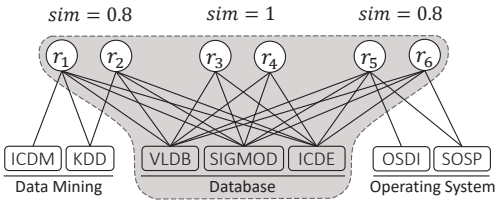


Figure 1: Researcher-Venue Bipartite Network.

1 INTRODUCTION

Bipartite graphs have been widely used to describe the relationships between two classes of entities, e.g., the author-paper relationships in the academia [20], the customer-product relationships in an e-commerce platform [33], and the user-content relationships in a social network [47]. Nowadays, people are producing large-scale bipartite graphs in various fields [11, 29].

The biclique model is a fundamental structure, and it has been applied to applications such as anomaly detection [2, 4, 26], gene expression analysis [23, 31, 44], and social recommendation [22, 26]. Let $G = (L, R, E)$ be a bipartite graph with two disjoint sets of vertices $L(G)$ and $R(G)$, and a set of edges $E(G)$ that connects the two vertex sets. Given G, a *biclique* is a subgraph C of G such that every pair of vertices between the two sides of C is adjacent. The *maximum biclique* [8, 26, 27], i.e., the biclique with the largest number of edges, is often used to analyze a bipartite graph.

Figure 1 presents a researcher-venue bipartite graph that depicts if a researcher has published in a venue. The shadowed subgraph is the maximum biclique, and all six researchers in the biclique have published in database conferences. However, the model fails to distinguish the preferences of these researchers: r_1, r_2 are interested in database and data mining, r_3, r_4 are focusing on database, while r_5, r_6 are interested in database and operating system. In other words, the vertices in a biclique are not necessarily similar, even though they share a set of common neighbors.

To address this limitation of the traditional model, Yao et al. [40] propose the model of *similar biclique*. Given a similarity threshold s, the model regards two vertices u and v as similar if the (Jaccard) similarity between their neighbor sets is no less than s. Yao et al. [40] aim to enumerate any maximal similar biclique where all vertices on one side are similar to each other, and they propose an efficient algorithm *MSBE* for similar biclique enumeration.

The similar biclique model can detect meaningful communities, and the running time is far less than enumerating traditional bicliques [40]. However, the number of resulting similar bicliques

is still excessively large, e.g., there are 4,524 similar bicliques on YouTube (a graph with 293 thousand edges) when $s = 0.5$. Therefore, any practitioner can hardly make sense of these similar bicliques.

This paper is the first to study the problem of *most similar biclique*. Let a r-similar biclique be a biclique where the minimum similarity equals r for every pair of vertices on one side. Intuitively, we aim to find (i) *local most similar biclique*, i.e., the biclique that contains a query node q and the similarity r is the highest, and (ii) *global most similar biclique*, i.e., the biclique with the highest similarity r in the whole network. In addition, we ensure that the resulting most similar biclique is maximal, and that the number of vertices on each side of the biclique is no less than a size constraint τ.

Our model can detect meaning communities from the author-venue graph, either globally or locally. Each community contains a group of researchers that focus on a set of closely related venues. For example, the global most similar biclique ($\tau = 2$) in Figure 1 is $\{r_3, r_4\}\times$\{VLDB, SIGMOD, ICDE\}. Given $\tau = 2$, the local most similar biclique containing r_1 is $\{r_1, r_2\}\times$\{KDD, VLDB, SIGMOD, ICDE\}, while the local most similar biclique containing r_5 is $\{r_5, r_6\} \times$ \{SOSP, VLDB, SIGMOD, ICDE\}. In comparison, the similar biclique model [40] would return {all authors} $\times$ {VLDB, SIGMOD, ICDE} when $s = 0.5$, and it cannot distinguish the differences of authors.

Challenges. Like many other biclique-related problems [26, 35, 40], we can prove that the problem of local/global most similar biclique search is NP-Hard. One may wonder if existing works on similar biclique can solve our problems. As a preliminary attempt, we develop a baseline based on the state-of-the-art method of enumerating similar bicliques, i.e., *MSBE* [40]. However, *MSBE* will return an empty result when the similarity threshold s is larger than the highest similarity. When s is too low, *MSBE* may enumerate a biclique that contains the most similar biclique and other dissimilar vertices. Therefore, the main difficulty is picking the largest r value s.t. the r-similar biclique exists, and set s to this r value. Intuitively, our baseline performs a binary search on r, and uses *MSBE* to verify the existence of a r-similar biclique. Despite its simplicity, the baseline incurs significant computation overheads, e.g., it takes over 24 hours on CiteULike, a graph with 2.3 million edges (see Section 6.2). Thus, this baseline cannot scale to large networks.

One may also wonder if we can solve the problems with the classical techniques of clique and biclique search. For the maximum clique search, existing works [7, 19, 24] use graph coloring and core decomposition to estimate the upper bound of the maximum clique size, and use this upper bound to prune vertices. Existing works of maximum biclique search [26] also use the size to prune the vertices. However, these techniques cannot be used in our studied problem, as our optimization objective (the similarity between vertices) is not necessarily correlated with the size. This motivates us to design new techniques for the most similar biclique search.

Contributions. This paper presents *Mosib*, an exact algorithm for Most similar biclique search. Different from existing works, *Mosib* prunes unpromising vertices and prioritizes promising ones, particularly from the view of vertex similarity. The algorithm incorporates graph reduction rules that aim to reduce the size of the bipartite graph while preserving the local most similar biclique. It also adopts similarity-first search rules that can prioritize the similar bicliques with high similarity in the search. As a result, *Mosib* can significantly speed up the local most similar biclique search (also referred to as local search), while *Mosib* solves the global search with a series of local searches. Compared with the baseline based on Yao et al. [40], our *Mosib* is up to six (resp. five) orders of magnitude faster in the local (resp. global) search. In addition, we compare our model with existing models such as similar biclique [40], (α, β)-core [21], and personalized maximum biclique [35], which verifies the effectiveness of our model (see Section 6.3).

Table 1: Summary of Notations.

Notation	Definition
G	a bipartite graph
n_L; n_R	the number of left-/right-side vertices in G
m	the number of edges in G
τ	the minimum number of vertices on each side of a biclique
N_u	the set of neighbors of a node u
N_u^2	the set of 2-hop neighbors of a node u
$sim(u, v)$	the Jaccard similarity between N_u and N_v

Moreover, we design an approximate algorithm *Mosib-GloApp* for the global search. Unlike the global search for *Mosib*, our *Mosib-GloApp* divides the nodes into groups and only considers the bicliques within each group. This avoids searching for any biclique whose nodes are located in different groups, thus significantly improving the efficiency. We also propose a novel dividing algorithm based on the well-known MinHash technique. The experiments show that our *Mosib-GloApp* can outperform *Mosib* in running time by up to two orders of magnitude.

In summary, our principal contributions are as follows:

1. To our best knowledge, we are the first to formulate the problem of the local/global most similar biclique search. We prove that the problems are NP-Hard.
2. We propose *Mosib*, an exact algorithm for the local (resp. global) most similar biclique search, including novel optimization rules for graph reduction and similarity first search.
3. We propose an approximate algorithm *Mosib-GloApp* for finding the global most similar biclique more efficiently.
4. The experiments on real-world datasets show that our algorithms can efficiently handle large-scale bipartite graphs while finding insightful results in the network.

2 PRELIMINARIES

In this section, we define the problems of finding the local and global most similar bicliques. We prove that the two problems are NP-Hard. Table 1 lists the frequently-used notations.

2.1 Problem Definition

Let $G = (L, R, E)$ be a bipartite graph with two disjoint sets of vertices $L(G)$ and $R(G)$ (i.e., the left-side and right-side vertices), and a set of edges $E(G)$ between the two vertex sets. Given a subgraph C of G, we let $L(C)$ (resp. $R(C)$) denote the left-side (resp. right-side) vertices from C. Then, a *biclique* is defined as a subgraph C of G such that every vertex pair between the two sides of C are connected, that is, $\forall(u \in L(C))(v \in R(C))\ (u, v) \in E$.

The *similar biclique* is a kind of biclique whose left-side vertices are similar to each other [40]. Let $N(u) = \{v \mid (u, v) \in E\}$ be

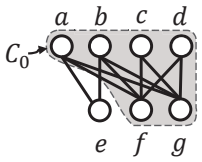


Figure 2: Maximal Biclique.

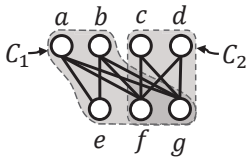


Figure 3: Similar Biclique.

the *neighbor set* of u. The similarity between u and v, denoted by $sim(u, v)$, is defined as the Jaccard similarity between the neighbor sets, i.e., $sim(u, v) = J(N(u), N(v)) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$.

Definition 1 (r-similar biclique)**.** A r-similar biclique C is a biclique such that the minimum similarity between every pair of left-side vertices of C is equal to r, i.e., $r = sim(C) = \min_{u,v \in L(C)} sim(u, v)$.

A r-similar biclique is *maximal* when it is not a subset of any other r-similar biclique. We assume that the similarity constraint is considered for the left-side, otherwise, we can simply swap the two sides before applying the algorithm.

In this paper, we aim to find (i) *global most similar biclique*, i.e., the maximal r-similar biclique with the highest similarity r in the whole graph, and (ii) *local most similar biclique*, i.e., the maximal r-similar biclique that contains q and the similarity r is the highest. For simplicity, we also refer to our studied problems as the local (resp. global) search problem, and they can be formally defined as:

Problem 1 (Local Most Similar Biclique)**.** Given a graph G, a size threshold τ, and a query node $q \in L(G)$, we ask for a r^*-similar biclique $C^*_{\tau,q}$ that satisfies

(i) **Node Containment:** $q \in C^*_{\tau,q}$;
(ii) **Size Constraint:** $|L(C^*_{\tau,q})| \geq \tau \wedge |R(C^*_{\tau,q})| \geq \tau$;
(iii) **Maximality:** $C^*_{\tau,q}$ is maximal;
(iv) **Most Similar:** for any r-similar biclique satisfying (i-iii), $r^* \geq r$.

Problem 2 (Global Most Similar Biclique)**.** Given a graph G and a size threshold τ, we ask for a r^*_g-similar biclique C^*_τ that satisfies

(i) **Size Constraint:** $|L(C^*_\tau)| \geq \tau \wedge |R(C^*_\tau)| \geq \tau$;
(ii) **Maximality:** C^*_τ is maximal;
(iii) **Most Similar:** for any r-similar biclique satisfying (i-ii), $r^*_g \geq r$.

There is an "at least τ" size constraint for the resulting biclique C, i.e., $|L(C)| \geq \tau$ and $|R(C)| \geq \tau$. Such a size constraint can avoid too small or too skewed bicliques, and is widely adopted in previous works [26, 35, 40]. If necessary, the techniques in this paper can be easily extended to handle different size constraints on two sides. In the real world, the size constraint τ can refer to the number of customers and products in an e-commerce platform, or the number of users and tweets in a bipartite social network.

The rationale of the "at least τ constraint" is as follows. Let $\tau = 5$. In the worst case, we will obtain a $(5, 5)$-biclique which is not too small or skewed. The constraint can also stop our algorithms from finding a $(1, 5)$-biclique or a $(1, 1)$-biclique. Therefore, the "at least τ" constraint can keep good bicliques and prune bad ones.

Example 1. Consider the maximal biclique C_0 in Figure 2, and the similar bicliques C_1 and C_2 in Figure 3. The nodes a and b have the same neighbor set (i.e., $\{e, f, g\}$), while nodes c and d have the same neighbor set $\{f, g\}$. The similar biclique can observe the difference between $\{a, b\}$ and $\{c, d\}$, but the raw biclique model cannot. Let $\tau = 2$. Given $q \in \{a, b\}$ (resp. $q \in \{c, d\}$), the local most similar biclique is C_1 (resp. C_2), because C_1 and C_2 are both 1.0-similar biclique. Both C_1 and C_2 are the global most similar biclique.

2.2 Problem Hardness

Theorem 1. *The global most similar biclique search is NP-Hard.*

Proof. The maximum balanced biclique (MBB) problem asks for a biclique C such that $|L(C)| = |R(C)|$ and the number of edges in C is the largest. This problem is NP-hard [17]. Assume that the global search can be solved in polynomial time. Then, we can solve the MBB problem as follows. For every τ from 1 to $\min\{|L(G)|, |R(G)|\}$, there exists a biclique C with $|L(C)| \geq \tau$ and $|R(C)| \geq \tau$ if and only if we run the global search with size constraint τ and it returns a non-empty result. Thus, we can determine the MBB in polynomial time. This contradicts the fact that the MBB problem is NP-hard. □

Theorem 2. *The local most similar biclique search is NP-Hard.*

Proof. Assume that the local search problem is solvable in polynomial time. For every any $v \in L(G)$, we compute the local most similar biclique C^*_v containing v. Then, we select from these local most similar bicliques the one with the largest similarity r, and return it as the final result. This is a polynomial-time solution to the global search problem, which contradicts Theorem 1. □

3 THE BASELINE SOLUTION

In what follows, we first present an overview of Yao et al. [40], the state-of-the-art method for enumerating similar bicliques. After that, we devise a baseline solution based on Yao et al's method.

3.1 Yao et al.'s Method and Our Insights

Let s be a similarity threshold. Yao et al. [40] aim to enumerate any r-similar biclique with $r \geq s$, and they present an algorithm (referred to as *MSBE*) for similar biclique enumeration. Formally speaking, given G, a size threshold τ, and a similarity threshold $s \in [0, 1]$, *MSBE* can enumerate all r-similar bicliques C such that (i) $r \geq s$, (ii) $|L(C)| \geq \tau \wedge |R(C)| \geq \tau$, and (iii) C is maximal.

Suppose we know the similarity of global most similar biclique C^*_τ (i.e., r^*_g in Problem 2). Then, we can answer the global most similar biclique by running *MSBE* with a similarity threshold $s = r^*_g$ and a size threshold τ. This will lead to the lowest computation cost. However, it is impossible to know r^*_g in advance.

On the one hand, if we adopt a large s, *MSBE* may return an empty result. On the other hand, if we use a small s (e.g., $s = 0$), *MSBE* may enumerate a biclique that contains the most similar biclique and other dissimilar vertices. Such a small s also incurs prohibitive overheads, e.g., when $s = 0$, *MSBE* cannot terminate in 24 hours on GitHub (a graph with 0.5 million edges). Therefore, the main difficulty is picking the largest r value s.t. the r-similar biclique exists, and set s to this r value.

3.2 Our Baseline Based on Yao et al's Method

Basically, our baseline *MSBE*$^+$ uses a binary search to select a proper s for *MSBE*. Algorithm 1 presents the pseudo-code of our baseline

Algorithm 1: *MSBE*$^+$*: Global*(G, τ)

```
1 Initialize the result C*_τ = ∅;
2 Initialize the binary-search boundaries s_l = 0 and s_u = 1;
3 while |s_u − s_l| is not sufficiently small do
4     Let s_m = (s_l + s_u)/2;
5     Use MSBE to find a maximal r-similar biclique C with
      r ≥ s_m and satisfying the size constraint τ;
6     if C is found in above then
7         Set s_l = r; Update C*_τ = C;
8     else Set s_u = s_m;
9 return C*_τ;
```

MSBE$^+$ when it handles global search. The algorithm starts with the binary-search boundaries $s_l = 0$ and $s_u = 1$, and then iteratively tests $s_m = (s_l + s_u)/2$ and updates the boundaries. Given s_m, it runs *MSBE* with similar threshold $s = s_m$ and size threshold τ, and immediately terminates the enumeration once *MSBE* finds any qualified r-similar biclique C (Line 5). If C is found, then the algorithm sets $s_l = r$ and updates the result $C^*_\tau = C$ (Line 7), otherwise, it sets $s_u = s_m$ (Line 8). Finally, C^*_τ is returned as the answer (Line 9).

Algorithm 1 can be extended to handle the local search. Note that *MSBE* enumerates all similar bicliques with a for-loop on $u \in L(G)$ (Lines 3-10 of Algorithm 1 in [40]), each iteration of which enumerates the similar bicliques containing u. In order to find the local most similar biclique containing a node q, we can simply replace the above-mentioned for-loop in *MSBE* with its inner body $(u = q)$ when we run Algorithm 1.

Let b be the number of iterations in the binary search, and n_L be $|L(G)|$, and m be $|E(G)|$. The baseline *MSBE*$^+$ takes $O(bm \cdot 2^{n_L})$ time for the local search and $O(bmn_L \cdot 2^{n_L})$ time for the global search. In particular, our baseline *MSBE*$^+$ runs the enumeration algorithm *MSBE* for b times, each of which takes $O(m \cdot 2^{n_L})$ time for local search and $O(mn_L \cdot 2^{n_L})$ time for global search [40].

Although the baseline *MSBE*$^+$ is conceptually simple, it incurs significant overheads, e.g., it requires 39 hours for the global search on CiteULike, a graph with 2.3 million edges (see Section 6.2).

4 PROPOSED EXACT SOLUTION

The *Bron-Kerbosch* algorithm [6] is an algorithm for listing all maximal cliques in an undirected graph. The algorithm attempts to add a vertex to a partial clique and then remove it to find more cliques. Such a branch-and-bound framework is adopted by almost all existing works of clique and biclique [1, 26, 35, 40, 44], as well as our solutions. Nevertheless, the framework only provides an enumeration method, and must be redesigned to fit our problem.

This section presents *Mosib*, an exact algorithm that borrows ideas from the *Bron-Kerbosch* algorithm [6] but significantly improves the efficiency with a novel algorithm design. At a high level, *Mosib* incorporates two techniques in its local search:

1. **Graph Reduction Rules.** This technique aims to reduce the size of the bipartite graph while preserving the local most similar biclique. It includes three graph reduction rules, namely, hop-based, degree-based, and similarity-based rules.
2. **Similarity-First Search.** This technique contains two similarity-first search rules that aim to prioritize the bicliques with high similarity in the search, such that more vertices can be pruned by the similarity-based graph reduction rule.

In general, our algorithm maintains a partial biclique $(L, R, L \times R)$ and recursively adds left-side vertices into L. Given a fixed set L, the set R is simply the common neighbors of all vertices in L, i.e., $R = \bigcap_{u \in L} N_u$. When we add vertices into L, we consider a vertex u as promising if it is similar to the vertices in L. Our *Mosib* uses the graph reduction rules to prune unpromising vertices and uses the similarity-first search to prioritize promising vertices.

Next, we first detail the graph reduction rules and similarity-first search. Then, we integrate these techniques into our algorithms.

4.1 Graph Reduction Rules

Given G and a subgraph G' of G, we can reduce G to G' if and only if G' preserves the local most similar biclique of G.

Definition 2 (LMSB-Preserved Subgraph)**.** Let G be a graph, τ be a size threshold, and $q \in L(G)$ be a query node. A subgraph G' of G is the LMSB-preserved subgraph, denoted by $G' \overset{\tau,q}{\equiv} G$, if the local most similar biclique on G and G' is the same given τ and q, i.e., $C^*_{\tau,q}(G) = C^*_{\tau,q}(G')$.

Lemma 1 (Transitivity of LMSB-Preserved Subgraph).

$$G_1 \overset{\tau,q}{\equiv} G_2 \wedge G_2 \overset{\tau,q}{\equiv} G_3 \implies G_1 \overset{\tau,q}{\equiv} G_3.$$

Proof. Let the local most similar biclique of G_1 be $C^*_{\tau,q}(G_1) = C$. Given $G_1 \overset{\tau,q}{\equiv} G_2 \wedge G_2 \overset{\tau,q}{\equiv} G_3$, it follows that $C^*_{\tau,q}(G_3) = C^*_{\tau,q}(G_2) = C^*_{\tau,q}(G_1) = C$. In addition, G_3 must be a subgraph G_1, as G_3 is a subgraph of G_2 and G_2 is a subgraph of G_1. Based on the above, G_3 is the LMSB-preserved subgraph of G_1. □

By the transitive property, we can repeatedly reduce a graph to its LMSB-preserved subgraph. We eliminate a node u from G by removing u and all its incident edges, denoted as $G \ominus u$.

Hop-based Rule (*HOP*). Given a node q, the first rule states that we can eliminate any node u whose distance to q is larger than 2, i.e., eliminating all nodes outside q's 2-hop neighborhood. Then, we can formalize the hop-based rule as follows:

Lemma 2 (Rule *HOP*). *Let q be a query node, and N^2_q be the set of 2-hop neighbors of a node q, then we have*

$$\forall u : u \notin N^2_q \implies G \ominus u \overset{\tau,q}{\equiv} G.$$

Proof. Let u be a node such that $u \notin N^2_q$. Proving $G \ominus u \overset{\tau,q}{\equiv} G$ is equivalent to proving u is not contained in $C^*_{\tau,q}(G)$, i.e., the local most similar biclique on G given τ and q. We prove this by contradiction. Assume that u is contained in $C^*_{\tau,q}(G)$. Recall that a biclique is a complete bipartite graph, so the distance between u and q must be no more than 2. This contradicts the fact that $u \notin N^2_q$. □

Degree-based Rule (*DEG*). Given the size threshold τ, the second rule recursively eliminates any node whose degree is less than τ until no more nodes can be removed.

Algorithm 2: graph reduction rules

```
def HOP(G_r, q):
    for ∀u : u ∉ N_q^2 do G_r ← G_r ⊖ u;
def DEG(G_r, τ):
    while ∃u : d(u, G_r) < τ do G_r ← G_r ⊖ u;
def SIM(G_r, q, r*):
    for u ∈ L(G_r) and sim(u, q) < r* do G_r ← G_r ⊖ u;
```

Lemma 3 (Rule *DEG*). *Let $d(u, G)$ be degree of u in graph G, then*

$$\forall u:\ d(u,G) < \tau \implies G \ominus u \overset{\tau,q}{\equiv} G.$$

Proof. Let u be a node such that $u \in L(G)$ and $sim(u, q) < r^*$. Proving $G \ominus u \overset{\tau,q}{\equiv} G$ is equivalent to proving u is not contained in $C^*_{\tau,q}(G)$, i.e., the local most similar biclique on G given τ and q. We prove this by contradiction. Assume that u is contained in $C^*_{\tau,q}(G)$. By the fact that $d(u, G) < \tau$ and $C^*_{\tau,q}(G)$ is a subgraph of G, we have $d(u, C^*_{\tau,q}(G)) < \tau$. But this contradicts the size constraint of the local most similar biclique, i.e., $d(u, C^*_{\tau,q}(G)) \geq \tau$. □

Similarity-based Rule (*SIM*). Let a r^*-similar biclique $C^*_{\tau,q}$ be the temporary result of the local most similar biclique that satisfies the conditions (i)-(iii) in Problem 1. If a left-side node $u \in L(G)$ is not sufficiently similar to q (i.e., $sim(u, q) < r^*$), then we can eliminate u from G, as any biclique containing $\{u, q\}$ is not the highest in similarity. This rule is referred to as the similarity-based rule (*SIM*):

Lemma 4 (Rule *SIM*). *Let q be a query node, and a r^*-similar biclique $C^*_{\tau,q}$ be the temporary result of the local most similar biclique. Then, for every update of r^* and $C^*_{\tau,q}$,*

$$\forall u \in L(G):\ sim(u,q) < r^* \implies G \ominus u \overset{\tau,q}{\equiv} G.$$

Proof. Let u be a node such that $u \in L(G)$ and $sim(u, q) < r^*$. Proving $G \ominus u \overset{\tau,q}{\equiv} G$ is equivalent to proving u is not contained in $C^*_{\tau,q}(G)$, i.e., the local most similar biclique on G given τ and q. We prove this by contradiction. Assume that u is contained in $C^*_{\tau,q}(G)$. By the fact that $C^*_{\tau,q}(G)$ contains u and q, we have $sim(C^*_{\tau,q}(G)) \leq sim(u, q) < r^* = sim(C^*_{\tau,q})$. Therefore, $C^*_{\tau,q}$ has a higher similarity than $C^*_{\tau,q}(G)$, which contradicts the fact that $C^*_{\tau,q}(G)$ is the local most similar biclique on G. □

Implementation. Algorithm 2 presents the pseudo-code of our graph reduction rules. Let G_r be a reduced graph. The algorithm starts with $G_r = G$, and then repeatedly reduces G_r if necessary. Given a size constraint τ, we implement *DEG* with a BFS-like method, that is, we remove any vertex u with $d(u, G_r) < \tau$ and recursively remove any neighbor v of u if the degree of v drops below τ. For *SIM*, we compute $sim(u, v)$ and memorize the result with a table. The running time of all these rules is $O(|E(G_r)|)$ time.

Note that *DEG* can interact with other rules, i.e., using *SIM* or *HOP* may cause degree reduction, creating a chance for further using *DEG*. The rules are applied on demand: we apply all rules *HOP, SIM, DEG* in the initialization, and then apply *SIM, DEG* once we find a biclique with a higher similarity than the current best (see Algorithm 3 Lines 2 and 11). Such an on-demand application is more efficient than other ways, e.g., sequential or fixed point.

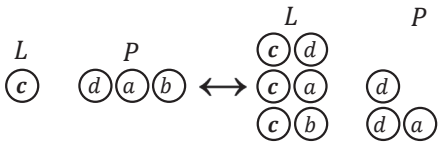


Figure 4: The Similarity-First Search on Figure 2.

4.2 Similarity-First Search

In a nutshell, our *Mosib* enumerates similar bicliques in a similarity-first search manner, i.e., it prioritizes the biclique with high similarity. As a result, the algorithm can prune more vertices with the similarity-based rule (see the last section) in the early stage, leading to a significantly improved performance.

Recall that the algorithm maintains a partial biclique $C = (L, R, L \times R)$ and recursively adds left-side vertices into L, where R can be simply computed as $R = \bigcap_{u \in L} N_u$. Let P be the candidate left-side vertices that we consider adding to L. In this process, we adopt the following two similarity-first search rules.

Similar Node First Rule (*SFS*). The first rule is to sort all candidate vertices in P in decreasing order of $sim(u, L)$, where $sim(u, L)$ is the minimum similarity between u and the vertices from L, i.e., $sim(u, L) = \min_{v \in L} sim(u, v)$. Every time a node is moved from P to L, the algorithm uses the rule to order the updated P.

The rationale of the rule is as follows. Let $sim(L)$ be the minimum pair-wise similarity in L, i.e., $sim(L) = \min_{u,v \in L} sim(u, v)$. If a vertex u is added to L, the similarity becomes $sim(L \cup \{u\}) = \min\{sim(L), sim(u, L)\}$. By sorting P in decreasing $sim(u, L)$, the algorithm can prioritize the node u that maximizes the similarity of the partial biclique after it adds u into L.

High-Similarity Node Sets First Rule (*SFS2*). Given a query node q, a naive approach is to start with $L = \{q\}$ and the set of vertices P that can be added to L. We have to consider all vertices in P to ensure algorithm correctness. However, some vertices in P are relatively dissimilar to q and may delay the search. Intuitively, the second rule aims to divide the naive search into several iterations of searches such that (i) dissimilar vertices are not considered in the early iterations, and (ii) the computation cost stays the same.

In particular, we assume that $P = \{v_1, v_2, \cdots, v_{|P|}\}$ is ordered by the similarity to q, i.e., $sim(v_1, q) \geq sim(v_2, q) \geq \cdots sim(v_{|P|}, q)$. The algorithm runs up to $|P|$ iterations, the k-th of which executes a local search with $L' = \{q, v_k\}$ and $P' = \{v_1, v_2, \cdots, v_{k-1}\}$. In other words, for every biclique C we detected in the k-th iteration, we ensure that any vertex in $L(C)$ is top-k most similar to q, i.e., $L(C) \subseteq \{q, v_1, v_2, \cdots, v_k\}$, while the searches on relatively dissimilar vertices are deferred to the subsequent iterations.

Moreover, the rule will not increase the computation cost. All bicliques enumerated in the k-th iteration are different from those enumerated in the previous iterations, because the former always contains v_k while the latter never contains v_k. Therefore, the algorithm enumerates every biclique exactly once and has the same worst-case time complexity as the naive approach.

Example 2. Let node c in Figure 2 be the query node. In Figure 4 (left), the similar node first rule (*SFS*) sorts the candidate vertices

Algorithm 3: *Mosib-Exact: Local*(G, q, τ)

1 Initialize the temporary local result $r^* = -\infty$ and $C^*_{\tau,q} = \emptyset$;
2 Copy G into G_r; Reduce G_r with *HOP* and *DEG*;
3 Sort $u \in L(G_r)$ in decreasing order of $sim(q, u)$; // *SFS2* ↓
4 **for** $u \in L(G_r)$ **and** $u \in G_r$ **do**
5 Let P be the set of u (Line 14) visited in previous iterations;
6 Run *Enum-BK*$(\{q, u\}, N_q \cap N_u, P \cap N^2_u, \emptyset, s(q, u))$;
7 **return** $\langle r^*, C^*_{\tau,q} \rangle$;
8 **def** *Enum* (L, R, P, X, r) **:**
9 **if** $|L| \geq \tau$ **and** $|R| \geq \tau$ **and** $r > r^*$ **then**
10 Update $C^*_{\tau,q} = (L, R, L \times R)$ and $r^* = r$;
11 Reduce G_r with *SIM* and *DEG*;
12 Let $sim(u, L)$ be $\min_{v \in L} sim(u, v)$;
13 Sort $u \in P$ in decreasing order of $sim(u, L)$; // *SFS*
14 **for** $u \in P$ **and** $u \in G_r$ **do**
15 $L' = L \cup \{u\}$; $R' = R \cap N_u$; $P' = P \cap N^2_u$; $X' = X \cap N^2_u$;
16 $r' = \min\{r, sim(u, L)\}$;
17 **if** $r' > r^*$ **and** $|L'| + |P'| \geq \tau$ **and** $|R'| \geq \tau$ **and** $\nexists\, v \in X'$ s.t. $R' \subseteq N_v$ **then**
18 *Enum* (L', R', P', X', r');
19 $P = P \setminus \{u\}$; $X = X \cup \{u\}$;

$P = \{d, a, b\}$ by similarity to $L = \{c\}$. Then, we will attempt to move candidate vertices from P to L using the order.

By the high-similarity node sets first rule (*SFS2*), Figure 4 will turn the naive enumeration (left) into three iterations of enumeration (right). The computation cost is the same, but the algorithm does not need to consider b in the first two iterations. Similarly, when the algorithm attempts to find the most similar biclique among the 10% vertices that are most similar to the query node, it can avoid considering the remaining 90% vertices, thus prioritizing the biclique with high similarity and reducing the running time.

4.3 Exact Algorithm for Local Search

Algorithm 3 presents our exact solution for local search. The algorithm incorporates the graph reduction rules (see Section 4.1) and the similarity-first search (see Section 4.2). The parameters of graph reduction rules are omitted, as they are consistent with the definitions in Algorithm 2, e.g., *HOP* is equivalent to *HOP*(G_r, q).

Lines 1-7 are the main body of Algorithm 3. Let a r^*-similar biclique $C^*_{\tau,q}$ be the temporary result of the local most similar biclique. The algorithm starts with $r^* = -\infty$ and $C^*_{\tau,q} = \emptyset$, and initializes a reduced graph G_r with rules *HOP* and *DEG* (Lines 1-2). After that, the algorithm uses rule *SFS2* to enumerate all similar bicliques that contain q (Lines 3-6). That is, for each vertex $u \in L(G_r)$ in decreasing order of the similarity to q, we execute a local search with $L = \{q, u\}$ and P is the set of u visited in the previous iterations.

Lines 8-19 depict the similar biclique enumeration of *Mosib*. The algorithm maintains a partial biclique $C = (L, R, L \times R)$, and recursively adds the candidate vertices from P to L. The function *Enum* requires five parameters (Line 8). In particular, L, R are the left- and right-side vertices of the partial biclique, P is the set of candidate vertices, X is a set of vertices used to check whether C is maximal, and $r = sim(C)$ is the similarity of C.

The computation of *Enum* works as follows. Every time the algorithm detects a maximal r-similar biclique C with $r > r^*$ and the size constraints satisfied, it updates $\langle r^*, C^*_{\tau,q} \rangle$ and reduces G_r with *SIM* and *DEG* (Lines 9-11). Note that we apply *SIM* first and then *DEG*, as the degrees of some nodes may be decreased after applying *SIM*. Then, we expand L to enumerate more bicliques (Lines 12-19). In particular, we sort P with *SFS* (Lines 12-13). For every vertex $u \in P$, we either move u into L and recursively expand the partial biclique (Line 18), or move u into X (Line 19). The expansion of u is valid if and only if (i) the similarity is large enough, i.e., $r' > r^*$; (ii) the sizes $|L'| \cup |P'|$ and $|R'|$ are no less than τ; and (iii) the partial biclique is maximal, i.e., $\nexists\, v \in X'$ s.t. $R' \subseteq N_v$.

Efficient Similarity Computation. The algorithm repeatedly computes the similarity between different pairs of vertices. To speed up the similarity computation, we store the result of $sim(u, v)$ into a table when we apply any rule related to similarity, that is, *SIM*, *SFS*, and *SFS2*. In addition, when we compute $sim(u, L)$ in (Line 12), we reuse the similarity results from the parent function. That is, for any $v \in P$, we store $sim(v, L)$ into a hash map and send this map to any child function (Line 18). Suppose we add a new vertex u into L when we call this child function, we can efficiently compute the updated similarity, i.e., $sim(v, L \cup \{u\}) = \min\{sim(v, L), sim(v, u)\}$. The base case is the computation of $sim(u, q)$ (Line 3).

Theorem 3. *Algorithm 3 takes* $O\left((m + n_L \log n_L) \cdot 2^{n_L}\right)$ *time.*

Proof. First, we analyze the running time of *Enum*. In particular, Lines 9-11 take $O(m)$ time as the graph reduction rule takes linear time (see Section 4.1). By the efficient similarity computation above, Lines 12-13 require $O(\sum_{u \in P} |N_u|) \subseteq O(m)$ time to update the similarity. Lines 12-13 also require $O(|P| \log |P|)$ time to sort the vertices. Lines 14-19 (excluding Line 8) take a running time of $O(m)$, as the cost of Line 15 is bounded by the two-hop neighborhood of P. Therefore, *Enum* (excluding Line 8) requires $O(m + |P| \log |P|)$ time. Each time we run *Enum*, it either moves a node from P to L or excludes it from L (by adding it to X), forming a recursive tree with $2^{|P|}$ leaves. By multiplying $2^{|P|}$ and $O(m + |P| \log |P|)$, the time complexity of *Enum* equals $O\left((m + |P| \log |P|) \cdot 2^{|P|}\right)$.

Next, we analyze Algorithm 3. Lines 1-3 need $O(m)$ time. After that, the algorithm runs in $|L(G_r)|$ iterations, the k-th of which runs *Enum* with $|P| = k$ and it takes $O\left((m + |k| \log |k|) \cdot 2^{|k|}\right)$ time. Then, the time complexity of Lines 4-6 equals

$$\sum_{k=1}^{|L(G_r)|} (m + |k| \log |k|) \cdot 2^{|k|} \;\leq\; (m + n_L \log n_L) \cdot 2^{n_L}. \qquad (1)$$

Observe that the time complexity of Algorithm 3 is bounded by the r.h.s. of Equation 1. Thus, the theorem is proved. □

On the theory side, the algorithm has an exponential time complexity. Due to the NP-hardness of the studied problems, it is impossible to design a polynomial time optimal solution unless P=NP. If time is of the essence, our algorithm can return the answer when the running time reaches a pre-set time threshold. On the practice side, the algorithm can outperform the baseline *MSBE*$^+$ in running

Algorithm 4: *Mosib-Exact: Global* (G, τ)

1 Initialize the temporary global result $r_g^* = -\infty$ and $C_\tau^* = \emptyset$;
2 **for** $q \in L(G)$ **do**
3 Initialize the temporary local result $r^* = r_g^*$ and $C_{\tau,q}^* = C_\tau^*$;
4 Copy G into G_r; Reduce G_r with *HOP*, *SIM*, and *DEG*;
5 Run Lines 3-7 of Algorithm 3 to update the local result;
6 **if** $r^* > r_g^*$ **then** Update $r_g^* = r^*$ and $C_\tau^* = C_{\tau,q}^*$;
7 **return** $\langle r_g^*, C_\tau^* \rangle$;

time by up to six orders of magnitude (and at least 2,955x), according to our experiments in Section 6.2.

Solution Uniqueness. Given a query node q, there may be multiple local most similar bicliques with the same similarity. To ensure solution uniqueness, Algorithm 3 by default returns the first detected biclique that satisfies the constraint.

However, a user may hope to obtain the most similar biclique with the largest size (or any user-preferred property), rather than the first detected one. We achieve this goal as follows. Given a user-preferred property f and a community $C = (L, R, L \times R)$, we can rewrite the if condition in Algorithm 3 Line 9 to

$$|L| \geq \tau \textbf{ and } |R| \geq \tau \textbf{ and } (r > r^* \textbf{ or } (r = r^* \textbf{ and } f(C) > f(C_{\tau,q}^*))).$$

Besides, we can ensure the solution uniqueness of the global most similar biclique with a similar method. That is, when we run Algorithm 4 Line 5 to call the local search (Algorithm 3), we rewrite the if condition in Algorithm 3 like what we do above.

4.4 Exact Algorithm for Global Search

Let C_τ^* be the global most similar biclique. Given any query node $q \in L(C_\tau^*)$, it follows that C_τ^* is also a local most similar biclique that contains q. As a result, we can obtain the global most similar biclique using a series of local searches.

Algorithm 4 presents our exact solution for global search. Let a r_g^*-similar biclique C_τ^* be the temporary result of the global most similar biclique. The algorithm first initializes the global result $r_g^*,\ C_\tau^*$ (Line 1). Then, for any $q \in L(G_{gr})$, it finds the local most similar biclique $C_{\tau,q}^*$ that contains q (Lines 3-5) and updates $r_g^*,\ C_\tau^*$ when the similarity of $C_{\tau,q}^*$ is higher than C_τ^* (Line 6). Lines 3-5 are almost the same to Algorithm 3, except that we set $r^* = r_g^*$ and reduce G_r with rule *SIM* in the initialization. Note that any r-similar biclique with $r < r_g^*$ is not the global most similar biclique. After that, the global result $\langle r_g^*,\ C_\tau^* \rangle$ is returned as the final answer.

Theorem 4. *Algorithm 4 takes* $O((m + n_L \log n_L) \cdot n_L \cdot 2^{n_L})$ *time.*

Proof. Lines 2-6 run the local search for each left-side vertex q. Therefore, Algorithm 4 is equivalent to running Algorithm 3 for up to n_L times, leading to the time complexity in the theorem. □

5 PROPOSED APPROXIMATE SOLUTION FOR GLOBAL SEARCH

The global search of *Mosib* may take tens of hours on large graphs in our preliminary test. This section proposes *Mosib-GloApp*, an approximate algorithm for improving the practical performance of global search. At a high level, *Mosib-GloApp* contains two phases:

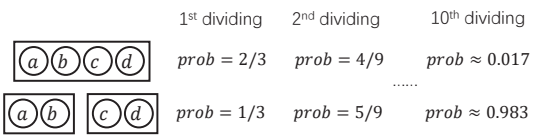


Figure 5: Example of *Mosib-GloApp* on Figure 2.

Algorithm 5: Dividing Step of *Mosib-GloApp*

1 Initialize h hash functions $f^{(1)}, \cdots, f^{(h)}$;
2 **for** $i = 1$ to h **do**
3 **for** each $u \in L(G)$ **do** $f_{\min}^{(i)}(u) = \min\{f^{(i)}(v) : v \in N_u\}$;
4 Divide the nodes in $L(G)$ into groups with $f_{\min}^{(1)}(u)$;
5 Recursively divide any group with $f_{\min}^{(2)}(u), \cdots, f_{\min}^{(h)}(u)$, until the size of each group is no more than M;
6 Let $S^{(1)}, \cdots, S^{(d)}$ be is the groups after recursive division;
7 **return** $S^{(1)}, \cdots, S^{(d)}$;

1. **Dividing**: This phase divides the left-side nodes into groups $S^{(1)}, \cdots, S^{(d)}$ such that the nodes in each group are similar to each other and the size of each group is not large.
2. **Biclique Search**: This phase looks for the global most similar biclique among the groups. For each group $S^{(i)}$, the algorithm enumerates the similar biclique within the group based on the local search algorithm of *Mosib*.

Our *Mosib-GloApp* is similar to the global search of *Mosib*, except that it divides nodes into groups and only considers the similar bicliques within each group. This avoids the search on any biclique whose nodes are located in different groups, thus significantly improving the efficiency of global search. Meanwhile, it is non-trivial to properly divide nodes such that we can retain the global most similar biclique in the groups and prune as many unnecessary bicliques as possible. To address this challenge, we design a novel dividing algorithm based on the well-known MinHash technique.

Example 3. Figure 5 demonstrates the idea of our *Mosib-GloApp*, using the graph in Figure 2. The algorithm uses MinHash to divide vertices with highly similar sets of neighbors into a group. After ten iterations of dividing, we will obtain two groups $\{a, b\}$ and $\{c, d\}$ with high probability. The biclique search is conducted on each group, i.e., the search cost can be largely reduced. As a result, our *Mosib-GloApp* can quickly detect the global most similar biclique when its similarity is close to 1 (this happens frequently in practice). When *Mosib-GloApp* returns a biclique with a similarity below 1, we recommend using the exact solution instead.

In what follows, we first present the dividing phase of *Mosib-GloApp*, then elaborate on the biclique search phase.

5.1 Dividing Phase of the Algorithm

The dividing phase of our *Mosib-GloApp* aims to divide the nodes into a set of group $S^{(1)}, \cdots, S^{(d)}$ such that the maximum group size is no larger than a constant M (spec., $M = 100$) and the nodes in each group are similar to each other. To explain how our dividing method works, we first introduce the MinHash technique:

Algorithm 6: *Mosib-GloApp* (G, τ)

1 Divide the nodes in $L(G)$ into $S^{(1)}, \cdots, S^{(d)}$; // Alg. 5
2 Initialize the temporary global result $r_g^* = -\infty$ and $C_\tau^* = \emptyset$;
3 **for** each $S^{(i)} \in \{S^{(1)}, \cdots, S^{(d)}\}$ **do**
4 Copy G into $G^{(i)}$; Remove from $G^{(i)}$ any $u \in L(G) \setminus S^{(i)}$;
5 **for** each $q \in L\left(S^{(i)}\right)$ **do**
6 Initialize local result $r^* = r_g^*$ and $C_{\tau,q}^* = C_\tau^*$;
7 Copy $G^{(i)}$ into G_r;
8 Reduce G_r with *HOP*, *SIM*, and *DEG*;
9 Run Lines 3-7 of Algorithm 3 to update the local result;
10 **if** $r^* > r_g^*$ **then** Update $r_g^* = r^*$ and $C_\tau^* = C_{\tau,q}^*$;
11 **return** $\langle r_g^*, C_\tau^* \rangle$;

The MinHash [5] is a well-known technique for estimating the Jaccard similarity between two sets. Given a hash function $f(\cdot)$ that maps a node to an integer, the MinHash of u is defined as the minimum hash value of u's neighbors, i.e., $f_{\min}(u) = \min_{v \in N_u}\{f(v)\}$. Then, the probability that two vertices u and v have an identical MinHash value is equal to the Jaccard similarity between the neighbor sets of u and v, that is, $\Pr\{f_{\min}(u) = f_{\min}(v)\} = Jaccard(N_u, N_v)$.

By the definition of the MinHash, the vertices with the same MinHash value are likely to have a high Jaccard similarity in their neighbor sets. Intuitively, our *Mosib-GloApp* divides the nodes into groups with an identical MinHash value, and recursively divides any group whose size is larger than M.

Algorithm 5 presents the dividing phase of our *Mosib-GloApp*. The algorithm first generates h hash functions, each is a permutation of the right-side nodes (Line 1). Then, we group any node $u \in L(G)$ by the MinHash value $f_{\min}^{(1)}(u)$. For those groups whose size is larger than a constant M, we recursively divide the groups using the MinHash $f_{\min}^{(2)}(\cdot), \cdots, f_{\min}^{(h)}(\cdot)$, until the size of each group is below M. In practice, we set $M = 100$ and $h = 10$.

Algorithm 5 takes $O(h \cdot |E|)$ time due to the initialization of MinHash values. In addition, we can establish the probability that Algorithm can retain a r-similar biclique C:

Lemma 5. *Let C be a r-similar biclique. Algorithm 5 can divide $L(C)$ into the same group with at least $r^{|L(C)|-1}$ probability.*

Proof. Let $k = |L(C)|$ and $v_1, v_2, \cdots, v_k$ be the vertices in $L(C)$. By the definition of r-similar biclique, we have $sim(v_1, v_i) \geq r$ for any i from 2 to k. Therefore, the probability that $v_1, v_2, \cdots, v_k$ have the same MinHash value (i.e., they are divided into the same group by Algorithm 5) is equal to $r^{|L(C)|-1}$. □

In other words, Algorithm 5 can retain a r-similar biclique C when its similarity r is equal to (or close to) 1.

5.2 Algorithm Overview and Biclique Search

Algorithm 6 depicts the pseudo-code of *Mosib-GloApp*. Given G, the algorithm first divides the left-side vertices into groups (discussed in the last section), and then finds the bicliques in the groups (Lines 3-10). The algorithm initializes the global result r_g^*, C_τ^* (Line 2). Then, for any group $S^{(i)}$, it removes any left-side vertices outside the group and stores the remaining graph into $G^{(i)}$ (Line 4). After that, the algorithm finds the local most similar biclique $C_{\tau,q}^*$ that contains q for any $q \in L(G^{(i)})$ (Lines 6-9), and updates the global result when the similarity of $C_{\tau,q}^*$ is higher than C_τ^* (Line 10).

Based on Lemma 5 and Theorem 4, Algorithm 6 returns the correct answer with a non-trivial probability:

Theorem 5. *Let a r_g^*-similar biclique C_τ^* be the global most similar biclique. Algorithm 6 returns C_τ^* with at least $r_g^{*|L(C_\tau^*)|-1}$ probability in a running time of $O\left((m + n_L \log n_L) \cdot n_L \cdot 2^{n_L}\right)$.*

Proof. By Lemma 5, Algorithm 5 can divide the left-side vertices of the global most similar biclique C_τ^* into the same group with probability $r_g^{*|L(C_\tau^*)|-1}$. The time complexity of Algorithm 6 is the same as Algorithm 4, which is $O\left((m + n_L \log n_L) \cdot n_L \cdot 2^{n_L}\right)$. More specifically, the number of different q in Line 5 is equal to $\sum_{i=1}^{d} |L(S^{(i)})| = |L(G_r)| \leq n_L$. Therefore, Algorithm 6 is also equivalent to running Algorithm 3 for n_L times, and its running time is the same as Algorithm 4. □

By Theorem 5, Algorithm 6 can return the exact answer only when $r_g^* = 1$, or when r_g^* is close to 1 and the number of left-side vertices in C_τ^* is not large. According to our experiments in Section 6.2, *Mosib-GloApp* can return the exact answer on most datasets and is particularly accurate when τ is not large.

Table 2: Dataset Statistics

Dataset	$\lvert L(G)\rvert$	$\lvert R(G)\rvert$	$\lvert E\rvert$	Type
YouTube (Y)	94,238	30,087	293,360	Membership
GitHub (G)	56,519	120,867	440,237	Membership
Bibsonomy (B)	767,447	5,794	801,784	Assignment
BookCross (Bo)	105,278	340,523	1,149,739	Rating
CiteULike (C)	731,769	153,277	2,338,554	Assignment
Discogs (Di)	1,754,823	270,771	5,302,276	Affiliation
Amazon (A)	2,146,057	1,230,915	5,743,258	Rating
DBLP (D)	1,953,085	5,624,219	12,282,059	Authorship
Delicious (De)	833,081	33,778,221	101,798,957	Interaction
Orkut (O)	2,783,196	8,730,857	327,037,487	Affiliation
MAG (M)	10,541,560	2,784,240	1,095,315,106	Composition

6 EXPERIMENTS

6.1 Experimental Setup

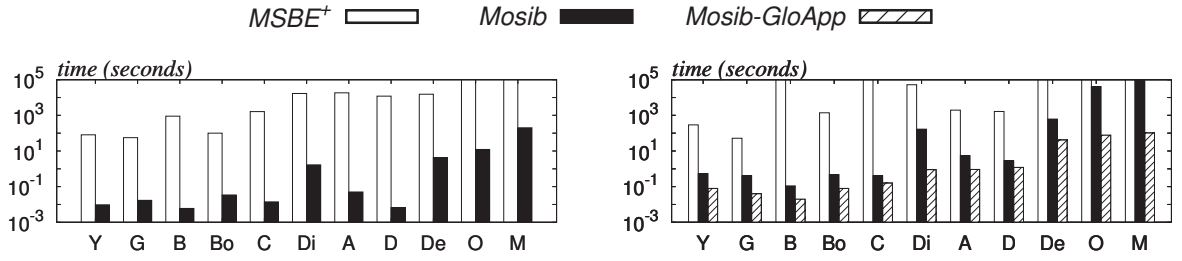
The code of this paper is available on GitHub[1].

Datasets. Table 2 lists the real-world datasets used in the experiments, and they are widely used in the literature of biclique search [26, 35, 40]. All datasets are available on KONECT[2]. We remove all edge directions, duplicated edges, and self-loops in the datasets.

Algorithms. We compare our solution with *MSBE*$^+$, a baseline algorithm based on the state-of-the-art algorithm for enumerating similar bicliques, i.e., *MSBE* [40]. We implement our algorithms in C++, and adopt the C++ implementation of *MSBE* [40] from the authors. Our experiments compare the following methods:

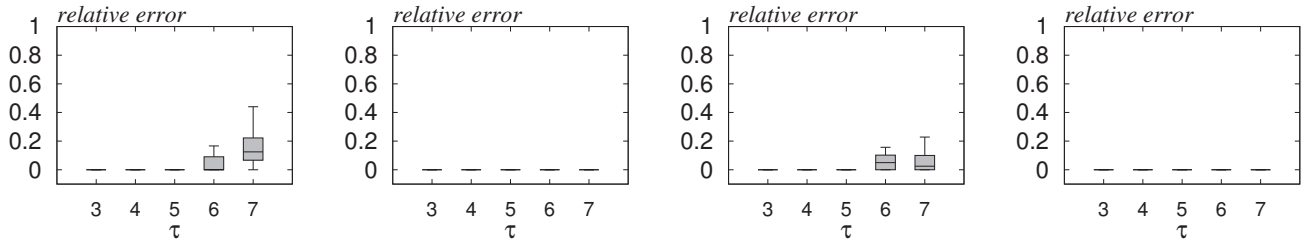
- *MSBE*$^+$: a baseline based on *MSBE* [40] (see Section 3). The best version of *MSBE* (i.e., *SS-MSBE*) is used.

[1] https://github.com/nedchu/mosib-release
[2] http://konect.cc/networks/



(a) local search (b) global search

Figure 6: Running Time of the Methods.



(a) YouTube (b) GitHub, Bibsonomy, Discogs, BookCross, CiteUlike (c) Amazon (d) DBLP, Delicious, Orkut, MAG

Figure 7: Accuracy of *Mosib-GloApp* (relative to *Mosib*).

- *Mosib*: our proposed exact algorithm (see Section 4).
- *Mosib-GloApp*: our proposed approximate algorithm for the global search (see Section 5).

Parameters. Unless otherwise specified, we set the size threshold $\tau = 5$ for the algorithms. For *Mosib-GloApp*, we set the number of hash functions $h = 10$ by default. In each experiment, we repeat the algorithm three times and report the average result. The program is terminated when it cannot finish within 24 hours.

Environment. The experiments are performed on a server with an Intel Xeon Silver 4210R 2.1GHz CPU and 256GB memory. All algorithms are implemented in C++ and compiled with g++7.5.0 under O3 optimization.

6.2 Efficiency and Effectiveness Analysis

Running Time of Local Search. For each dataset, we select 100 random vertices from the left-side vertices with top-500 high degree, and then report the average running time of local search on the 100 selected vertices. Figure 6a presents the average running time of local search ($\tau = 5$) of *MSBE*$^+$ and *Mosib*. The results show that our *Mosib* can outperform *MSBE*$^+$ in running time by up to six orders of magnitude (and at least 2,955x). On the largest dataset (i.e., MAG), our *Mosib* only takes 203 seconds for a local search query, while loading the graph into memory requires 452 seconds.

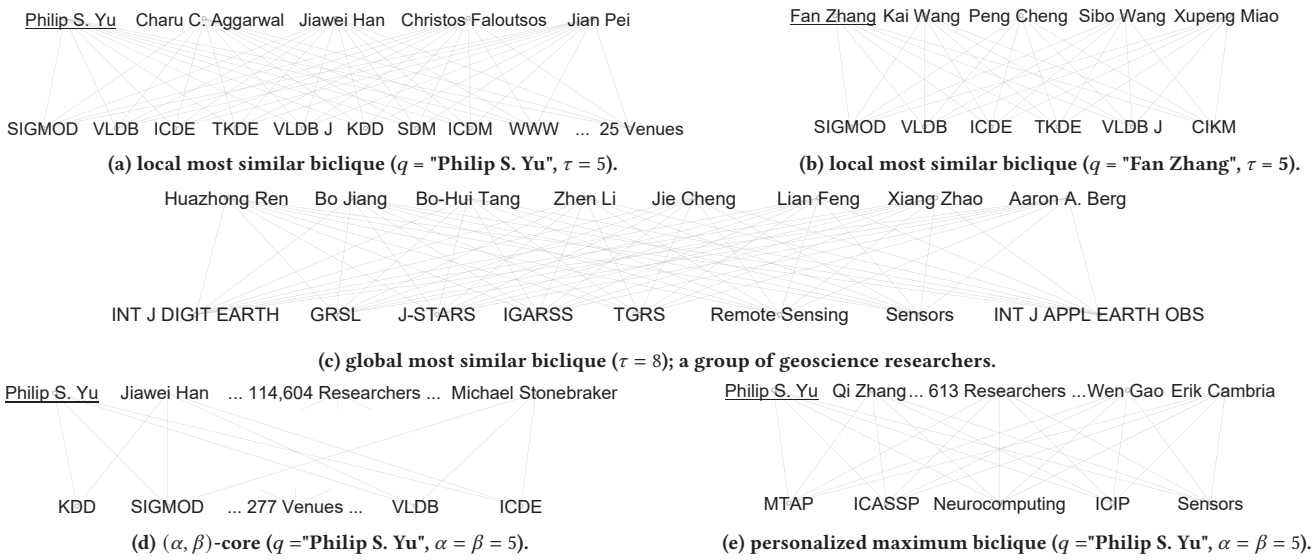
Running Time of Global Search. Figure 6b illustrates the running time of the global search ($\tau = 5$) of *MSBE*$^+$, *Mosib*, and *Mosib-GloApp*. Observe that our *Mosib* outperforms *MSBE*$^+$ in running time by up to five orders of magnitude (and at least 126x), while our approximate solution *Mosib-GloApp* is faster than *Mosib* by up to two orders of magnitude (and at least 2.5x). Unfortunately, our global exact algorithm cannot terminate in 24 hours on MAG, while *Mosib-GloApp* only takes 78 seconds, proving the importance of our approximate algorithms. In practice, the algorithms slow down when the number of vertices becomes relatively small (the average degree becomes large), e.g., on Bibsonomy and Discogs. This phenomenon significantly influences the running time of *MSBE*$^+$, while its impact on our *Mosib* and *Mosib-GloApp* is not strong.

Accuracy of *Mosib-GloApp*. We evaluate the accuracy of *Mosib-GloApp* with the relative error of *Mosib-GloApp* (compared with *Mosib*) in the similarity of the resulting global most similar biclique. In each test, we report the average of 100 independent tests, varying τ in $\{3, 4, 5, 6, 7\}$. Figure 7 presents the accuracy of *Mosib-GloApp*. The results show that *Mosib-GloApp* can produce the exact result on most datasets (9 out of 11) and is particularly accurate when τ is not large (e.g., $\tau \leq 5$). On the other hand, *Mosib-GloApp* produces inaccurate results on YouTube and Amazon when $\tau = 6, 7$. By Theorem 5, our *Mosib-GloApp* returns inaccurate results if and only if the similarity of the resulting biclique is less than 1. In this case, we recommend using *Mosib* to obtain the exact result.

6.3 Case Study: Results of Different Bipartite Cohesive Subgraph Models (DBLP)

This case study evaluates the effectiveness of our models on a real-world DBLP network. In particular, we construct a researcher-venue bipartite network using the latest release of DBLP[3], where a researcher-venue relationship exists if and only if the researcher

[3]https://dblp.org/xml/release/dblp-2024-03-01.xml.gz



(a) local most similar biclique (q = "Philip S. Yu", $\tau = 5$). **(b) local most similar biclique (q = "Fan Zhang", $\tau = 5$).**

(c) global most similar biclique ($\tau = 8$); a group of geoscience researchers.

(d) (α, β)-core (q ="Philip S. Yu", $\alpha = \beta = 5$). **(e) personalized maximum biclique (q ="Philip S. Yu", $\alpha = \beta = 5$).**

Figure 8: Case Study: Results of Different Models (DBLP).

has published at least one paper in a venue. The network contains 3,509,155 researchers, 16,751 venues, and 13,303,652 edges.

We report the most similar biclique on this network, and compare the results with well-known models such as similar biclique [40], (α, β)-core [21], and personalized maximum biclique [35].

Results of the Most Similar Biclique Search. Figure 8a depicts the local most similar biclique containing "Philip S. Yu" ($\tau = 5$), a professor whose H-index is among the top ten in computer science. The biclique contains 5 researchers and 34 venues, and the similarity between researchers is at least 0.201. All researchers are well-established scholars interested in data management, data mining, artificial intelligence, and information retrieval.

Figure 8b presents the local most similar biclique that contains "Fan Zhang" ($\tau = 5$), a faculty member in Guangzhou University. The biclique contains 5 researchers and 6 venues, and the similarity between researchers is no less than 0.412. Interestingly, these five researchers are all in their early careers, and they have a common interest in top-tier data management and mining venues.

Figure 8c shows the global most similar biclique when $\tau = 8$. The biclique contains 8 researchers and 8 venues, and the similarity between researchers is at least 0.889. All eight researchers are researchers in geoscience and remote sensing, and they published papers on well-known venues in geoscience.

Besides, the local search in Figure 8a (resp. 8b) takes 1.9 (resp. 0.8) seconds, while the global search in Figure 8c takes 2,610 seconds.

Results of Similar Biclique [40]. Given a similarity threshold s and a size constraint τ, the similar biclique problem aims to enumerate all maximal r-similar bicliques that satisfy $r \geq s$ and the size constraint. The *MSBE* [40] algorithm is the state-of-the-art for enumerating similar biclique.

Suppose we set $\tau = 5$ and start with a threshold $s = 0.5$, *MSBE* will not return any biclique that contains "Philip S. Yu" or "Fan Zhang", as the local most similar biclique that contains either of these vertices has a similarity less than 0.5. In addition, when we set $s = 0.5$ and $\tau = 5$, *MSBE* cannot terminate within three days.

Results of (α, β)-Core [21]. Given size thresholds α, β, the (α, β)-core model aims to find a bipartite subgraph C such that $|L(C)| \geq \alpha$ and $|R(C)| \geq \beta$. Liu et al. [21] are the first to study the (α, β)-core model, and they propose an index-based algorithm for the model.

We report the (α, β)-core connected component containing a query node q. The size thresholds are set to $\alpha = \beta = 5$. Figure 8d presents the $(5, 5)$-core that contains "Philip S. Yu", and the community contains 114, 607 researchers and 281 venues. The $(5, 5)$-core containing "Fan Zhang" has 7348 researchers and 18 venues. Therefore, it is hard to interpret the results given by the (α, β)-core model, as the resulting communities are too large to understand.

Results of Maximum Biclique [35]. To detect the maximum biclique containing a particular node, Wang et al. [35] develop the personalized maximum biclique model. Given a query node q and size constraints α, β, the personalized maximum biclique is the biclique C such that (i) C contains q, (ii) $|L(C)| \geq \alpha$ and $|R(C)| \geq \beta$, and (iii) C has the largest number of edges.

Figure 8e presents the personalized maximum biclique that contains "Philip S. Yu" ($\alpha = \beta = 5$), and it contains 617 researchers and 5 venues. However, the researchers and venues in the biclique are contrary to our intuition. The reason is that the maximum biclique model seeks the most popular sets of venues (i.e., the sets that can maximize the number of researchers), instead of the ones most relevant to "Philip S. Yu". In addition, the size of the personalized maximum biclique is too large to interpret. When q ="Fan Zhang", the personalized maximum biclique consists of 94 researchers and 5 venues, and the issues mentioned above still exist.

6.4 Parameter Analysis

In this section, we analyze how each parameter influences the performance of our algorithms, and validate the effectiveness of each

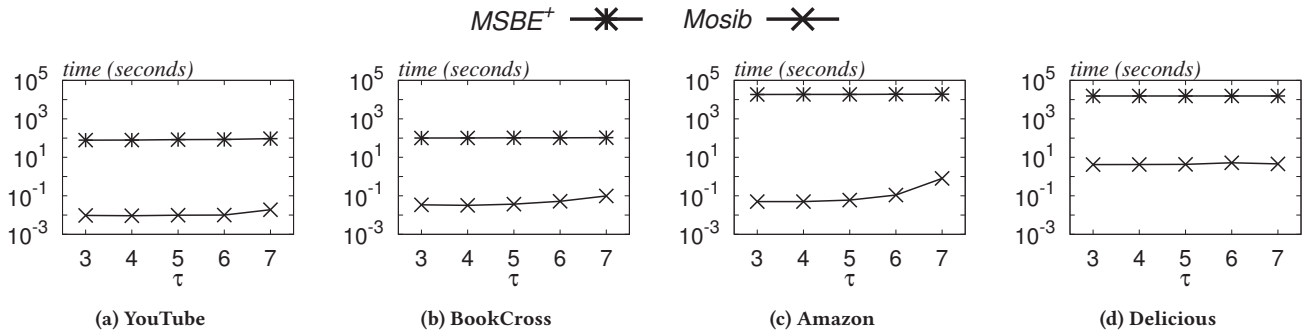


Figure 9: Running Time vs. τ (Local Search).

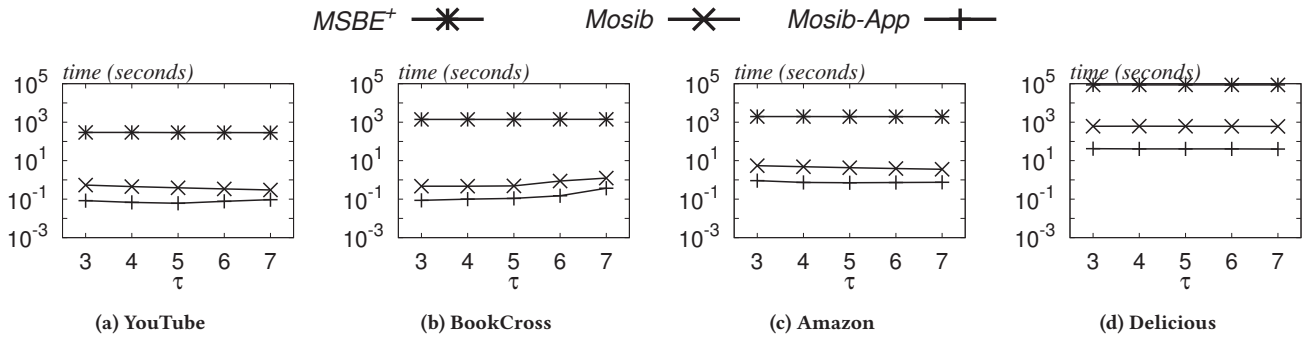


Figure 10: Running Time vs. τ (Global Search).

technique in our algorithms. The (α, β)-core [21] and personalized maximum biclique [35] models are not considered in this experiment, because the problem settings are different. For example, the average similarity of the $(5, 5)$-cores is 0, and the average similarity of the personalized maximum biclique ($\tau = 5$) is 0.01. Therefore, these methods can hardly meet the requirements of our problem.

Running Time v.s. Size Threshold τ. Figure 9 presents the local search time of the methods as a function of τ. On all datasets, our *Mosib* can outperform the baseline *MSBE*$^+$ by at least 433x in the running time of the local search. For *Mosib*, increasing τ from 3 to 7 will lead to an average increase of 2.97x in the running time of the local search, because it takes more effort to find an eligible biclique when τ becomes large.

Figure 10 reports the global search time of the methods for different τ. The experimental results show that our *Mosib* (resp. *Mosib-GloApp*) consistently outperforms the baseline *MSBE*$^+$ by at least 10x (resp. 260x) in the running time of the global search. When τ increases, the running time of *Mosib* (resp. *Mosib-GloApp*) stays almost the same. Compared with local search, global search is less sensitive to τ as it considers the results of a series of local searches.

Ablation Study on *Mosib*'s Rules (Local Search). This experiment evaluates the effectiveness of *Mosib*'s technique, including three graph reduction rules and two similarity-first search rules (see Section 6). For each rule (e.g., *SIM*), we implement a version of *Mosib* without this rule (e.g., w/o *SIM*), and then report the running time of this version in the relative percentage to that of *Mosib*.

Table 3 presents the results of this ablation study on *Mosib*'s rules. All of *Mosib*'s rules offer a non-trivial average speed up (see the second to last column), demonstrating the effectiveness of our algorithm design. Observe that rules *HOP*, *SIM*, and *SFS2* can provide an average speedup of over 18x in the running time, and that rules *SIM* and *SFS2* are particularly effective on large networks such as Delicious and Orkut. This shows that the rules based on similarity (i.e.,*SIM* and *SFS2*) are of great importance in the most similar biclique search. On sparse networks (e.g., Amazon and DBLP) whose average degree is relatively small, rules *HOP* and *DEG* are useful, because the degrees of vertices and the size of the 2-hop neighborhood are also relatively small on these networks.

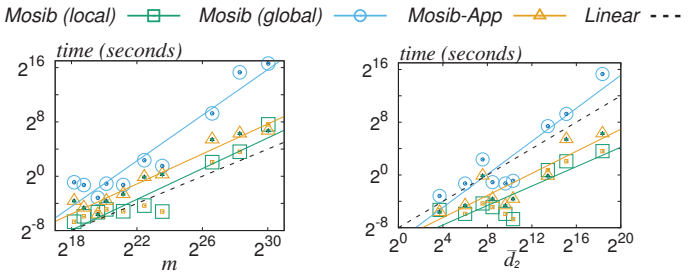
Scalability v.s. Different Topological Properties. Figure 11 reports the scalability of the methods, varying the size and topological properties of datasets. The linear scalability is marked with a dashed line. In Figure 11a, when the number of edges increases, the running time of *Mosib* (local) and *Mosib-GloApp* grows almost linearly while that of *Mosib* (global) grows a little faster than linear scalability. Figure 11b plots the scalability regarding the average number of 2-hop neighbors of each dataset, where the running time of our methods still scales almost linearly.

6.5 Degree Distributions

The skewed degree distribution in real-world bipartite graphs may be a potential issue to the performance of our algorithms. Figure 12 analyzes the degree distributions of Orkut. The degree follows

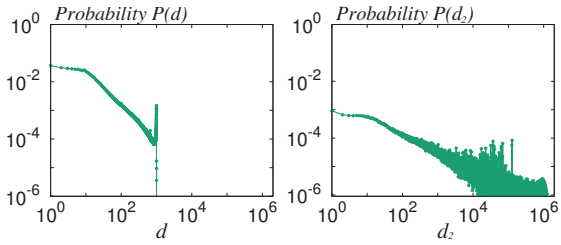
Table 3: Ablation Study on *Mosib*'s Rules (Local Search). The mark "–" means the test cannot terminate in 24 hours.

	YouTube	GitHub	Bibsonomy	BookCross	CiteULike	Discogs	Amazon	DBLP	Delicious	Orkut	MAG	Average	Min – Max
Mosib	0.009s	0.017s	0.006s	0.034s	0.014s	1.67s	0.051s	0.006s	4.20s	12.23s	203s	20.1s	0.006s – 203s
w/o *HOP*	438%	306%	413%	559%	1,857%	332%	4,380%	22,727%	450%	217%	120%	2,891%	120% – 22,727%
w/o *DEG*	823%	141%	165%	168%	129%	103%	4,440%	212%	106%	109%	103%	591%	103% – 4,440%
w/o *SIM*	302%	1,588%	133%	1,029%	600%	25,478%	280%	111%	20,643%	–	–	5,574%	111% – 25,478%
w/o *SFS*	115%	147%	117%	159%	171%	163%	220%	112%	145%	134%	157%	149%	112% – 220%
w/o *SFS2*	146%	312%	167%	471%	171%	173%	160%	105%	13,882%	2,584%	189%	1,669%	105% – 13,882%



(a) m: number of edges (b) $\bar{d}_2$: average number of 2-hop neighbors

Figure 11: Scalability v.s. Dataset Topological Properties.



(a) d: the degree (b) d_2: the 2-hop degree

Figure 12: Degree Distributions (Orkut).

a skewed power-law distribution (see Figure 12a). In addition, the 2-hop degrees of vertices are high and skewed, i.e., many areas are extremely dense (see Figure 12b). Such a skewed degree phenomenon exists in all datasets used in our experiments. Despite the skewed degrees of Orkut, our *Mosib* can answer the local search within 12 seconds while loading Orkut into memory takes 88 seconds. Our proposed algorithms can handle real-world graphs with skewed degrees, because they can quickly find high-similarity bicliques and prune the unpromising vertices in the graph.

7 RELATED WORKS

Biclique Search and Enumeration. The biclique search aims to find a single biclique with the desired properties. Lyu et al. [26, 27] study the maximum edge biclique problem (i.e., finding the biclique with the largest number of edges), and they develop an efficient algorithm that can scale to billion-scale graphs. Wang et al. [35] aim to find the maximum edge biclique containing a query node q. The maximum vertex biclique problem can be solved in polynomial time via integer programming or maximum matching [17].

There is also a line of works on enumerating maximal bicliques. Zhang et al. [44] propose an efficient branch-and-bound algorithm for enumerating bicliques. Abidi et al. [1] propose a pivot-based algorithm for maximal biclique enumeration that significantly improves efficiency. After that, Chen et al. [9] design an algorithm based on unilateral order and batch-pivot, and the algorithm achieves state-of-the-art performance. Zhao et al. [45] and Wang et al. [37] study the maximal biclique enumeration on uncertain bipartite graphs. The algorithms above cannot solve our studied problems, as they do not consider the structural similarity between vertices.

Simlarity-based Community Search. Yao et al. [40] propose *MSBE*, an efficient algorithm to enumerate all similar bicliques. Zhang et al. [43] propose the (k, r)-core model on attributed graphs. For a specific (k, r)-core, each vertex has at least k neighbors, and the attribute similarity of every vertex pair is at least r. Given a road network, Rai et al. [28] aim to retrieve k communities with high POI similarities and spatial closeness. These works require unipartite input graphs and auxiliary information (i.e., attributes or POI), so they cannot be applied to solve our studied problems.

Other Cohesive Subgraph Models. Cohesive subgraphs are fundamental models in analyzing large networks [13, 37–39, 45]. Besides the biclique model, there is a large body of literature on bipartite cohesive subgraphs, e.g., (α, β)-core [21, 36], bi-plex [25, 41], bi-truss [30, 34, 48]. Notably, this book [16] presents a comprehensive survey of bipartite cohesive subgraph mining. On unipartite graphs, there are various models for finding cohesive subgraphs, including k-core [3, 12, 14, 18, 46], k-truss [15, 32], and clique [10, 42].

8 CONCLUSION

This paper is the first to study the local/global most similar biclique search problem. Despite the NP-hardness, we develop an exact algorithm *Mosib*. The algorithm incorporates three graph reduction rules and two similarity-first search rules that can significantly improve practical efficiency. Besides, we devise an approximate algorithm *Mosib-GloApp* that can speed up the global most similar biclique search. The experiments show that our algorithms can outperform the baseline in running time by orders of magnitude, and provide meaningful insights into a network with the case studies.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China (U2436208, 62372129), the Guangdong S&T Programme under Grant 2024B0101010002, the Guangdong Basic and Applied Basic Research Foundation (2024A1515011501, 2023A1515012603) and the Australian Research Council (DP230101445, FT210100303). Deming Chu is supported by the scholarship of the China Scholarship Council (202006140012).

REFERENCES

[1] Aman Abidi, Rui Zhou, Lu Chen, and Chengfei Liu. 2020. Pivot-based Maximal Biclique Enumeration.. In *IJCAI*. 3558–3564.

[2] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Seyed-Mehdi-Reza Beheshti, Elisa Bertino, and Norman Foo. 2013. Collusion detection in online rating systems. In *APWeb*. Springer, 196–207.

[3] Vladimir Batagelj and Matjaz Zaversnik. 2003. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[4] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*. 119–130.

[5] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 2000. Min-Wise Independent Permutations. *J. Comput. Syst. Sci.* 60, 3 (2000), 630–659.

[6] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *CACM* 16, 9 (1973), 575–577.

[7] Lijun Chang. 2019. Efficient maximum clique computation over large sparse graphs. In *KDD*. 529–538.

[8] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2021. Efficient exact algorithms for maximum balanced biclique search in bipartite graphs. In *SIGMOD*. 248–260.

[9] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2022. Efficient maximal biclique enumeration for large sparse bipartite graphs. *PVLDB* 15, 8 (2022), 1559–1571.

[10] James Cheng, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. 2011. Finding maximal cliques in massive networks. *TODS* 36, 4 (2011), 1–34.

[11] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: Graph processing at facebook-scale. *PVLDB* 8, 12 (2015), 1804–1815.

[12] Deming Chu, Fan Zhang, Xuemin Lin, Wenjie Zhang, Ying Zhang, Yinglong Xia, and Chenyi Zhang. 2020. Finding the best k in core decomposition: A time and space optimal solution. In *ICDE*. IEEE, 685–696.

[13] Deming Chu, Fan Zhang, Xuemin Lin, Wenjie Zhang, Ying Zhang, Yinglong Xia, and Chenyi Zhang. 2024. Discovering and Maintaining the Best k in Core Decomposition. *TKDE* (2024).

[14] Deming Chu, Fan Zhang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2022. Hierarchical core decomposition in parallel: From construction to subgraph search. In *ICDE*. IEEE, 1138–1151.

[15] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008), 1–29.

[16] Yixiang Fang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2022. *Cohesive subgraph search over large heterogeneous information networks*. Springer.

[17] Michael R Garey and David S Johnson. 1979. Computers and intractability. (1979).

[18] Wissam Khaouid, Marina Barsky, Venkatesh Srinivasan, and Alex Thomo. 2015. K-core decomposition of large networks on a single PC. *PVLDB* 9, 1 (2015), 13–23.

[19] Janez Konc and Dušanka Janezic. 2007. An improved branch and bound algorithm for the maximum clique problem. *proteins* 4, 5 (2007), 590–596.

[20] Michael Ley. 2002. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *International symposium on string processing and information retrieval*. Springer, 1–10.

[21] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2019. Efficient (α, β)-core computation: An index-based approach. In *WWW*. 1130–1141.

[22] Guimei Liu, Kelvin Sim, and Jinyan Li. 2006. Efficient mining of large maximal bicliques. In *Data Warehousing and Knowledge Discovery*. Springer, 437–448.

[23] Jinze Liu and Wei Wang. 2003. Op-cluster: Clustering by tendency in high dimensional space. In *ICDM*. IEEE, 187–194.

[24] Can Lu, Jeffrey Xu Yu, Hao Wei, and Yikai Zhang. 2017. Finding the maximum clique in massive graphs. *PVLDB* 10, 11 (2017), 1538–1549.

[25] Wensheng Luo, Kenli Li, Xu Zhou, Yunjun Gao, and Keqin Li. 2022. Maximum Biplex Search over Bipartite Graphs. In *ICDE*. IEEE, 898–910.

[26] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2020. Maximum biclique search at billion scale. *PVLDB* (2020).

[27] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2022. Maximum and top-k diversified biclique search at scale. *VLDB J.* 31, 6 (2022), 1365–1389.

[28] Niranjan Rai and Xiang Lian. 2021. Top-k community similarity search over large road-network graphs. In *ICDE*. IEEE, 2093–2098.

[29] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. 2017. The ubiquity of large graphs and surprising challenges of graph processing. *PVLDB* 11, 4 (2017), 420–431.

[30] Jessica Shi and Julian Shun. 2022. Parallel algorithms for butterfly computations. In *Massive Graph Analytics*. Chapman and Hall/CRC, 287–330.

[31] Amos Tanay, Roded Sharan, and Ron Shamir. 2002. Discovering statistically significant biclusters in gene expression data. *Bioinformatics* 18, suppl_1 (2002), S136–S144.

[32] Jia Wang and James Cheng. 2012. Truss Decomposition in Massive Networks. *PVLDB* 5, 9 (2012).

[33] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*. 501–508.

[34] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient bitruss decomposition for large-scale bipartite graphs. In *ICDE*. IEEE, 661–672.

[35] Kai Wang, Wenjie Zhang, Xuemin Lin, Lu Qin, and Alexander Zhou. 2022. Efficient personalized maximum biclique search. In *ICDE*. IEEE, 498–511.

[36] Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, Lu Qin, and Yuting Zhang. 2021. Efficient and effective community search on large-scale bipartite graphs. In *ICDE*. IEEE, 85–96.

[37] Kai Wang, Gengda Zhao, Wenjie Zhang, Xuemin Lin, Ying Zhang, Yizhang He, and Chunxiao Li. 2023. Cohesive Subgraph Discovery over Uncertain Bipartite Graphs. *TKDE* (2023).

[38] Yiqi Wang, Long Yuan, Zi Chen, Wenjie Zhang, Xuemin Lin, and Qing Liu. 2023. Towards efficient shortest path counting on billion-scale graphs. In *ICDE*. IEEE, 2579–2592.

[39] Jiadong Xie, Zehua Chen, Deming Chu, Fan Zhang, Xuemin Lin, and Zhihong Tian. 2024. Influence Maximization via Vertex Countering. *PVLDB* 17, 6 (2024), 1297–1309.

[40] Kai Yao, Lijun Chang, and Jeffrey Xu Yu. 2022. Identifying similar-bicliques in bipartite graphs. *PVLDB* 15, 11 (2022), 3085–3097.

[41] Kaiqiang Yu, Cheng Long, P Deepak, and Tanmoy Chakraborty. 2021. On efficient large maximal biplex discovery. *TKDE* 35, 1 (2021), 824–829.

[42] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. 2016. Diversified top-k clique search. *VLDB J.* 25, 2 (2016), 171–196.

[43] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. When Engagement Meets Similarity: Efficient (k, r)-Core Computation on Social Networks. *PVLDB* (2017).

[44] Yun Zhang, Charles A Phillips, Gary L Rogers, Erich J Baker, Elissa J Chesler, and Michael A Langston. 2014. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. *Bioinformatics* 15 (2014), 1–18.

[45] Gengda Zhao, Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Yizhang He. 2022. Efficient computation of cohesive subgraphs in uncertain bipartite graphs. In *ICDE*. IEEE, 2333–2345.

[46] Zhongxin Zhou, Wenchao Zhang, Fan Zhang, Deming Chu, and Binghao Li. 2022. VEK: a vertex-oriented approach for edge k-core problem. *World Wide Web* 25, 2 (2022), 723–740.

[47] Zhiguo Zhu, Jingqin Su, and Liping Kong. 2015. Measuring influence in online social network based on the user-content bipartite graph. *Computers in Human Behavior* 52 (2015), 184–189.

[48] Zhaonian Zou. 2016. Bitruss decomposition of bipartite graphs. In *DASFAA*. Springer, 218–233.