Theoretically and Practically Efficient Maximum Biclique Search

ABSTRACT

Identifying the maximum edge biclique in bipartite graphs, a complete bipartite subgraph with the largest number of edges, plays a crucial role in uncovering densely-connected communities and has significant applications in domains such as recommendation systems and biological network analysis. However, this problem is NP-hard, and existing methods face inefficiencies both in practice and theory. In this paper, we propose two novel algorithms with distinct branching strategies and provable time guarantees for solving the maximum edge biclique search problem. The first is a refined pivot-based branching algorithm that systematically exploits vertex adjacency relationships to bound the search for the maximum edge biclique, achieving a time complexity of $O(m \cdot 1.348^n)$. The second is a cover-based algorithm that uncovers a novel duality between maximal bicliques and minimal vertex covers in bipartite graphs, attaining a complexity of $O(m \cdot 1.381^n)$. To the best of our knowledge, these two algorithms achieve the best-known worst-case time complexities for this problem. Notably, while the cover-based algorithm has a marginally higher theoretical complexity, it typically provides superior practical performance on dense graphs due to its inherent pruning efficiency. To further enhance performance, we introduce advanced pruning techniques, including polynomial-time solvable graph cases, neighbor and non-neighbor constraint-based upper bounds, vertex cover constraint-driven upper bounds, heuristic prioritization, and an improved progressive bounding approach. Additionally, we propose a hybrid framework that deploys the pivot-based approach for sparse graph regions and the cover-based approach for dense regions, balancing efficiency across varying graph structures. Extensive experiments on 12 real-world bipartite graphs demonstrate that our hybrid framework outperforms the state-of-the-art baseline by up to four orders of magnitude and achieves speedups of several times to orders of magnitude over the pivot-based approach on dense graphs.

1 INTRODUCTION

Bipartite graphs serve as a foundational structure for modeling relationships between entities in various real-world domains, such as user-item networks [45, 46], author-publication collaborations [22], and biological systems [25]. In these graphs, vertices are divided into two disjoint sets, and edges connect vertices exclusively across sets. A key task in analyzing bipartite graphs is the identification of cohesive subgraphs, as densely-connected substructures often correspond to meaningful communities or functional groups and can offer intrinsic insights for various applications. For instance, in user-item networks, cohesive subgraphs might represent the user groups with identical item preferences, offering valuable signals for targeted recommendations [17, 36] and community detection [11, 20, 22]. In biological networks, such cohesive subgraphs can reveal genes co-expression patterns, shedding light on shared regulatory mechanisms [7, 15, 37, 42].

In recent years, various cohesive subgraph models have been developed for bipartite graphs, including the biclique [2, 31, 35, 53], k-biplex [12, 39, 51], (α, β)-core [8, 28], k-bitruss [47, 57], and quasi-biclique [18, 30, 49]. Among them, the biclique is regarded as a fundamental structure, defined as a complete bipartite subgraph where every vertex in one set connects to all vertices in the opposing set. The maximum edge biclique problem, which identifies the biclique with the largest number of edges among all bicliques of a bipartite graph, is of significant interest due to its ability to capture the densely interconnected group of entities in various applications [4–7, 19, 21, 43]. Three detailed examples are listed below.

Anomaly detection. In bipartite networks, such as user-item graphs [45, 46], anomalous behavior frequently corresponds to the emergence of unusually dense substructures. A typical example is coordinated fraudulent activity, where a group of users collaboratively provides disproportionately positive ratings to a specific set of items. These behaviors can be effectively modeled and identified through the search for maximum bicliques, which reveal tightly coupled user-item interactions indicative of suspicious activity. Recently, biclique structures have been employed in applications such as detecting collusion in online rating systems [4] and mitigating group attacks in social networks [5]. Furthermore, our experiments (see Exp-9) also demonstrate the high precision and F1-scores when using the maximum bicliques for identifying anomalous patterns.

Recommendation system. In the context of targeted recommendation tasks [17, 36], cohesive subgraph structures are effective in modeling strong co-preference relationships between users and items. Identifying such patterns enables the extraction of high-confidence user communities along with their commonly favored items, which are instrumental in enhancing the precision of personalized recommendations. In particular, bicliques, complete bipartite subgraphs exhibiting full cohesiveness, serve as robust representations of mutual affinity, offering resilience to noise and improved interpretability. These properties make bicliques especially suitable for applications such as customized recommendations in user-movie networks, where they can be used to uncover groups of users with similar preferences [19].

Visualization analysis. Cohesive Subgraph Visualization (CSV) techniques project graph nodes and edges into a multi-dimensional space, wherein dense regions correspond to cohesive subgraphs [41, 50, 54]. This spatial embedding enables analysts to intuitively identify and interpret tightly interconnected bipartite communities, thereby supporting various graph analysis tasks. In this context, the maximum biclique plays a central role by quantifying the local density around each vertex. Specifically, the size of the largest biclique to which a vertex belongs is used as a measure of density. These density values are then visualized using the CSV technique introduced in [50], enabling the effective depiction of vertex density distributions in multi-dimensional space. Furthermore, we demonstrate that the maximum biclique-based CSV approach effectively captures and visualizes cohesive substructures within bipartite graphs, as illustrated in the experimental results (Exp-10) in Section 6.

SIGMOD '26, May 31 - June 05, 2026, Bengaluru, India

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Despite its practical significance, the maximum edge biclique problem is NP-hard [35], requiring exhaustive evaluation of vertex subsets to identify the optimal biclique. In recent years, several exact algorithms have been developed [2, 31, 40] to address this problem. However, these algorithms are typically limited to sparse bipartite graphs and fail to scale efficiently to dense bipartite graphs. For example, For instance, in the case of the MovieLens dataset (which contains only 943 and 1,682 vertices on each side, but has 100K edges), the state-of-the art existing algorithm cannot finish the computation within 24 hours as shown in Table 2. This renders a critical bottleneck, particularly for large, locally dense bipartite graphs. Moreover, to the best of our knowledge, the time complexity of these solutions are still bounded by a trivial worst-case complexity $O(P(n)2^n)$, leaving the theoretical complexity unresolved, where $P(n)$ is a polynomial function of the input size n.

Recent advances in maximal biclique enumeration [1, 10, 11, 14, 53] have improved the efficiency of discovering all maximal bicliques. However, these methods do not directly address the optimization challenge of extracting the maximum edge biclique, as the exponential number of maximal bicliques in local dense graphs renders enumeration intractable. For example, the state-of-the-art maximal biclique enumeration algorithm [11] focus mainly on pruning non-maximal bicliques but fail to efficiently filter the maximal bicliques that are not the maximum, leading to significant inefficiencies. Consequently, leveraging maximal biclique enumeration for solving this problem remains infeasible, and the development of theoretically sound and practically efficient solutions remains an open challenge.

Contributions. In this paper, we address the maximum edge biclique search problem by proposing two novel algorithmic frameworks with provable worst-case time guarantees, complemented by advanced branch reduction techniques and optimized upper bounds for practical efficiency. Our contributions are summarized as follows:

Two new search algorithms. We propose two distinct algorithms for maximum biclique search, each with complementary strengths. The first algorithm is a pivot-based algorithm, which extends pivot techniques from [24] by leveraging vertex pair neighborhood relationships to eliminate redundant search branches. This approach achieves a worst-case time complexity of $O(m \cdot 1.348^n)$, where n and m denote the number of vertices and edges, respectively. While this method provides marked efficiency gains on sparse graphs, its performance on dense graphs remains constrained. To overcome this limitation, we introduce a cover-based algorithm grounded in a novel duality between maximal bicliques and minimal vertex covers in bipartite graphs. By reformulating the problem through complement graph processing, this method achieves a time complexity of $O(m \cdot 1.381^n)$, surpassing existing state-of-the-art approaches to the best of our knowledge. Finally, we unify these advances into an adaptive hybrid framework that deploys the pivot-based method for sparse subgraphs and the cover-based method for dense subgraphs, ensuring robust performance across diverse graph densities.

Novel branch pruning techniques. To accelerate both algorithms, we propose a suite of branch-and-bound optimizations and tight upper bounds. For the pivot-based algorithm, we characterize polynomial time solvable cases for bipartite graphs where each vertex has at most one non-neighbor in the opposing partition, enabling systematic elimination of non-promising branches. We further develop a

Table 1: Frequently-used notations

Notations	Descriptions
$G = (U, V, E)$	The undirected and unweighted bipartite graph
$N_v(G), \overline{N}_v(G)$	The set of neighbors, non-neighbors of the vertex v in G
$d_v(G), \overline{d}_v(G)$	The degree of the vertex v in G, and the cardinality of $\overline{N}_v(G)$
$G(S_U, S_V)$	The subgraph of G induced by sets $S_U \subseteq U$ and $S_V \subseteq V$
τ_U (τ_U^i), $\tau_V(\tau_V^i)$	The size constraints τ_U (τ_U^i) in side U and τ_V (τ_V^i) in side V
$I = (G, S_U, S_V, C_U, C_V)$	The instance for finding the maximum biclique containing (S_U, S_V) in $G(S_U \cup C_U, S_V \cup C_V)$
$B = (\overline{G}, S_U, S_V, C_U, C_V)$	The instance for finding the maximum biclique containing (S_U, S_V) in $G(S_U \cup C_U, S_V \cup C_V)$
ω, ρ_G	The size of the maximum biclique of G, and the density of G

linear-time computable upper bound derived from vertex neighbor and non-neighbor constraints in ordered vertex sequences, which prunes search branches dramatically. For the cover-based algorithm, we identify a structural property: graphs where vertices in the complement bipartite graph have at most two neighbors admit polynomial-time exact solutions. We also derive a linear-time computable upper bound on biclique size by establishing the lower bound of the minimal vertex cover size in the complement graph. These advancements are further enhanced through the heuristic strategy and an improved progressive bounding mechanism to iteratively refine the search space during execution.

Extensive experiments. We evaluate our algorithms on 12 real-world bipartite graphs, demonstrating significant efficiency and scalability improvements over state-of-the-art baselines. Experimental results reveal that our methods outperform existing solutions by up to four orders of magnitude under diverse parameter settings. For instance, on the Sualize dataset (449K edges), our algorithms find the maximum bicliques with size ≥ 5 of each side in under 6.2 seconds, whereas prior methods fail to terminate within 24 hours. These results underscore the practical impact of our theoretical advancements. Moreover, our hybrid framework achieves speedups of several times to orders of magnitude over the pivot-based approach on dense graphs, which further demonstrates the practical efficiency of the proposed cover-based approach. For reproducibility, our source code is publicly available at an anonymized repository: https://anonymous.4open.science/r/MaximumBiclique-2BC3.

2 PROBLEM DEFINITION

In this paper, we consider an unweighted and undirected bipartite graph $G = (U, V, E)$, where U and V represent two disjoint independent sets and E is the set of edges between U and V, i.e., $E \subseteq U \times V$. The total number of vertices and edges in G are denoted as $n = |U| + |V|$ and $m = |E|$, respectively. Given a vertex $u \in U$, we define $N_u(G) = \{w \in V \mid (u, w) \in E\}$ (and $\overline{N}_u(G) = V \setminus N_u(G)$) as the set of neighbors (and non-neighbors) of u in G. The degree (and non-degree) of vertex u in G is represented as $d_u(G) = |N_u(G)|$ (and $\overline{d}_u(G) = |V| - d_u(G)$). Let $S_U \subseteq U$ and $S_V \subseteq V$ be subsets of U and V, respectively. We denote by $G(S_U, S_V) = (S_U, S_V, E(S_U, S_V))$ the subgraph of G induced by the vertex subsets S_U and S_V, where $E(S_U, S_V) = \{(u, v) \in E \mid u \in S_U, v \in S_V\}$. For convenience, we also use $N_u(S_V)$ $(\overline{N}_u(S_V))$ and $d_u(S_V)$ $(\overline{d}_u(S_V))$ to denote the set of neighbors (non-neighbors) and the degree (non-degree) of u in the subgraph $G(S_U, S_V)$, respectively. Similar definitions apply for the vertices in V. Given a bipartite graph $G = (U, V, E)$, we define $\overline{G} = (U, V, \overline{E})$ as the complement of G, where $\overline{E} = U \times V \setminus E$, i.e., for each pair of $u \in U$ and $v \in V$, (u, v) is either included in E or in $\overline{E}$. We now proceed to define the concept of a biclique.

(a) The running example graph G (b) The subbranch I_0 containing v_5 (c) The subbranch I_1 containing u_6 and v_6 (d) The subbranch I_2 containing u_2

Figure 1: The branching strategies of Theorem 3.3, when selected v_5 as the pivot vertex. We obtain three subbranches I_0, I_1 and I_2 to find the maximum biclique of G, where the gray (black) vertices in (a) are not considered for generating new subbranches.

Definition 1 (Biclique). Given a bipartite graph $G = (U, V, E)$, the subgraph $G(S_U, S_V)$ with $S_U \subseteq U$ and $S_V \subseteq V$ is a biclique in G if every vertex $v \in S_U$ is connected to all vertices in S_V.

Given a biclique $G(S_U, S_V)$ in G, if there is no other vertex in G that can be added to $G(S_U, S_V)$ to form a larger biclique, we say that $G(S_U, S_V)$ is maximal in G. Moreover, if a maximal biclique $G(S_U, S_V)$ contains the most edges among all maximal bicliques in G, we refer to $G(S_U, S_V)$ as the maximum edge biclique of G. In this paper, we simply call the maximum edge biclique the maximum biclique. However, in real-world bipartite graph applications, the maximum biclique may be one where one side has a large number of vertices, while the other side contains only one or two vertices, which can have limited practical utility. To avoid identifying overly skewed bicliques, we can constrain the size of the biclique on each side, as investigated in [31]. Thus, the maximum biclique search problem studied in this paper is defined as follows.

Problem Definition: Given a bipartite graph $G = (U, V, E)$ and two positive integers τ_U and τ_V, our problem is to find a biclique $G(S_U, S_V)$ in G with the maximum number of edges, subject to the constraints $|S_U| \geq \tau_U$ and $|S_V| \geq \tau_V$. Note that this problem can well be regarded as a specific constrained version of the classical maximum biclique search problem, i.e., finding a maximum biclique among all the bicliques that satisfy $|S_U| \geq \tau_U$ and $|S_V| \geq \tau_V$.

NP-hardness. This problem is NP-complete [35], and is also challenging to approximate in polynomial time with theoretical guarantees [33]. Consequently, finding a maximum biclique in bipartite graphs remains a difficult problem. In recent years, significant efforts have been made to develop practical algorithms [2, 31, 38, 40]. We now briefly review the current state-of-the-art (SOTA) solution.

2.1 The State-of-the-Art Solution

To the best of our knowledge, the current SOTA solution for finding the maximum biclique in bipartite graphs is presented in [31]. The main idea of this solution is to apply a progressive bounding technique to partition the original problem into a series of subproblems. Specifically, given a bipartite graph G and the size ω' of a near-maximum biclique in G, the task of finding the maximum biclique $G(S_U, S_V)$ in G such that $|S_U| \geq \tau_U$ and $|S_V| \geq \tau_V$ reduces to finding the maximum biclique of G for each pair of τ_U^i and τ_V^i, where $\tau_U^i = 2^i \times \tau_U$ and $\tau_V^i = \omega'/(2 \times \tau_U^i)$. Subsequently, the authors further leverage techniques from [53] and develops pruning strategies to efficiently address each subproblem.

However, the SOTA solution still faces several challenges. First, the efficiency of the progressive bounding technique heavily depends on the size ω' found by the heuristic algorithm. If ω' is small, the size constraints τ_U^i and τ_V^i become too loose, resulting in inefficiencies. Second, as the bipartite graphs grow denser, the efficiency of this solution declines significantly, as the pruning techniques become less effective. Lastly, the time complexity of this SOTA solution is still bounded by trivial worst-case complexity $O(P(n)2^n)$, leaving its theoretical complexity unresolved, where $P(n)$ is a polynomial function of the input size n.

To overcome these issues, we propose two distinct algorithms for maximum biclique search, each with complementary strengths. Notably, both of our algorithms ensure a non-trivial worst-case time complexity while significantly enhancing practical efficiency. The details of these techniques are provided in the following sections.

3 A PIVOT-BASED SEARCH FRAMEWORK

In this section, we present a new framework for searching the maximum biclique based on refined pivoting techniques and novel upper bound techniques. We prove that the time complexity of the proposed framework is bounded by $O(m \cdot 1.348^n)$. To the best of our knowledge, this approach represents the algorithm with the best worst-case time complexity for this problem to date.

3.1 New Branch-and-Bound Rules

To enumerate the maximum biclique of a given bipartite graph G, a straightforward approach is to adapt the technique developed in [11], originally designed for enumerating maximal bicliques. Generally, let $I = (G, S_U, S_V, C_U, C_V)$ be an instance representing the task of finding the maximum biclique containing $G(S_U, S_V)$, where $C_U \subseteq U \setminus S_U$ and $C_V \subseteq V \setminus S_V$ serve as the candidate sets used to expand S_U and S_V, i.e., for each $u \in C_U$ (and $v \in C_V$), $G(S_U \cup \{u\}, S_V)$ (and $G(S_U, S_V \cup \{v\})$) forms a larger biclique in G. The original instance I can be iteratively partitioned into a series of subinstances by moving each vertex u from C_U (or v from C_V) to S_U (or S_V) to identify the maximum biclique containing $G(S_U, S_V)$ and u (or v). By exhaustively exploring all subinstances derived from initial instance $I = (G, \emptyset, \emptyset, U, V)$, one can eventually determine the globally optimal solution in G. To mitigate combinatorial explosion inherent in this process, [11] introduced a pivot-based branching strategy formalized as follows:

THEOREM 3.1 ([11]). *Given an instance $I = (G, S_U, S_V, C_U, C_V)$ and a vertex $u \in C_U$ (called the pivot vertex), let $\overline{N}_u(C_V) = \{v_1, v_2, \ldots, v_{\overline{d}_u}\}$ denote the set of non-neighbors of u in C_V. Then, the maximum biclique in I is equivalent to the maximum among the following $\overline{d}_u + 1$ subinstances: (1) $I_0 = (G, S_U \cup \{u\}, S_V, C_U \setminus \{u\}, N_u(C_V))$; (2) $I_i = (G, S_U, S_V \cup \{v_i\}, N_{v_i}(C_U), C_V \setminus \{v_1, \ldots, v_i\})$, where $i \in [1, \overline{d}_u]$.*

Note that, in instance $I = (G, S_U, S_V, C_U, C_V)$, Theorem 3.1 selects only one vertex from candidate sets as the pivot vertex to generate $\overline{d}_u + 1$ subinstances, which is dramatically less than $|C_U| + |C_V|$ subinstances generated by the traditional method. To obtain a better performance, we typically select the vertex with the fewest neighbors in $G(C_U, C_V)$ as the pivot vertex in practical implementations.

Improved pivot-based techniques. We identify additional opportunities to eliminate redundant computations in Theorem 3.1. Consider an instance $I = (G, S_U, S_V, C_U, C_V)$ seeking the maximum biclique containing $G(S_U, S_V)$. After processing pivot vertex $u \in C_U$ via subinstance I_0, Theorem 3.1 will generate $\overline{d}_u$ subinstances by iteratively adding non-neighbors $v \in \overline{N}_u(C_V)$ to S_V. However, if two vertices $v_1, v_2 \in \overline{N}_u(C_V)$ satisfy $N_{v_1}(C_U) \subseteq N_{v_2}(C_U)$, then any biclique in $I_1 = (G, S_U, S_V \cup \{v_1\}, N_{v_1}(C_U), C_V)$ is strictly suboptimal. This is because, for every biclique $G(R_U, R_V)$ in I_1, $G(R_U, R_V \cup \{v_2\})$ forms a larger biclique discoverable via $I_2 = (G, S_U, S_V \cup \{v_2\}, N_{v_2}(C_U), C_V)$, making I_1 redundant. Based on the above analysis, we derive the following result.

Theorem 3.2. *Given an $I = (G, S_U, S_V, C_U, C_V)$ and a pivot vertex $u \in C_U$, let $D = \{v_1, v_2, \ldots, v_{\overline{d}'}\}$ be the largest subset of $\overline{N}_u(C_V)$ where $\overline{N}_{v_i}(C_U) \setminus \overline{N}_{v_j}(C_U) \neq \emptyset$ for all distinct $v_i, v_j \in D$. Then, the maximum biclique in I is equivalent to the maximum among the following $|D| + 1$ subinstances: (1) $I_0 = (G, S_U \cup \{u\}, S_V, C_U \setminus \{u\}, N_u(C_V))$; (2) $I_i = (G, S_U, S_V \cup \{v_i\}, N_{v_i}(C_U), C_V \setminus \{v_1, ..., v_i\})$, where $v_i \in D$ and $|D| \leq |\overline{N}_u(C_V)|$.*

Proof. Let $D' = D \setminus N_u(C_V)$ be the subset of $N_u(C_V)$ excluding D. We obtain that for each $v \in D'$, there always exists a vertex v_i in $\overline{N}_u(C_V)$ such that $N_v(C_U) \subseteq N_{v_i}(C_U)$. This result implies that any maximum biclique of G that includes v can be detected through the subinstance I_i. Thus, it is unnecessary to use the vertices in D' to expand $G(S_U, S_V)$ for generating subinstances. This completes the proof. □

Theorem 3.2 clearly establishes from our earlier analysis. We now further refine it to reduce more redundant computations. Consider two vertices $v_1, v_2 \in \overline{N}_u(C_V)$ in instance $I = (G, S_U, S_V, C_U, C_V)$ with the pivot vertex $u \in C_U$. Let w be the only vertex in $\overline{N}_{v_2}(C_U) \setminus \overline{N}_{v_1}(C_U)$, i.e., $|\overline{N}_{v_2}(C_U) \setminus \overline{N}_{v_1}(C_U)| = 1$. We derive that the maximum biclique in the subinstance $I_1 = (G, S_U, S_V \cup \{v_1\}, N_{v_1}(C_U), C_V)$ must include w if it exceeds the size of bicliques in $I_2 = (G, S_U, S_V \cup \{v_2\}, N_{v_2}(C_U), C_V)$. This follows because any biclique $G(R_U, R_V)$ in I_1 excluding w satisfies $R_U \subseteq N_{v_2}(C_U)$, rendering it smaller than the bicliques in I_2. Hence, w is necessarily included in I_1, which allows us to derive the following refined theorem.

Theorem 3.3. *Under the conditions of Theorem 3.2, let $v_i \in D$ be a vertex such that $|\overline{N}_{v_j}(C_U) \setminus \overline{N}_{v_i}(C_U)| = 1$ for some $v_j \in D$. Suppose that u_i is the only vertex in $\overline{N}_{v_j}(C_U) \setminus \overline{N}_{v_i}(C_U)$. The subinstance I_i in Theorem 3.2 can be pruned unless u_i is included the result of I_i.*

Proof. The validity of this theorem follows directly. Because any biclique $G(R_U, R_V)$ in I_1 excluding w satisfies $R_U \subseteq N_{v_j}(C_U)$, which implies that a larger biclique can be determined by I_j. □

In Theorem 3.3, checking whether each pair of vertices v_i and v_j in D satisfies $|\overline{N}_{v_j}(C_U) \setminus \overline{N}_{v_i}(C_U)| = 1$ incurs significant computational cost. To alleviate this overhead, we employ an approximate strategy that slightly compromises pruning strength in exchange for improved efficiency. Specifically, we first identify a vertex $v_j \in D$ with the highest degree, and subsequently verify for each $v_i \in D$ whether the condition $|\overline{N}_{v_j}(C_U) \setminus \overline{N}_{v_i}(C_U)| = 1$ holds for the fixed vertex v_j. Our experimental results (see Exp-3) confirm the practical effectiveness of this approximation. The following example illustrates the application of Theorem 3.3.

Example 1. Consider the bipartite graph G in Fig.1(a), where v_5 is selected as the pivot vertex. By Theorem 3.1, the maximum biclique of G must contain v_5 or at least one vertex from $\overline{N}_{v_5}(G) = \{u_1, u_2, u_6\}$. Since all neighbors of u_1 are in $N_{u_2}(G)$, we can exclude u_1 based on Theorem 3.2. Thus, only $D = \{u_2, u_6\}$ and v_5 remain as candidates for generating subinstances. Additionally, since $v_6 \in N_{u_6}(G)$ is not adjacent to u_2 ($|N_{u_6}(G) \setminus N_{u_2}(G)| = 1$), any maximum biclique containing u_6 must include v_6 following Theorem 3.3. This leads to three subbranches, shown in Figs. 1(b), 1(c), and 1(d), guiding the search for the maximum biclique in Fig. 1(a).

3.2 Branch Reduction Techniques

In this subsection, we introduce several branch pruning rules to reduce non-productive branches during maximum biclique search. Given a branch $I = (G, S_U, S_V, C_U, C_V)$ to find the maximum biclique containing $G(S_U, S_V)$, our techniques operate as follows.

Reduction with $\overline{d}_u(C_V) = 0$ or $\overline{d}_v(C_U) = 0$. For this case, the following result can be directly derived.

Lemma 1. Given an instance $I = (G, S_U, S_V, C_U, C_V)$, assume $D_1 = \{u \in C_U | \overline{d}_u(C_V) = 0\}$ and $D_2 = \{v \in C_V | \overline{d}_v(C_U) = 0\}$. The maximum biclique in I must include the vertices in D_1 and D_2.

Proof. This lemma holds trivially. □

Reduction for special bipartite graphs with $\max_{u \in U} \overline{d}_u(G) \leq 1$ **and** $\max_{v \in V} \overline{d}_v(G) \leq 1$. We observe that the maximum biclique in such special bipartite graphs can be computed in polynomial time. The following theorem formalizes this result.

Theorem 3.4. *Given a bipartite graph $G = (U, V, E)$ where each vertex $u \in U$ (and $v \in V$) has at least $|V| - 1$ (and $|U| - 1$) neighbors, let $x = |U| \times |V| - |E|$, $y = ||U| - |V||$, and $z = \min(|U|, |V|)$. Then, the maximum biclique size ω in G is given by: (1) if $y \geq x$, $\omega = z \times (z + y - x)$; (2) otherwise, $\omega = (z - \lfloor (x - y)/2 \rfloor) \times (z - \lceil (x - y)/2 \rceil)$.*

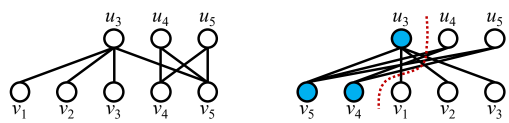
Proof. Let $\overline{E} = \{(u_1, v_1), (u_2, v_2), \cdots, (u_x, v_x)\}$ represent the set of missing edges in G, i.e., $\overline{E} = (U \times V) \setminus E$. We observe that by removing exactly one vertex from each edge in $\overline{E}$, the subgraph induced by the remaining vertices corresponds to a maximal biclique of G. Let a be the number of vertices removed from $\{u \in U | (u, v) \in \overline{E}\}$. Then, our goal is to find the value of a that maximizes the function of $f(a) = (|U| - a) \times (|V| - (x - a))$. Since $a \in \mathbb{N} \cap [0, x]$, we derive that if $y \geq x$, the value of a is either 0 or x, corresponding to the case (1); otherwise if $y < x$, the value of a is either $y + \lfloor \frac{x-y}{2} \rfloor$ or $\lfloor \frac{x-y}{2} \rfloor$, corresponding to the case (2). This completes the proof. □

By Theorem 3.4, we derive the following branch reduction.

Lemma 2. Given a branch $I = (G, S_U, S_V, C_U, C_V)$, let $u \in C_U$ and $v \in C_V$ be the two vertices satisfying $(u, v) \notin E$, $\overline{d}_u(C_V) \leq 1$, and $\overline{d}_v(C_U) \leq 1$. The maximum biclique in I is contained within the subinstance $I' = (G, S_U \cup \{u\}, S_V \cup \{v\}, C_U \setminus \{u\}, C_V \setminus \{v\})$.

Proof. Let $G(S_U, S_V)$ be a result obtained by subbranch I'. It follows that every $u' \in S_U$ and $v' \in S_V$ satisfies $\overline{d}_{u'}(S_V) \leq 1$ and $\overline{d}_{v'}(S_U) \leq 1$, enabling direct application of Theorem 3.4 to compute the maximum biclique in $G(S_U, S_V)$. Moreover, suppose for contradiction that the maximum biclique $G(S_U^*, S_V^*)$ in I is not contained in $G(S_U, S_V)$. Then, there exists a vertex $w \in S_U^* \cup S_V^*$ with $w \neq u, v$ such that either $(w, v) \notin E$ or $(u, w) \notin E$. This violates the constraints $\overline{d}_u(C_V) \leq 1$ and $\overline{d}_v(C_U) \leq 1$ as $(u, v) \notin E$. Thus, $G(S_U^*, S_V^*)$ must reside within $G(S_U, S_V)$ of I', computable via Theorem 3.4. □

Example 2. Fig.2 illustrates our branch reduction techniques. We consider the subgraph G in Fig.2(a) as the instance I. The vertex v_5 satisfies $\overline{d}_{v_5}(G) = 0$, indicating that the maximum biclique in I must include v_5 by Lemma 1. Furthermore, since $(u_3, v_4) \notin E$ satisfies



(a) An instance I (b) The subbranch I' of I

Figure 2: Illustration of branch reduction techniques. Fig. 2(a) is an initial instance I. Fig. 2(b) is the reduced subinstance I' of I derived from Lemma 1 and Lemma 2, which contains vertices u_3, v_4, and v_5.

Algorithm 1: $UB(G, U, V)$

1 Sort each vertex u in U with non-increasing order of $d_u(V)$;
2 Sort each vertex v in V with non-increasing order of $d_v(U)$;
3 $\omega_{ub} \leftarrow 0$; $ub_l \leftarrow 0$; $ub_r \leftarrow |V|$; $e_l \leftarrow 0$; $e_r \leftarrow 0$; $j \leftarrow |V|$;
4 Let u_i and v_j be the i-th and j-th vertex in U and V, respectively;
5 **foreach** i *in* 1 *to* $|U|$ **do**
6     $ub_l \leftarrow i$; $ub_r \leftarrow |d_{u_i}(V)|$; $e_l \leftarrow e_l + \overline{d}_{u_i}(V)$;
7   **if** $\omega_{ub} < ub_l \times ub_r$ **then**
8   **while** $e_r < e_l$ *and* $j \geq 1$ **do**
9             $e_r \leftarrow e_r + \overline{d}_{v_j}(U)$; $j \leftarrow j - 1$;
10         $ub_r \leftarrow \min(ub_r, j)$;
11   **if** $ub_l \geq \tau_U$ *and* $ub_r \geq \tau_V$ **then**
12         $\omega_{ub} \leftarrow \max(\omega_{ub}, ub_l \times ub_r)$
13 **return** ω_{ub};

$\overline{d}_{u_3}(G) = 1$ and $\overline{d}_{v_4}(G) = 1$, we construct a subinstance I' (depicted in Fig.2(b)) containing both u_3 and v_4 by Lemma 2. Notably, the residual subgraph $G(\{u_4, u_5\}, \{v_1, v_2, v_3\})$ in I' is edgeless, enabling direct resolution of the maximum biclique. Thus, applying these reduction rules, the instance I in Fig. 2(a) is solved in polynomial time, underscoring the efficacy of our techniques.

Reduction with $\overline{d}_u(C_V) = 1$ or $\overline{d}_v(C_U) = 1$. Beyond the special cases in Lemma 2, we extend pruning to branches where one vertex has exactly one non-neighbor. This leverages the current maximum biclique size ω for early termination.

Lemma 3. Given a branch $I = (G, S_U, S_V, C_U, C_V)$ with current maximum biclique size ω, let $u \in C_U$ satisfy $\overline{d}_u(C_V) = 1$, and let $w \in C_V$ be the sole non-neighbor of u ($(u, w) \notin E$). Then, the maximum biclique in I is contained in the subbranch $I' = (G, S_U \cup \{u\}, S_V, C_U \setminus \{u\}, C_V \setminus \{w\})$ if $d_w(S_U \cup C_U) < \sqrt{\omega}$.

Proof. Let $G(S_U, S_V)$ be the biclique in I with $|S_U| \times |S_V| \geq \omega$. Since w is the only non-neighbor of u in C_V, the biclique $G(S_U, S_V)$ must include either u or w. Assume $w \in S_V$ and $u \notin S_U$. When replacing w with u in $G(S_U, S_V)$, we obtain the biclique $G(S_U \cup \{u\}, S_V \setminus \{w\})$. Given $d_w(S_U \cup C_U) < \sqrt{\omega}$, we have $|S_U| < |S_V|$ and $|S_U| \times |S_V| \leq |S_U \cup \{u\}| \times |S_V \setminus \{w\}|$. Therefore, the maximum biclique in I must contain u. This completes the proof. ◻

3.3 Upper Bound Techniques

Except for the basic bounds in [31], we introduce novel upper bounds to early terminate branches that cannot exceed the current maximum biclique size ω. Our approach is based on the following principle: for any biclique $G(S_U, S_V)$, the maximum size of S_V is bounded by $\min_{u \in S_U} d_u(V)$. Consequently, we yield the following foundational bound.

Table 2: The upper bounds of the maximum biclique size in Fig. 1(a), where ub_1 and ub_2 are obtained by Lemma 4 and Lemma 5, respectively.

Ordering	u_2	u_3	u_1	u_4	u_5	u_6	v_1	v_2	v_3	v_4	v_5	v_6
id	1	2	3	4	5	6	1	2	3	4	5	6
d	4	4	3	3	3	1	3	3	3	3	3	3
$ub_1(id \times d)$	4	8	9	12	**15**	1	3	6	9	12	15	**18**
ub_2	4	8	**9**	6	3	0	3	6	**9**	6	3	0

Lemma 4. Given a bipartite graph $G = (U, V, E)$ with $U = \{u_1, u_2, \ldots, u_n\}$ ordered in non-increasing order of $d_{u_i}(V)$, the size of the maximum biclique on G is bounded by $\max_{i \in [1,|U|]}\{i \times d_{u_i}(V)\}$.

Proof. Assume for contradiction a biclique $G(S_U, S_V)$ in G exceeds this bound. Let x be the value that maximizes $x \times d_{u_x}(V)\}$. Then, $G(S_U, S_V)$ must satisfy either $|S_U| = x$ and $|S_V| > d_{u_x}(V)$ or $|S_V| = d_{u_x}(V)$ and $|S_U| > x$. In either case, a contradiction arises, as there must exist a vertex u_x (or u_{x+1}) in S_U that is non-adjacent to at least one vertex in S_V, violating the definition of a biclique. Thus, no biclique in G has a size that exceeds our bounds. ◻

We further refine Lemma 4 by analyzing the balance of excluded non-neighbors. Specifically, for a biclique $G(S_U, S_V)$ in G, vertices in $V \setminus S_V$ must account for all non-neighbors of S_U, i.e., $\sum_{u \in S_U} \overline{d}_u(V) \leq \sum_{v \in V \setminus S_V} \overline{d}_v(U)$. This holds because each non-neighbor $v \in V$ of a vertex $u \in S_U$ must be excluded in S_V. Otherwise, there would exist a vertex $v \in S_V$ with $\overline{d}_v(S_U) \geq 1$, contradicting the definition of the biclique. By incorporating this constraint into Lemma 4, we establish the following tighter upper bound.

Lemma 5. Given a bipartite graph $G = (U, V, E)$ with $U = \{u_1, u_2, \ldots, u_n\}$ and $V = \{v_1, v_2, \ldots, v_n\}$ ordered in non-increasing order of $d_{u_i}(V)$ and $d_{v_i}(U)$, respectively, the size of the maximum biclique in G is at most $\max_{i \in [1,|U|]}\{i \times \min(d_{u_i}(V), j - 1)\}$, where j is the largest integer satisfying $\sum_{i'=1}^{i} \overline{d}_{u_{i'}}(V) \leq \sum_{j'=j}^{|V|} \overline{d}_{v_{j'}}(U)$.

Following Lemma 4 and earlier observations, Lemma 5 holds directly, thus we omit its proof. An example illustrating the effectiveness of the proposed upper bounds is given below.

Example 3. Consider again the bipartite graph G in Fig.1(a). The computed upper bounds are summarized in Table 2, where the rows labeled d, ub_1, and ub_2 correspond to the degree, and the upper bounds derived from Lemma 4 and Lemma 5, respectively. As shown, ub_2 closely approximates the actual size of the maximum biclique in G. This indicates that the proposed upper bounds offer a more precise estimation and enhance the efficiency of pruning strategies.

The upper bound-based algorithm. Building upon Lemma 5, we develop an algorithm that computes an upper bound on the maximum biclique size in a bipartite graph G, as outlined in Algorithm 1. The algorithm begins by ordering the vertices of U and V in non-descending order of their degrees (lines 1–2). It then iterates over the vertices in U to determine the upper bound ω_{ub} (lines 3–12), initializing necessary variables for each iteration (lines 3). Specifically, when processing vertex $v_i \in U$, an initial upper bound is derived from $i \times d_{v_i}(V)$ per Lemma 4 (line 6). Then, this upper bound is subsequently refined following the approach of Lemma 5 (lines 7–12). Ultimately, the maximum ω_{ub} observed throughout the iterations is reported as the final upper bound (line 13).

Theorem 3.5. *Algorithm 1 operates in linear time.*

PROOF. Lines 1–2 sort the vertices in U and V using bin sort in $O(|U|+|V|)$ time. Within the for-loop (lines 5–11), the iterations at line 9 are bounded by $O(|V|)$, so the loop also executes in $O(|U|+|V|)$ time. Given that $|U|+|V| \leq n$, the overall time complexity is $O(n)$. □

3.4 Implementation of the Search Framework

Building on previously introduced techniques, we propose a novel algorithm for finding the maximum biclique in a bipartite graph G, as outlined in Algorithm 2. The algorithm initializes the current maximum biclique H as an empty set with size $\omega = 0$ (line 1). It then invokes the recursive procedure *Branch*, implemented by the proposed pivot-based branching framework (line 2). The recursion operates on a subgraph $G(S_U \cup C_U, S_V \cup C_V)$ initialized with $S_U = \emptyset$, $S_V = \emptyset$, $C_U = U$, and $C_V = V$, respectively, where S_U and S_V consist of vertices that the maximum biclique must contain, C_U and C_V are candidate sets of vertices such that for each $u \in C_U$ (and $v \in C_V$), $S_V \subseteq N_u(G)$ (and $S_U \subseteq N_v(G)$) holds during recursive exploration. The parameter x denotes the number of non-edges in $G(S_U, S_V)$.

The *Branch* procedure begins by verifying whether the current candidate set C_U or C_V is empty. If either set is empty, it terminates the current branch and updates H and ω via Theorem 3.4 (lines 4-5). Otherwise, candidate sets are pruned first (lines 6–11): vertices in C_U (or C_V) with no non-neighbors in C_V (or C_U) are directly added to $G(S_U, S_V)$ by Lemma 1 (lines 7, 10), and vertices with degrees below the thresholds τ_V (and τ_U) are removed (line 8, 11). When such pruning expands $G(S_U, S_V)$, the algorithm evaluates whether the resulting subgraphs $G(S_U, S_V \cup C_V)$ and $G(S_U \cup C_U, S_V)$ could surpass the current biclique, as per Theorem 3.4 (line 12). Subsequently, the algorithm employs the upper bound algorithm (Algorithm 1) to eliminate branches that cannot improve the solution. A pivot-based branch-and-bound strategy is then applied following Theorem 3.3: the pivot vertex u is selected from C_U or C_V based on the minimal number of non-neighbors. If $u \in C_U$, a vertex w is chosen from its non-neighbors $D = \overline{N}_u(C_V)$ of u. The set $\{u, w\}$, along with vertices in $D \setminus \{w\}$ that have neighbors in $\overline{N}_w(C_U)$, is used to expand $G(S_U, S_V)$ according to the branching strategy. In the special case where $\overline{d}_u(C_V) = 1$ and $\overline{d}_w(C_U) = 1$, both u and w are added to $G(S_U, S_V)$, and the recursion continues.

Algorithm 2: The pivot-based search algorithm

Input: The graph $G = (U, V, E)$, the size constraints τ_U and τ_V
Output: The maximum biclique H and its size ω

1 $H \leftarrow G(\emptyset, \emptyset)$; $\omega \leftarrow 0$;
2 $Branch(\emptyset, \emptyset, 0, U, V)$;
3 **Function:** $Branch(S_U, S_V, x, C_U, C_V)$
4   **if** $C_U = \emptyset$ *or* $C_V = \emptyset$ **then**
5   Compute H and ω with Theorem 3.4 and **return**;
6   **foreach** $u \in C_U$ **do**
7   **if** $\overline{d}_u(C_V) = 0$ **then** Move u from C_U to S_U;
8   **if** $d_u(C_V) + |S_V| < \tau_V$ **then** $C_U \leftarrow C_U \setminus \{u\}$;
9   **foreach** $v \in C_V$ **do**
10   **if** $\overline{d}_v(C_U) = 0$ **then** Move v from C_V to S_V;
11   **if** $d_v(C_U) + |S_U| < \tau_U$ **then** $C_V \leftarrow C_V \setminus \{v\}$;
12   Update H and ω in $G(S_U, S_V \cup C_V)$ and $G(S_U \cup C_U, S_V)$ based on Theorem 3.4;
13   **if** $UB(G, S_U \cup C_U, S_V \cup C_V) \leq \omega$ **then return**;
14   $u \leftarrow \arg\min_{u \in C_U} \overline{d}_u(C_V)$; $v \leftarrow \arg\min_{v \in C_V} \overline{d}_v(C_U)$;
15   **if** $\overline{d}_u(C_V) \leq \overline{d}_v(C_U)$ **then**
16     $D \leftarrow C_V \setminus N_u(G)$; $w \leftarrow \arg\max_{w \in D} d_w(C_U)$;
17   **foreach** $w' \in D \setminus \{w\}$ *s.t.* $\overline{N}_w(C_U) \cap N_{w'}(G) \neq \emptyset$ **do**
18       $C_V \leftarrow C_V \setminus \{w'\}$; $C'_U \leftarrow N_{w'}(C_U)$;
19   **if** $|\overline{N}_w(C_U) \cap N_{w'}(G)| > 1$ **then**
20         $Branch(S_U, S_V \cup \{w'\}, x, C'_U, C_V)$;
21   **else**
22         $u' \leftarrow$ the only vertex in $\overline{N}_w(C_U) \cap N_{w'}(G)$;
23         $Branch(S_U \cup \{u'\}, S_V \cup \{w'\}, x, C'_U, N_{u'}(C_V))$;
24   **if** $\overline{d}_w(C_U) = 1$ **then**
25       $Branch(S_U \cup \{u\}, S_V \cup \{w\}, x+1, C_U \setminus \{u\}, C_V \setminus \{w\})$;
26   **else**
27       $Branch(S_U, S_V \cup \{w\}, x, N_w(C_U), C_V \setminus \{w\})$;
28       $Branch(S_U \cup \{u\}, S_V, x, C_U \setminus \{u\}, C_V \setminus D)$;
29   **else**
30   Branch strategies in lines 16-28 using the pivot vertex v;

THEOREM 3.6. *The time complexity of Algorithm 2 is* $O(m \cdot 1.348^n)$.

PROOF. We express the overall complexity as $O(P(n) \times T(n))$, where $P(n)$ denotes the time per recursive call and $T(n)$ represents the total number of leaves in the recursion tree. Since each recursive call requires $O(m)$ time to compute vertex degrees, we have $P(n) = O(m)$, and it suffices to bound $T(n)$. Suppose the vertex u in C_U is selected as the pivot vertex, and let $\overline{d}$ denote the number of non-neighbors of u in C_V. For each vertex $v \in \overline{N}_u(C_V)$, we have $\overline{d}_v(C_U) \geq \overline{d}$. Then the branching yields the recurrence:

$$T(n) \leq T(n - \overline{d} - 1) + \sum_{i=1}^{\overline{d}} T(n - i - \overline{d}), \tag{1}$$

where $T(n - \overline{d} - 1)$ corresponds to the branch that includes u in $G(S_U, S_V)$ and $T(n - i - \overline{d})$ corresponds to the branch that includes the i-th vertex from $\overline{N}_u(C_V)$.

We now analyze several specific cases for $\overline{d}$:

(1) Case $\overline{d} = 1$: let v be the only vertex in $\overline{N}_u(C_V)$. If $\overline{d}_v(C_U) = 1$, we obtain $T(n) = T(n-1)$ by Theorem 3.4 (lines 24-25). Otherwise, if $\overline{d}_v(C_U) \geq 2$, we have: $T(n) \leq T(n-2) + T(n-3)$.

(2) Case $\overline{d} = 2$: for each $v \in \overline{N}_u(C_V)$, we have $\overline{d}_v(C_U) \geq 2$. Suppose there exists a vertex $v \in \overline{N}_u(C_V)$ with $\overline{d}_v(C_U) = 2$. Let w be the other vertex in $\overline{N}_u(C_V) \setminus \{v\}$ with $|\overline{N}_v(C_U) \cap N_w(C_U)| = 1$. Theorem 3.3 (lines 21–23) yields $T(n) \leq T(n-3) + T(n-6) + T(n-4)$, where $T(n-6)$ corresponds to the branch including both w and v. If every v in $\overline{N}_u(C_V)$ satisfies $\overline{d}_v(C_U) \geq 3$, then:

$$T(n) \leq T(n-3) + T(n-4) + T(n-5). \tag{2}$$

(3) Case $\overline{d} \geq 3$: the worst-case recurrence becomes:

$$T(n) \leq T(n-4) + T(n-4) + T(n-5) + T(n-6). \tag{3}$$

This is because when $\overline{d} = 3, 4, 5$, the recurrence in Eq. (1) is bounded by Eq. (3). If $\overline{d} \geq 6$, we derive that: $T(n) \leq (\overline{d} + 1)T(n - \overline{d} - 1) \leq (\overline{d} + 1)^{\frac{n}{\overline{d}+1}} T(1) \leq O(1.321^n)$.

Therefore, Eq. (3) yields the worst-case recurrence for Algorithm 2. Assuming $T(n) = O(x^n)$, the value of x corresponds to the largest real root of $x^n = 2x^{n-4} + x^{n-5} + x^{n-6}$. Solving this yields $x \leq 1.348$, thereby, establishing the time complexity of $O(m \cdot 1.348^n)$. □

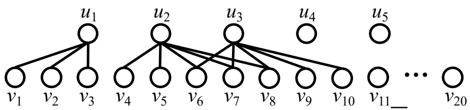


Figure 3: The complement bipartite graph $\overline{G} = (U, V, \overline{E})$ of G with $|U| = 5$ and $|V| = 20$, where vertices in $\{v_4, v_5\}$ and $\{u_{11}, ..., u_{20}\}$ are the isolated vertices.

4 THE COVER-BASED SEARCH ALGORITHM

In this section, we present a vertex cover-based algorithm for processing dense bipartite graphs, grounded in a novel duality between bicliques in G and vertex covers in its complement $\overline{G}$. The algorithm guarantees a worst-case time complexity of $O(m \cdot 1.381^n)$, which, while marginally higher than that of Algorithm 2 (Sec.3), demonstrates superior empirical performance on large and dense bipartite graphs (see Sec.6). This efficiency gain arises from the structural sparsity of the complement graph and tightly integrated upper-bound pruning strategies, both of which contribute to the elimination of redundant search branches. We now formalize the theoretical relationship between bicliques and vertex covers in bipartite complements, and then introduce our proposed algorithms.

4.1 Biclique V.S. Bipartite Graph Cover

Given a bipartite graph $G = (U, V, E)$, we define the concept of a bipartite vertex cover as follows.

Definition 2. Given a bipartite graph $G = (U, V, E)$, a pair of subsets (S_U, S_V), where $S_U \subseteq U$ and $S_V \subseteq V$, is a vertex cover of G if every edge $(u, v) \in E$ satisfies $u \in S_U$ or $v \in S_V$.

The pair (S_U, S_V) is called a *minimum vertex cover* of G if $|S_U \cup S_V|$ is minimized over all such covers. For comparison, in a general (non-bipartite) graph $G = (V, E)$, a vertex cover is a subset $C \subseteq V$ such that every edge has at least one endpoint in C. Prior work [44] establishes a tight relationship between the maximum clique and the minimum vertex cover in general graphs, as summarized below.

Lemma 6. [44] Given an arbitrary (non-bipartite) graph $G = (V, E)$, let C be a minimum vertex cover in the complement graph $\overline{G}$. Then, $G(V \setminus C)$ forms a maximum clique in G.

However, we observe that Lemma 6 does not hold when extended to the bipartite graphs. Figure 3 provides a counterexample illustrating this limitation. Consider the complement graph $\overline{G}$ shown in Figure 3. The subgraph $G(\{u_1, u_4, u_5\}, \{v_4, \dots, v_{20}\})$, which contains $51 = 3 \times 17$ edges, is the maximum biclique of the original bipartite graph G. However, the minimum vertex cover of $\overline{G}$ is the pair $(S_U = \{u_1, u_2, u_3\}, S_V = \emptyset)$, and the subgraph induced by $(U \setminus S_U, V \setminus S_V)$ does not correspond to the maximum biclique in G. Furthermore, this discrepancy persists when considering the problem of finding a maximum biclique subject to size constraints. Taking the same complement graph $\overline{G}$ and assuming $\tau_U = \tau_V = 4$, we observe that the pair (S_U, S_V) with $|S_U| = 1$ and $|S_V| = 8$ forms a minimum vertex cover of $\overline{G}$ that induces a biclique $G(U \setminus S_U, V \setminus S_V)$ satisfying the constraints $|U \setminus S_U| \geq \tau_U$ and $|V \setminus S_V| \geq \tau_V$. However, a larger vertex cover (S_U, S_V) with $|S_U| = 0$ and $|S_V| = 10$ yields an even larger biclique in G. These observations indicate that Lemma 6 does not generalize to bipartite graphs. Therefore, it is necessary to establish new structural relationships between bicliques in G and vertex covers in its complement $\overline{G}$.

Novel duality relationships. Given a vertex cover (S_U, S_V) of the complement $\overline{G}$, we say that (S_U, S_V) is a *minimal vertex cover* if it covers all edges of $\overline{G}$, and there exist no proper subsets $S'_U \subseteq S_U$ and $S'_V \subseteq S_V$ such that (S'_U, S'_V) also forms a vertex cover of $\overline{G}$. We now can establish a novel relationship between maximal bicliques in G and minimal vertex covers in $\overline{G}$.

Lemma 7. Given a bipartite graph $G = (U, V, E)$, let $G(S_U, S_V)$ be a maximal biclique in G such that $|S_U| \geq \tau_U$ and $|S_V| \geq \tau_V$. Then, $(U \setminus S_U, V \setminus S_V)$ forms a minimal vertex cover of $\overline{G}$.

Proof. Since $G(S_U, S_V)$ is the maximal biclique of G, we obtain that $(U \setminus S_U, V \setminus S_V)$ is a vertex cover in $\overline{G}$. Moreover, there is no vertex $u \in U \setminus S_U$ (or $v \in V \setminus S_V$) such that $(U \setminus S_U \setminus \{u\}, V \setminus S_V)$ (or $(U \setminus S_U, V \setminus S_V \setminus \{v\})$) is a vertex cover in $\overline{G}$, or u (or v) is available for expanding the maximal biclique $G(S_U, S_V)$. Thus, $(U \setminus S_U, V \setminus S_V)$ is the minimal vertex cover in $\overline{G}$. We complete this proof. □

Based on Lemma 7, we derive a symmetric result: let (S_U, S_V) be a minimal vertex cover of $\overline{G}$ such that $|S_U| \leq |U| - \tau_U$ and $|S_V| \leq |V| - \tau_V$. Then, $G(U \setminus S_U, V \setminus S_V)$ constitutes the maximal biclique of G with $|U \setminus S_U| \geq \tau_U$ and $|V \setminus S_V| \geq \tau_V$. This observation leads to the following duality theorem, which reformulates the maximum biclique search problem.

Theorem 4.1. *Given a bipartite graph $G = (U, V, E)$, the problem of finding the maximum edge biclique in G such that each side has cardinality no less than τ_U and τ_V is equivalent to the problem of finding a minimal vertex cover (S_U, S_V) of $\overline{G}$ such that $|U \setminus S_U| \times |V \setminus S_V|$ is maximized, subject to the constraints $|S_U| \leq \alpha_U$ and $|S_V| \leq \alpha_V$, where $\alpha_U + \tau_U = |U|$, $\alpha_V + \tau_V = |V|$, and $\tau_U > 0, \tau_V > 0$.*

It is important to note that the solutions developed in Sec. 3 cannot be used to solve the problem of minimal vertex cover on $\overline{G}$, since the minimal vertex cover and the maximum biclique are two distinct concepts based on their definitions. Moreover, it is easy to verify that Theorems 3.1-3.3 no longer apply to minimal vertex cover on $\overline{G}$, and therefore the algorithms proposed based on these theorems clearly cannot solve the minimal vertex cover search problem. To address this new problem, we will develop several new techniques in subsequent sections.

4.2 Cover-based Branching Rules

Let $B = (\overline{G}, S_U, S_V, C_U, C_V)$ denote an instance for computing the minimal vertex covers of $\overline{G}$, where S_U and S_V represent the partial vertex cover of $\overline{G}$ (i.e., every minimal vertex cover must include S_U and S_V), C_U and C_V induce a subgraph $G(C_U, C_V)$ to cover. The problem can be addressed via a branch-and-bound framework that leverages a key property: any minimal vertex cover of $\overline{G}$ must either include a given vertex u or all of its neighbors in $\overline{G}$. Formally, consider a vertex $u \in C_U$ (or analogously, $v \in C_V$). The instance B is then branched into two subinstances: one where u is added to the cover, and the other where all of its neighbors in C_V (with respect to $\overline{G}$) are included. Thus, this minimal vertex cover problem in $\overline{G}$ can be resolved by recursively applying this branching rule to the initial instance $B_0 = (\overline{G}, \emptyset, \emptyset, U, V)$ until each of the subinstances reaches a trivial state (e.g., C_U or C_V becomes empty).

For convenience, we refer to a vertex u as the *branching vertex* if it is used to partition the instance $B = (\overline{G}, S_U, S_V, C_U, C_V)$ into multiple subinstances. To enhance the practical performance, we observe that selecting the vertex u with the highest degree in the subgraph $\overline{G}(C_U, C_V)$ as the branching vertex in B yields notable performance improvements. The rationale is that a branching vertex with a larger degree enables more substantial pruning of the search space in the subinstance B', where all neighbors of u (in $\overline{G}$) are included in the vertex cover. By integrating Theorem 4.1, we derive a cover-based branching technique for the maximum biclique search.

The proposed cover-based branching rule. Given an instance $I = (G, S_U, S_V, C_U, C_V)$ to find the maximum biclique of G containing S_U and S_R, we reduce I to a minimal vertex cover problem $B = (\overline{G}, R_U, R_V, C_U, C_V)$ on the complement graph $\overline{G}$ of G, where $R_U = U \setminus (S_U \cup C_U)$ and $R_V = V \setminus (S_U \cup C_U)$. The branch-and-bound procedure to solve B is defined as follows.

(1) If $\overline{G}(C_U, C_V)$ is edgeless and the instance B satisfies $|R_U| \le \alpha_U$ and $|R_V| \le \alpha_V$, terminate the recursion and update the current maximum biclique with $G(U \setminus S_U, V \setminus S_V)$.
(2) Otherwise, select a vertex u from $C_U \cup C_V$ that has the largest degree in $\overline{G}(C_U, C_V)$ (without loss of generality, $u \in C_U$).
(3) Generate two subinstances with the branching vertex u: $B_1 = (\overline{G}, R_U \cup \{u\}, R_V, C_U \setminus \{u\}, C_V)$, which includes u in the vertex cover, and $B_2 = (\overline{G}, R_U, R_V \cup (C_V \cap N_u(\overline{G})), C_U \setminus \{u\}, C_V \setminus N_u(\overline{G}))$, which excludes u but includes its all neighbors in C_V.
(4) Recursively apply steps (1)-(3) on each subinstance B_1 and B_2 until C_U or C_V becomes empty.

To further enhance the efficiency of our vertex cover–based branching process, we introduce three supplementary branch reduction techniques, which are delineated below.

Branch reduction for $\max_{u \in U} d_u(\overline{G}) \le 1$ **or** $\max_{v \in V} d_v(\overline{G}) \le 1$. In this case, the maximum biclique of G can be computed in polynomial time. Without loss of generality, assume that $\max_{u \in U} d_u(\overline{G}) \le 1$ for the complement graph $\overline{G} = (U, V, \overline{E})$. Define $C_U = \{u \in U \mid d_u(\overline{G}) = 1\}$ and $C_V = \{v \in V \mid d_v(\overline{G}) \ge 1\}$. Then, any pair of subsets (R_U, R_V) satisfying $R_U = C_U \setminus (\cup_{v \in R_V} N_v(\overline{G}))$ and $R_V \subseteq C_V$ forms a minimal vertex cover of $\overline{G}$; the corresponding biclique in G is induced by $G(U \setminus R_U, V \setminus R_V)$. Based on this analysis, identifying a minimal vertex cover (R^*_U, R^*_V) of $\overline{G}$ that maximizes $|U \setminus R^*_U| \times |V \setminus R^*_V|$ directly yields the maximum biclique of G. We formalize this observation in the following lemma.

Lemma 8. Given $\overline{G} = (U, V, \overline{E})$, where $d_u(\overline{G}) \le 1$ for all $u \in U$, let $C = \{v \in V | d_v(\overline{G}) \ge 1\} = \{v_1, v_2, \cdots, v_x\} \subseteq V$ ordered in non-decreasing order of $d_{v_i}(\overline{G})$. Then, the size of maximum biclique of G is given by $\max_{i \in [0,|C|]}\{(|V \setminus C| + i) \times (|U| - \sum_{j=1}^{i} d_{v_i}(U))\}$ subject to constraints $|U| - \sum_{j=1}^{i} d_{v_i}(U) \ge \tau_U$ and $|V \setminus C| + i \ge \tau_V$.

Proof. For a minimal vertex cover (R_U, R_V) of $\overline{G}$, if $|R_V| = |C| - i$, then R_U must cover remaining $\sum_{j=1}^{i} d_{v_i}(U)$ non-edges. Consequently, $|U \setminus R_U| = |U| - \sum_{j=1}^{i} d_{v_i}(U)$ and $|V \setminus R_V| = |V \setminus C| + i$. The product of these terms gives the biclique size in G. Maximizing over all $i \in [0, |C|]$ under the constraints τ_U and τ_V yields the result. □

Branch Reduction $\max_{u \in U} d_u(\overline{G}) \le 2$ **and** $\max_{v \in V} d_v(\overline{G}) \le 2$. When $\overline{G}$ is connected and forms a cycle or a path, its minimal vertex covers can be computed directly. Let $\overline{d}_{max} = \max_{u \in U \cup V} d_u(\overline{G})$. We derive the following results.

Lemma 9. Let $\overline{G} = (U, V, \overline{E})$ be the complement of G with $\overline{d}_{max} \le 2$. Suppose $|U| \le |V|$. Then, the size of the maximum biclique of G is given by $\max\{(|U| - x)(x - 1)\}$ with $x \in [|V| + 1 - \alpha_V, \alpha_U] \cap \mathbb{N}$ if $\overline{G}$ forms a single cycle, or by $\max\{(|U| - x)x\}$ with $x \in [|V| - \alpha_V, \alpha_U] \cap \mathbb{N}$ if $\overline{G}$ forms a single path.

Proof. The maximum biclique of G is produced by the minimal vertex cover (R_U, R_V) of $\overline{G}$ that maximizes $|U \setminus R_U| \times |V \setminus R_V|$. If $\overline{G}$ is a cycle, then $|U| = |V|$, and (R_U, R_V) has total size either $|V|$ or $|V| + 1$. The optimum is achieved when $|R_U| + |R_V| = |V| + 1$, so that the biclique has size $(|U| - x)(x - 1)$, where $x \in [0, |V| + 1]$. In contrast, if $\overline{G}$ is a path then either $|U| = |V|$ or $|U| = |V| - 1$, and in both cases the optimum is achieved when $|R_U| + |R_V| = |V|$, giving a biclique of size $(|U| - x)x$, where $x \in [0, |V|]$. Finally, incorporating the constraints α_U and α_V imposes that $x \in [|V| + 1 - \alpha_V, \alpha_U] \cap \mathbb{N}$ in the cycle case and $x \in [|V| - \alpha_V, \alpha_U] \cap \mathbb{N}$ in the path case. This completes the proof. □

Note that if the complement graph $\overline{G}$ is not connected and consists of multiple cycles or paths, Lemma 9 no longer applies and becomes difficult to modify. In this case, we address the challenge using our branch-and-bound rules with a slight modification in the selection of branching vertices.

- Let S denote the set of vertices in U (or V) that have degree exactly two in $\overline{G} = (U, V, \overline{E})$. Among these, we select the vertex $u \in S$ that minimizes $|N_{v_1}(\overline{G}) \cap N_{v_2}(\overline{G})|$ as the branching vertex, where v_1 and v_2 are the two neighbors of u in $\overline{G}$.

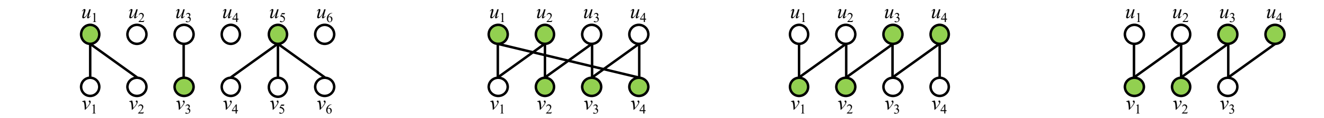
The advantage of this strategy is that if $|N_{v_1}(\overline{G}) \cap N_{v_2}(\overline{G}) \setminus| \le 2$, either the vertex v_1 (or v_2) is isolated in the subinstance $B_1 = (\overline{G}, \{u\}, \emptyset, U \setminus \{u\}, V)$, or the vertex $u' \in N_{v_1}(\overline{G}) \cap N_{v_2}(\overline{G}) \setminus \{u\}$ that is isolated in the subinstance $B_2 = (\overline{G}, \emptyset, \{v_1, v_2\}, U \setminus \{u\}, V \setminus \{v_1, v_2\})$. In both cases, the search space is reduced because these isolated vertices need not be considered for further branching.

Example 4. Figure 4 illustrates the application of our cover-based branch reductions. As shown in Fig.4(a), if the bipartite graph satisfies $\max_{v \in V}(\overline{d}_v(G)) \le 1$, Lemma 8 applied to directly determine the maximum biclique size of G, yielding a minimal vertex cover $(R_U = \{u_1, u_5\}, R_V = \{v_1\})$. Moreover, when $\overline{G}$ forms a cycle or a path, the corresponding minimal vertex cover size for the maximum biclique is $|U| + 1$ or $|U|$, respectively, based on Lemma 9. This is consistent with the cases in Fig. 4(b-d), where the green vertices indicate the minimal vertex covers of G that yield the maximum bicliques.

Branch reduction for $u \in U$ **with** $d_u(\overline{G}) > \alpha_V$ **(or** $v \in V$ **with** $d_v(\overline{G}) > \alpha_U$**).** To compute the minimal vertex cover of the complement graph $\overline{G}$ to have size no larger than α_U and α_V, we notice that a vertex u must be included in the minimal vertex cover if it satisfies $d_u(\overline{G}) > \alpha_V$. This is because if u is excluded, all of its neighbors in $\overline{G}$ would have to be included, causing the cover to exceed the allowed size. Formally, we have the following lemma.

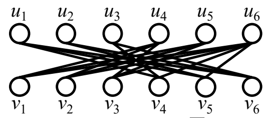
Lemma 10. Given a complement graph $\overline{G} = (U, V, \overline{E})$ of G, the maximum biclique of G, subject to constraints τ_U and τ_V, must be included in the subinstance $B = (\overline{G}, S_U, S_V, U \setminus S_U, V \setminus S_V)$, where $S_U = \{u \in U | d_u(\overline{G}) > \alpha_V\}$, $S_V = \{v \in V | d_v(\overline{G}) > \alpha_U\}$, $\tau_U + \alpha_U = |U|$, and $\tau_V + \alpha_V = |V|$.

Proof. Let (R_U, R_V) be any minimal vertex cover of $\overline{G}$ with $S_U \nsubseteq R_U$ (or $S_V \nsubseteq R_V$). Then there exists some $u \in S_U \setminus R_U$. For (R_U, R_V) to cover $\overline{G}$ minimally, all neighbors of u must be included in R_V. Since $d_u(\overline{G}) > \alpha_V$, it follows that $|R_V| \ge \alpha_U$, thus violating



(a) The bipartite graph $\overline{G}$ with $\max_{v \in V}(\overline{d}_v(G)) \leq 1$ (b) A circle $\overline{G}$ (c) A path $\overline{G}$ with $|U| = |V|$ (d) A path $\overline{G}$ with $|U| \neq |V|$

Figure 4: An illustrative example of cover-based branch reductions in Lemma 8 and Lemma 9, where the green vertices are the minimal vertex covers of G that produces the maximum bicliques.



(a) The complement graph $\overline{G}$ in Fig. 1(a)

	$\sum_{v \in U^i \cup V^j} d_v(\overline{G}) \geq \lvert\overline{E}\rvert$						
$\lvert U^i\rvert$	0	1	2	3	4	5	6
$\lvert V^j\rvert$	6	5	4	3	2	1	0
ub	0	5	8	**9**	8	5	0

(b) The upper bounds of $\overline{G}$

Figure 5: The upper bounds derived by Lemma 12.

the size constraint. Such a result must necessarily not exist, which completes the proof. □

4.3 Cover-based Upper Bounds

We develop several upper bound techniques to early terminate the vertex cover-based branch-and-bound process for finding the maximum biclique of G. The main idea of our upper bounds is derived from a newly-established relationship between the lower bound on the size of any minimal vertex cover of $\overline{G}$ and size of the maximum biclique of G. The underlying relationship between these two bounds is formalized as follows.

Lemma 11. Given a bipartite graph G, let x be a positive value such that for any minimal vertex cover (R_U, R_V) in $\overline{G}$, we have $x \leq |R_U| + |R_V|$. Then, the size of the maximum biclique of G is no larger than $\max_{i \in [0,x]}\{(|U| - i) \times (|V| - x + i)\}$.

Proof. Suppose there exists a biclique (S_U, S_V) in G such that $|S_U| + |S_V|$ exceeding this upper bound. Then the corresponding minimal vertex cover (R_U, R_V) satisfies $|R_U| + |R_V| > |U| + |V| - x$, contradicting the fact that $x \leq |R_U| + |R_V|$. Hence, the maximum biclique size cannot exceed the stated bound. □

Subject to the results of Lemma 11, we focus on deriving a lower bound for the minimal vertex covers in $\overline{G}$, based on the following observation: if (R_U, R_V) is a minimal vertex cover of $\overline{G}$, then $\sum_{u \in R_U} d_u(\overline{G}) + \sum_{v \in R_V} d_v(\overline{G}) \geq |\overline{E}|$ following the definition of the vertex cover. If we can find subsets (S_U, S_V) vertices in $\overline{G}$ with the largest degrees such that $\sum_{u \in S_U} d_u(\overline{G}) + \sum_{v \in S_V} d_v(\overline{G}) \leq |\overline{E}|$, then $|S_U| + |S_V|$ must serve as a lower bound for all vertex covers of G. Based on this, we establish the following lemma for determining an upper bound on the size of the maximum biclique in G.

Lemma 12. Given a complement graph $\overline{G} = (U, V, \overline{E})$ of G, let U^i (and V^i) be the first i vertices in U (and V) with the largest degree in V (and U), respectively. Then, the size of the maximum biclique in G is bounded by $\max_{i \in [1,|U|]}\{(|U| - i) \times (|V| - j)\}$, where j is the smallest integer such that $\sum_{v \in U^i} d_v(\overline{G}) + \sum_{u \in V^j} d_u(\overline{G}) \geq |\overline{E}|$.

Proof. It is straightforward to verify that $|U^i| + |V^j|$ provides a lower bound for the minimal vertex cover of $\overline{G}$ containing exactly i vertices from U, thus the desired result follows. □

The cover-based upper bound algorithm. Based on Lemma 12, we propose an efficient algorithm, outlined in Algorithm 3, to compute an upper bound on the maximum biclique size of G, thereby

---

Algorithm 3: $UBC(\overline{G}, U, V)$

1 Sort each vertex u in U with non-increasing order of $d_v(\overline{G})$;
2 Sort each vertex v in V with non-increasing order of $d_u(\overline{G})$;
3 Let u_i and v_j be the i-th and j-th vertex in U and V, respectively;
4 $e_U = 0$; $e_V = |\overline{E}|$; $j = |V|$; $\omega_{ub} \leftarrow |U| \times (|V| - j)$;
5 **foreach** i *in* 1 *to* $|U|$ **do**
6 $\quad e_U \leftarrow e_U + d_{u_i}(\overline{G})$;
7 $\quad$ **while** $j > 0$ *and* $e_U + e_V - d_{v_j}(\overline{G}) \geq |\overline{E}|$ **do**
8 $\quad\quad e_V \leftarrow e_V - d_{v_j}(\overline{G})$; $j \leftarrow j - 1$;
9 $\quad$ **if** $i \leq \alpha_U$ *and* $j \leq \alpha_V$ **then**
10 $\quad\quad \omega_{ub} \leftarrow \max(\omega_{ub}, (|U| - i) \times (|V| - j))$
11 **return** ω_{ub};

---

reducing redundant computations during the branch-and-bound process. The core of the algorithm lies in its iterative loop (lines 5–10). Specifically, it initializes the upper bound ω_{ub} using $i = 0$ and $j = |V|$ (lines 3–4), and then iteratively increases i from 1 to $|U|$, deriving the corresponding minimal j such that the constraint in Lemma 12 is satisfied. At each step, it updates ω_{ub} accordingly (lines 9-10). Finally, the algorithm returns the largest value of ω_{ub} as the computed upper bound (line 11).

Example 5. Consider again the bipartite graph G depicted in Fig. 1(a), with its complement graph $\overline{G}$ shown in Fig. 5(a). We iteratively expand the subset $U^i \subseteq U$ from size 0 to $|U|$ and compute the corresponding subset $V^j \subseteq V$ via a constraint $\sum_{u \in U^i} d_u(\overline{G}) + \sum_{v \in V^j} d_v(\overline{G}) \geq |\overline{E}|$. Fig. 5(b) presents the results throughout this process. As illustrated, the derived upper bounds closely match the true maximum biclique size of G, validating the pruning efficacy of Lemma 12.

Theorem 4.2. *Algorithm 3 operates in linear time.*

Proof. Lines 1-2 takes $O(|U| + |V|)$ time via bin sort. In for-loop (lines 5–10), it performs line 8 a total number of $O(|V|)$ iterations, resulting in the overall time complexity of lines 6–10 being bounded by $O(|U| + |V|)$. Thus, the total complexity is $O(n)$. □

4.4 Implementation Details

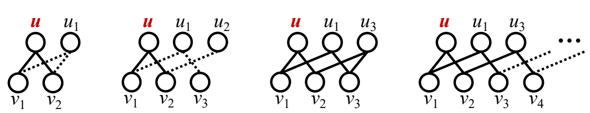
We now present a vertex cover-based branch-and-bound algorithm for computing the maximum biclique of G, incorporating the proposed techniques. The pseudocode is detailed in Algorithm 4.

Algorithm 4 begins by constructing the complement graph $\overline{G}$ from G, initializing $\alpha_U = |U| - \tau_U$, $\alpha_V = |V| - \tau_V$, and sets the current maximum biclique size ω to 0 (lines 1–2). It then calls *CBranch*, which recursively searches for the maximum biclique of G using the cover-based strategy (line 3). This procedure takes four parameters: S_U, S_V, C_U, and C_V, initialized as $S_U = \emptyset$, $S_V = \emptyset$, $C_U = U$, and $C_V = V$. Here, (S_U, S_V) denotes the partial vertex cover, and $\overline{G}(C_U, C_V)$ is the remaining subgraph to cover.

Algorithm 4: The cover-based search algorithm

Input: The graph $G = (U, V, E)$, the size constraints τ_U and τ_V
Output: The maximum biclique H of G and its size ω

1 Let $\overline{G}$ be the complement graph of G;
2 $\alpha_U = |U| - \tau_U$; $\alpha_V = |V| - \tau_V$; $H \leftarrow G(\emptyset, \emptyset)$; $\omega = 0$;
3 *CBranch*$(\emptyset, \emptyset, U, V)$;
4 **Function:** *CBranch*(S_U, S_V, C_U, C_V)
5 **if** $\overline{G}(C_U, C_V)$ *contains no edges* **then**
6 Update H and ω; **return**;
7 Move each $u \in C_U$ with $d_u(S_V \cup C_V) < \tau_V$ from C_U to S_U;
8 Move each $v \in C_V$ with $d_v(S_U \cup C_U) < \tau_U$ from C_V to S_V;
9 **if** $UBC(\overline{G}, S_U \cup C_U, S_V \cup C_V) \leq \omega$ **then return**;
10 Let $u \in C_U$ be a vertex with the largest degree in $\overline{G}(C_U, C_V)$;
11 Let $v \in C_V$ be a vertex with the largest degree in $\overline{G}(C_U, C_V)$;
12 **if** $\overline{d}_u(C_V) \leq 1$ *or* $\overline{d}_v(C_U) \leq 1$ **then**
13 Update H and ω with Lemma 8; **return**;
14 **if** $\overline{d}_u(C_V) \leq 2$ *and* $\overline{d}_v(C_U) \leq 2$ **then**
15 **if** $\overline{G}(C_U, C_V)$ *is a connected graph* **then**
16 Update H and ω with Lemma 9; **return**;
17 **else**
18 Reselect the branching vertex with the reduction rule;
19 **if** $d_u(C_V) \geq d_v(C_V)$ **then**
20 *CBranch*$(S_U \cup \{u\}, S_V, C_U \setminus \{u\}, C_V)$;
21 *CBranch*$(S_U, S_V \cup N_u(C_V), C_U \setminus \{u\}, C_V \setminus N_u(C_V))$;
22 **else**
23 *CBranch*$(S_U, S_V \cup \{v\}, C_U, C_V \setminus \{v\})$;
24 *CBranch*$(S_U \cup N_v(C_V), S_V, C_U \setminus N_v(C_V), C_V \setminus \{v\})$;



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Figure 6: Subgraphs when branching vertex u of degree 2 is selected, dashed lines indicate possible edges.

Within the *CBranch* procedure, if $\overline{G}(C_U, C_V)$ contains no edges, the procedure terminates the current call and updates ω based on (S_U, S_V) (lines 5–6). Otherwise, it proceeds with the branch-and-bound routine (lines 7-24). Firstly, it moves vertices in C_U (resp. C_V) with degrees exceeding α_V (resp. α_U) from C_U (resp. C_V) to S_U (resp. S_V), and then applies the proposed cover-based upper-bound pruning techniques (lines 7-9). Next, it selects a vertex with the highest degree in $\overline{G}(C_U, C_V)$ as the branching vertex and explores three scenarios. (1) If the branching vertex has degree at most 1, Lemma 8 is applied to directly determine the maximum biclique in $G(S_U \cup C_U, S_V \cup C_V)$ (lines 12-13). (2) If the branching vertex has degree at most 2, Lemma 9 is applied to compute the maximum biclique when $G(S_U \cup C_U, S_V \cup C_V)$ is connected (lines 15-16); otherwise, it enforces the branching vertex constraints based on the reduction technique in Sec 3.1 (lines 17-18). (3) In all other cases, the standard branching strategy is executed (lines 19–24). Finally, the process continues until all branches are resolved.

Theorem 4.3. *The time complexity of Algorithm 4 is $O(m \cdot 1.381^n)$.*

Proof. Algorithm 4 processes each recursive call in $O(m)$ time. Let $T(n)$ denote the total number of leaves in the recursion tree. We analyze $T(n)$ for various degrees of the branching vertex u (without loss of generality, $u \in C_U$).

(1) Case $d \leq 1$. The maximum biclique can be computed directly by Lemma 8, implying polynomial time complexity for this case.
(2) Case $d = 2$. If $\overline{G}(C_U, C_V)$ is connected, Lemma 9 applies. Otherwise, the recurrence is given by:
$$T(n) \leq T(n-1) + T(n-3), \tag{4}$$
where $T(n-3)$ is the branch that finds the minimal vertex cover containing $N_u(\overline{G})$. We note that this recurrence can be further tightened. Specifically, we notice that the branch vertex u can only appear in one of four graph types, as shown in Fig. 6. For graph types G_1 (or G_2), if $d_{v_1}(G_1) = 1$ or $d_{v_2}(G_1) = 1$, the subbranch $T(n-1)$ is reduced to $T(n-2)$, since v_1 or v_2 will become isolated after u is moved from C_U to S_U. On the other hand, if $d_{v_1}(G_1) = 2$ and $d_{v_2}(G_1) = 2$, the subbranch $T(n-3)$ can be reduced to $T(n-4)$, as vertex u_1 (or u_2) will be isolated in $\overline{G}(C_U, C_V \setminus N_u(\overline{G}))$. Hence, for graph types G_1 or G_2, the recurrence tightens to: $T(n) \leq T(n-1) + T(n-4)$.
For graph types G_3 (or G_4), the branch $T(n-1)$, where u is pushed into S_U, will select u_1 as the branching vertex, based on our branching strategy. Then, v_1 will become an isolated vertex in the subgraph $\overline{G}(C_U \setminus \{u\}, C_V)$ when u_1 is moved from C_U to S_U. This leads to the recurrence $T(n-1) \leq T(n-3) - T(n-4)$. Similarly, $T(n-3)$, which contains all neighbors of u in $\overline{G}$, can be bounded by $T(n-5) - T(n-6)$. Thus, for graph types G_3 (or G_4), the recurrence becomes: $T(n) \leq T(n-1) + T(n-3) \leq T(n-3) - T(n-4) + T(n-5) - T(n-6) \leq T(n-1) - T(n-4)$, and the improved recurrence for the case where $d = 2$ is:
$$T(n) \leq T(n-1) + T(n-4). \tag{5}$$
(3) $d \geq 3$. In this case, the recurrence is:
$$T(n) \leq T(n-1) + T(n-d-1), \tag{6}$$
where $T(n-d-1)$ represents the subbranch for finding the minimal vertex cover of $\overline{G}$ that includes all neighbors of u.

Following above analysis, Eq. (5) represents the final recurrence. Let $T(n) = O(x^n)$. This yields the equation $x^n = x^{n-1} + x^{n-4}$, whose real root is $x < 1.381$. Therefore, the time complexity of Algorithm 4 is bounded by $O(m \cdot 1.381^n)$. □

We also note that all proposed can solve special bipartite graphs that are themselves bicliques in linear time. Thus, the time taken for the proposed algorithms to process the real-world bipartite graphs lies between $O(n)$ and $O(m \cdot 1.381^n)$.

In addition to providing rigorous worst-case time complexity guarantees, our algorithms also achieve state-of-the-art empirical performance (as shown in Sec. 6). These characteristics offer advantages that extend beyond performance metrics; they provide a principled foundation for algorithm selection, allowing practitioners to reliably choose methods suited for diverse applications. By contrast, methods without formal complexity bounds or practical efficiency often suffer from unpredictable runtimes, complicating both performance analysis and system-level optimization in production environments.

5 FURTHER OPTIMIZATIONS

This section presents additional techniques to enhance the efficiency of finding the maximum biclique in G, including a heuristic

Algorithm 5: The heuristic algorithm

Input: Bipartite graph $G = (U, V, E)$
Output: A near-maximum biclique S^* and its size ω^*

1 Compute the degeneracy ordering O of G, where $O = U \cup V$;
2 $S^* \leftarrow G(\emptyset, \emptyset)$; $\omega^* = 0$;
3 **foreach** $u_i \in O_U$ *in reverse order of* O_U **do**
4     $C_V \leftarrow N_{u_i}(G)$; $v \leftarrow \arg\max_{v \in C_V} d_v(G)$;
5     $C_U \leftarrow N_v(G)$; $S_U \leftarrow \{v\}$; $O_U \leftarrow O_U \setminus \{u_i\}$;
6   **while** $|C_V| \geq \tau_V$ **do**
7         $u \leftarrow$ a vertex in C_U with largest degree in G;
8   Move u from C_U to S_U and update C_V with $C_V \cap N_u(G)$;
9   **if** $|S_U| \geq \tau_U$ *and* $|C_V| \geq \tau_V$ *and* $|S_U| \times |C_V| > \omega^*$ **then**
10             $S^* \leftarrow G(S_U, C_V)$; $\omega^* = |S_U| \times |C_V|$;
11   **if** $|U \setminus O_U| >$ *the threshold* 1000 **then break;**
12 Repeat lines 3-11 with vertices in O_V;
13 **return** S^* and ω^*;

approach, and improved progressive bounding, and a hybrid search framework.

The heuristic approach. It aims to find a near-maximum biclique by leveraging a simple yet effective approach. First, it computes an ordering of the vertices in G, and then generates a sequence of subgraphs G_i based on this ordering. For each subgraph G_i, it iteratively selects vertices with the largest degrees to generate a biclique, then it selects the largest one as the near-maximum biclique. We next introduce a degeneracy ordering to order the vertices, which is used in our algorithms.

Definition 3 (degeneracy ordering). Given a bipartite graph $G = (U, V, E)$, the degeneracy ordering $O = U \cup V$ is a permutation of the vertices $\{v_1, v_2, \cdots, v_n\}$ such that v_i has the smallest degree in the subgraph of G induced by $\{v_i, \cdots, v_n\}$.

Let O denote the total ordering of vertices in G. We define O_U (and similarly O_V) as the ordering of vertices in U (and V) induced by O; that is, $O_U = O \setminus V$. Consequently, if $v_i \prec v_j$ in O_U, then v_i also precedes v_j in O. Based on this ordering, we observe that the maximum biclique in G must reside within one of the subgraphs $G_i = G(\{u_j \in O_U | j \geq i, N_{u_i}(G) \cap N_{u_j}(G) \neq \emptyset\}, N_{u_i}(G))$, where u_i is a vertex in O_U. Using this insight, our heuristic approach iteratively selects vertices in each G_i with the largest degrees, expanding an initially empty subgraph until termination. For brevity, a detailed pseudocode of the algorithm is omitted.

In Algorithm 5, the degeneracy ordering of vertices is first computed using the algorithm in [32], requiring $O(m)$ time (line 1). Then, the algorithm generates each subgraph G_i based on the ordered vertices and iteratively selects vertices with the largest degree to form a near-maximum biclique (lines 3-12). To enhance practical performance, we restrict the selection to the last 1000 vertices in O_U and O_V for generating the subgraphs (line 11).

Improved progressive bounding. In [31], a progressive bounding technique is developed to solve the maximum biclique search problem by reducing the task of finding a maximum biclique $G(S_U, S_V)$ with $|S_U| \geq \tau_U$ and $|S_V| \geq \tau_V$ to finding the maximum biclique for each pair of bounds τ_U^i and τ_V^i, where $\tau_U^i = 2^i \times \tau_U$ and $\tau_V^i = \frac{\omega}{2 \times \tau_U^i}$. This approach dramatically improves efficiency since the size constraints always satisfy $\tau_U^i \geq \tau_U$ and $\tau_V^i \geq \tau_V$ for all $i \geq 1$. However,

Algorithm 6: iMEBS

Input: Bipartite graph $G = (U, V, E)$, τ_U, τ_V, δ, and s
Output: The maximum biclique H of G and its size ω

1 Compute degeneracy ordering O of G, where $O = U \cup V$;
2 Obtain a biclique H and its size ω with a heuristic approach;
3 **foreach** $u_i \in O_U$ **do**
4     $C_V \leftarrow N_{u_i}(G)$; $C_U \leftarrow \{u_j \in O_U | j > i, N_{u_j}(G) \cap C_V \neq \emptyset\}$;
5     $S_U \leftarrow \{u_i\}$; $S_V \leftarrow \emptyset$; $r \leftarrow \lceil \log_s(\frac{\omega}{\tau_U \times \tau_V}) - 1 \rceil$;
6   **foreach** *j from* 0 *to* r **do**
7         $\tau_U^j \leftarrow \min(\frac{\omega}{s \times \tau_V}, \tau_U \times s^j)$; $\tau_V^j \leftarrow \omega / (s \times \tau_U^j)$;
8   Remove unnecessary vertices in $G(C_U, C_V)$ [31];
9   **if** $|E(C_U, C_V)| / (|C_U| \times |C_V|) \geq \delta$ **then**
10   Let $\overline{G}$ be the complement of $G(C_U, C_V)$;
11             $\alpha_U = |C_U| + 1 - \tau_U^j$; $\alpha_V = |C_V| - \tau_V^j$;
12   *CBranch*$(\emptyset, \emptyset, C_U, C_V)$;
13   **else** *Branch*$(S_U, S_V, 0, C_U, C_V)$;
14 **return** H and ω;

for dense bipartite graphs, these constraints may not be sufficiently tight (i.e., $\tau_U^i \times \tau_V^i \leq \omega/2$).To address these issues, we modify the progressive bounding technique as presented below.

Lemma 13. Given a bigraph graph G, the problem of finding the maximum biclique $G(S_U, S_V)$ in G with $|S_U| \geq \tau_U$ and $|S_V| \geq \tau_V$ is equivalent to solving a series of subproblems of finding the maximum biclique of G with size constrains τ_U^i and τ_V^i, where $\tau_U^i = s^i \times \tau_U$, $\tau_V^i = \frac{\omega}{s \times \tau_U^i}$, $s > 1$, and i is an integer in $[0, \log_s(\frac{\omega}{\tau_U \times \tau_V}) - 1]$.

Proof. This technique ignores subproblems seeking bicliques with sizes of each side no larger than τ_U^{i+1} and τ_V^i. For each i, we have $\tau_U^{i+1} \times \tau_V^i = s^{i+1} \times \tau_U \times \frac{\omega}{s \times s^i \times \tau_U} \leq \omega$. Consequently, any biclique in these subproblems would contain at most ω edges, ensuring no larger solution exists and validating their exclusion. □

A hybrid search algorithm. Building on the techniques discussed, we present a hybrid search algorithm that combines the proposed pivot-based and cover-based branching methods. The algorithm is outlined in Algorithm 6. In the initial step, it computes the degeneracy order and identifies a near-maximum biclique (lines 1–2). It then generates a series of subgraphs G_i following the degeneracy order, aiming to compute the maximum biclique of G containing vertex u_i in O_U (lines 3–13). For each subgraph, the algorithm uses the proposed improved progressive bounding technique to determine the maximum biclique under the pair of size constraints τ_U^j and τ_V^j (lines 6–13). Notably, if the density of current subgraph $G(C_U, C_V)$ (defined as $\rho_{G(C_U, C_V)} = \frac{|E(C_U, C_V)|}{|C_U| \times |C_V|}$) is at least the threshold δ, the vertex cover-based branching procedure *CBranch* is used to compute the maximum biclique (lines 9–12). Otherwise, the pivot-based branching procedure *CBranch* is applied (line 13). After processing all vertices in O_U, the algorithm returns H and its size ω as the maximum biclique of G (line 14).

We note that our proposed solutions can be naturally extended to find the top-r maximum bicliques of G. Generally, we can use an empty set $\mathcal{D}$ to store the results. Then, the algorithm first collects the initial r maximal bicliques into $\mathcal{D}$. During the branch-and-bound search process, the smallest biclique in $\mathcal{D}$ is iteratively replaced whenever a larger biclique is discovered. Due to space limited, we omit the detailed implementations.

6 EXPERIMENTS

6.1 Experimental Setup

Algorithm Implementation. We implemented two algorithms, MEBS and iMEBS, to compute the maximum biclique in bipartite graphs. The MEBS algorithm follows the framework detailed in Algorithm 2, incorporating the techniques from Section 3, the heuristic approach, and improved progressive bounding technique in Section 5. The iMEBS algorithm extends MEBS by integrating all methodologies proposed in this paper (Algorithm 6). For performance evaluation, we benchmarked our algorithms against the SOTA MBC algorithm [31]. Although the source code of MBC was unavailable, we implemented an efficient version of MBC that achieves performance comparable to, or exceeding, the results reported in the original study [31]. All algorithms were implemented in C++ and executed on a CentOS system with a 2.2 GHz CPU and 128 GB of memory.

Datasets. We evaluated the algorithms using 12 real-world bipartite graphs, covering various categories such as affiliation networks, interaction networks, and rating networks, with varying densities. For instance, the datasets YouTube, IMDB, Mind and LiveJournal are the sparse graphs, while the datasets MovieLens, Epinions, Libimseti are dense graphs. Several of these graphs have been used in previous studies on maximum biclique search algorithms [31]. Table 3 provides detailed statistics of the bipartite graphs, including the maximum degrees of vertices in sets U (d_{1max}) and V (d_{2max}). All datasets are accessible for download from the KONECT project (http://konect.cc) and Network Repository (https://networkrepository.com/index.php).

Parameters. In our experiments, we vary the size constraints $\tau_U = \tau_V$ from 3 to 11. the two partitions of a maximal biclique can differ substantially; larger values of τ_U, τ_V help mitigate this imbalance (Table 4). We fix the parameter $s = 1.6$ for the improved progressive bounding technique (as we observed in experiments $s = 1.6$ is a good selection). Finally, in iMEBS, we set the density threshold to $\delta = 0.4$ by default.

6.2 Experimental Results

Exp-1: Runtime of all algorithms on benchmark graphs. In this experiment, we evaluated the runtime of our two proposed algorithms, MEBS and iMEBS, against the SOTA MBC algorithm on all benchmark real-world bipartite graphs. Table 4 reports the runtime (in seconds) for each algorithm under various size constraints, where $\tau = \tau_U = \tau_V$. The symbol "—" indicates that the algorithm either timed out within 24 hours or could not find a maximum biclique under the current parameter constraints. The results clearly show that both MEBS and iMEBS consistently outperform MBC across all datasets. Notably, while MBC required hundreds seconds or timed out on larger or denser graphs, our methods achieved dramatic speedups. For example, on the Sualize dataset with $\tau = 3$, MBC took 3410.85 seconds compared to merely 3.90 and 3.96 seconds for MEBS and iMEBS, respectively, highlighting the efficiency of our proposed algorithm in finding the maximum biclique. Although runtime generally increased with larger τ, our methods exhibited slower growth rates, particularly excelling on dense or large-scale graphs (e.g., Sualize and Mind) where MBC frequently

Table 3: The statistics of tested bipartite graphs.

Dataset	$\|U\|$	$\|V\|$	m	d_{1max}	d_{2max}
MovieLens	943	1,682	100,000	689	583
YouTube	94,238	30,087	293,360	761	2,338
Sualize	17,122	82,035	449,503	4,382	4,720
EachMovie	1,623	61,265	2,811,717	32,561	1,455
Digg-votes	139,409	3,553	3,010,898	3,415	20,052
TV-Tropes	64,415	87678	3,232,134	6,507	1,2400
IMDB	303,617	896,302	3,782,463	933	1,590
Flickr	395,979	103,631	8,545,307	2,186	34,989
Epinions	120,492	755,760	13,668,320	162,169	1,195
Libimseti	135,359	168,791	17,359,346	24,581	33,389
Mind	876,956	97,509	18,149,915	1,001	76,843
LiveJournal	3,201,203	7,489,073	112,307,385	276	1,038,343
WebTrackers	27,665,730	12,756,244	140,613,762	1,100,065	11,571,952

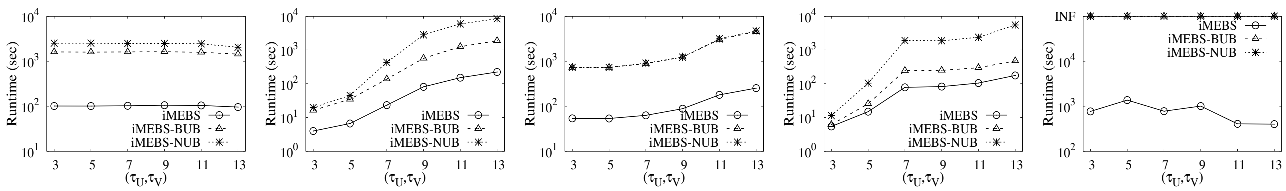
timed out. Moreover, the optimizations in iMEBS enabled it to outperform MEBS in many cases (e.g., in Movielens, a 17× speedup at $\tau = 13$, and over 100× faster in Epinions at $\tau = 3$). These outcomes demonstrate that iMEBS is robust across a range of τ values (with occasional runtime reductions observed for higher τ in datasets like Epinions and Libimseti due to specific graph properties). Overall, our methods offer scalable, practical solutions, with iMEBS proving especially effective for imbalanced or dense real-world graphs.

Exp-2: Efficiency of the proposed upper bounds. In this experiment, we assessed the impact of our upper-bound techniques integrated into iMEBS. To conserve space, we omit the results for MEBS, as they follow the same trend. Figure 7 shows runtime comparisons between iMEBS, iMEBS-BUB (using basic upper bounds in [31]), and iMEBS-NUB (with no upper bounds). The results reveal the critical role of proposed upper-bound pruning strategies in enhancing algorithmic efficiency across diverse bipartite graphs. For example, on the Epinions dataset with $\tau_U = \tau_V = 3$, iMEBS finished in 766.56 seconds, whereas both iMEBS-BUB and iMEBS-NUB did not finish within a the 24-hour limit. Furthermore, the performance gap between iMEBS and its variants widened as τ increased. For example, on the Flickr dataset, the speedups of iMEBS over iMEBS-BUB and iMEBS-NUB increased from 1.13× and 2.09× at $\tau_U = \tau_V = 3$ to 2.51× and 29.21× at $\tau_U = \tau_V = 13$, respectively. These results indicate that our optimized upper-bound techniques significantly mitigates inherent complexity barriers in certain bipartite structures by balancing pruning effectiveness with computational overhead, thereby enabling practical scaling of τ in large, real-world graphs.

Exp-3: Efficiency of the branch reduction techniques. We further evaluated the performance of the proposed branch reduction techniques in iMEBS. The performance trend for MEBS is similar to that of iMEBS, so we omit its separate evaluation. Fig. 8 displays the runtime results for iMEBS and iMEBS-NRD with varying the size constrains on 5 represented bipartite graphs, where iMEBS-NRD is the version of iMEBS without the branch reduction techniques discussed in Sections 3.1 and 4.2. The results demonstrate that iMEBS consistently outperforms iMEBS-NRD across all tested bipartite graphs as the size constraints τ_U and τ_V varies. Moreover, as τ_U (or τ_V) increases, the performance gap between iMEBS and iMEBS-NRD widens in most cases. For instance, on the Sualize dataset, when $\tau_U = \tau_V = 3$ or 5, the runtime of both algorithms is almost identical. However, at $\tau_U = \tau_V = 13$, iMEBS achieves a

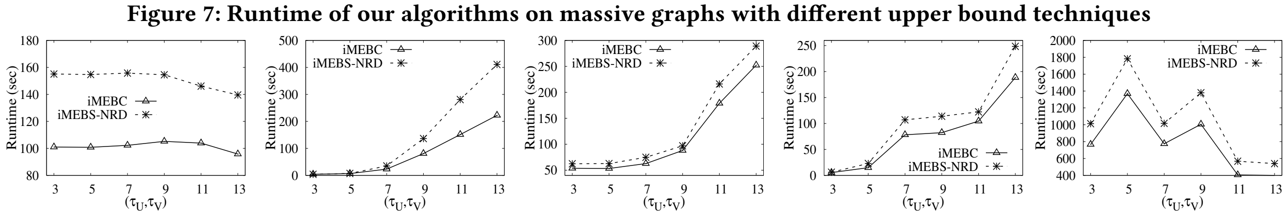
Table 4: Runtime of each algorithm on benchmark real-world graphs (in second), where τ represents the size constraint ($\tau = \tau_U = \tau_V$) and (ω_U, ω_V) denotes the size of the maximum clique.

Dataset	τ	(ω_U, ω_V)	Runtime (in seconds)			τ	(ω_U, ω_V)	Runtime (in seconds)			τ	(ω_U, ω_V)	Runtime (in seconds)		
			MBC	MEBS	iMEBS			MBC	MEBS	iMEBS			MBC	MEBS	iMEBS
MovieLens	3	(181,7)	—	1749.44	**100.97**	7	(181,7)	—	1700.86	**102.26**	11	(115,11)	—	1739.54	**103.91**
	5	(181,7)	—	1747.48	**100.80**	9	(115,11)	—	1759.81	**105.20**	13	(13,96)	—	1240.02	**95.81**
YouTube	3	(3,313)	0.03	**0.024**	0.024	7	(7,30)	10.046	**0.052**	0.053	11	(15,11)	0.867	0.047	**0.044**
	5	(5,71)	1.148	**0.039**	0.04	9	(20,9)	4.607	0.049	**0.046**	13	—	0.177	0.015	**0.014**
Sualize	3	(3,936)	3410.85	**3.90**	3.96	7	(7,252)	—	44.92	**23.18**	11	(11,101)	—	744.92	**151.09**
	5	(5,492)	—	7.51	**6.61**	9	(9,143)	—	305.51	**80.80**	13	(13,79)	—	1266.09	**222.38**
EachMovie	3	(5,15836)	8588.06	217.53	**53.62**	7	(7,10953)	14162.40	251.83	**62.65**	11	(11,5675)	80425.72	992.67	**178.81**
	5	(5,15836)	9528.51	216.70	**53.27**	9	(9,8078)	25907.87	384.17	**87.98**	13	(13,4504)	—	1533.63	**252.24**
Digg-votes	3	(2196,5)	—	1099.02	**1094.99**	7	(1260,7)	—	4302.55	**3151.34**	11	(386,11)	—	—	**56864.52**
	5	(2196,5)	—	**1101.00**	1102.95	9	(696,9)	—	35623.78	**11675.35**	13	—	—	—	—
TV-Tropes	3	(5,1209)	247.67	32.57	**32.05**	7	(7,501)	15779.44	**46.50**	47.20	11	(11,69)	—	279.15	**272.96**
	5	(5,1209)	5622.51	23.83	**23.18**	9	(9,138)	—	**165.32**	168.82	13	(15,29)	—	476.61	**471.28**
IMDB	3	(186,3)	0.99	**0.364**	0.393	7	(68,7)	76.88	**0.230**	0.234	11	(33,11)	5.53	0.212	**0.207**
	5	(102,5)	35.03	**0.280**	0.284	9	(44,9)	17.47	0.240	**0.238**	13	(27,13)	1.87	0.112	**0.109**
Flickr	3	(3,7609)	49.27	5.58	**5.41**	7	(145,62)	6122.95	79.94	**78.19**	11	(145,62)	—	115.67	**104.54**
	5	(5,3051)	328.69	**13.65**	14.84	9	(145,62)	6774.05	85.47	**82.27**	13	(145,62)	—	218.11	**175.88**
Epinions	3	(32,1801)	—	—	**766.56**	7	(32,1801)	—	—	**774.79**	11	(32,1801)	—	—	**405.54**
	5	(32,1801)	—	—	**1368.92**	9	(32,1801)	—	—	**1005.88**	13	(32,1801)	—	—	**399.93**
Libimseti	3	(3,16629)	—	971.17	**809.02**	7	(13,3675)	—	2462.01	**1076.70**	11	(13,3675)	—	1271.20	**760.89**
	5	(13,3675)	—	4697.02	**1650.68**	9	(13,3675)	—	3063.07	**1007.37**	13	(13,3675)	—	969.97	**556.08**
Mind	3	(3,2437)	1426.47	**67.82**	69.928	7	(7,77)	—	**1459.29**	1463.93	11	(11,13)	—	**2115.01**	2123.14
	5	(5,319)	20114.52	**657.14**	670.22	9	(9,28)	—	2310.516	**2239.74**	13	—	—	706.02	**701.43**
LiveJournal	3	(3,176689)	59.92	**47.88**	49.77	7	(7,13817)	3762.60	**176.70**	207.50	11	(11,1825)	—	**2686.16**	2702.46
	5	(5,41036)	446.61	**86.95**	111.31	9	(9,4419)	38054.01	**818.22**	876.70	13	(13,953)	—	5582.05	**5488.03**



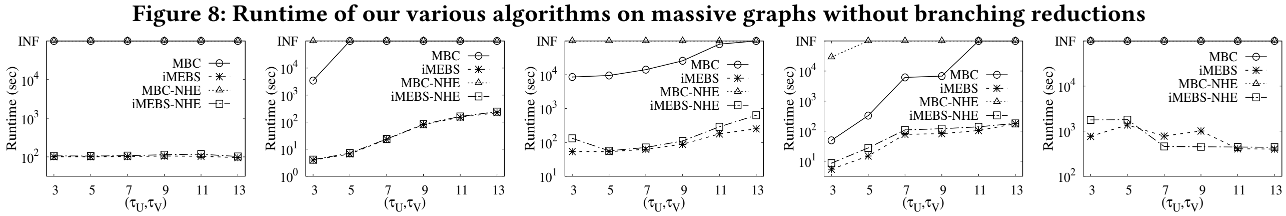
(a) Movielens (b) Sualize (c) EachMovie (d) Flickr (e) Epinions

Figure 7: Runtime of our algorithms on massive graphs with different upper bound techniques



(a) Movielens (b) Sualize (c) EachMovie (d) Flickr (e) Epinions

Figure 8: Runtime of our various algorithms on massive graphs without branching reductions

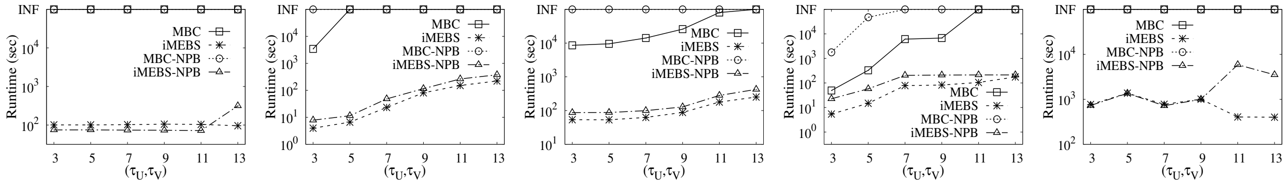


(a) Movielens (b) Sualize (c) EachMovie (d) Flickr (e) Epinions

Figure 9: Runtime of our various algorithms on massive graphs without heuristic algorithms

1.84× speedup over iMEBS-NRD. This improvement is due to the increased number of branches generated as τ_U and τ_V grows. The branch reduction techniques reduce unnecessary branches more effectively when τ_U (or τ_V) is large, leading to better performance for iMEBS. These results suggest that the proposed branch reduction techniques are efficient in pruning unnecessary computations.

Exp-4: Impact of the heuristic approach on performance. In this experiment, we evaluated the effect of the heuristic approach on the performance of our proposed algorithms. We also

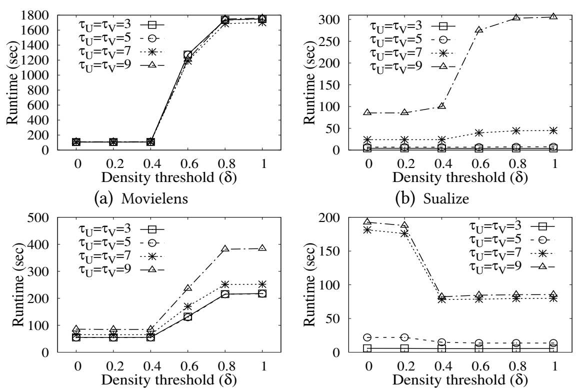


(a) Movielens (b) Sualize (c) EachMovie (d) Flickr (e) Epinions

Figure 10: Runtime of our various algorithms on massive graphs without progressive boundings

Table 5: The quality returned by our heuristic approach, where ω' is the result obtained by the heuristic approach.

Dataset	τ	$\frac{\omega'}{\omega}$	Dataset	τ	$\frac{\omega'}{\omega}$	Dataset	τ	$\frac{\omega'}{\omega}$
MovieLens	3	0.808	YouTube	3	1.0	Sualize	3	1.0
	5	0.876		5	0.873		5	1.0
EachMovie	3	1.0	digg-votes	3	1.0	Epinions	3	0.677
	5	1.0		5	1.0		5	0.622
IMDB	3	1.0	Flickr	3	1.0	LiveJournal	3	1.0
	5	1.0		5	1.0		5	1.0

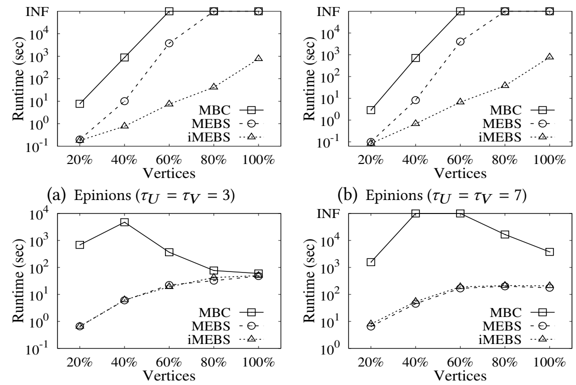


(a) Movielens (b) Sualize (c) EachMovie (d) Flickr

Figure 11: Efficiency of iMEBS with varying δ

compare the performance of MBC when the heuristic approach is not employed. Fig. 9 shows the runtime of four algorithms, MBC, iMEBS, MBC-NHE, and iMEBS-NHE, across 5 representative bipartite graphs with varying τ_U (and τ_V). Here, MBC-NHE and iMEBS-NHE denote the versions of MBC and iMEBS without the heuristic approaches, respectively. The results indicate that the runtime gap between iMEBS and iMEBS-NHE is relatively small as τ_U (or τ_V) varies, suggesting that the performance of our algorithm is not highly dependent on the application of the heuristic approach during preprocessing. However, the performance of MBC drops significantly when the heuristic approach is not used. For example, on the Flickr dataset with $\tau_U = \tau_V = 3$, the runtime for iMEBS and iMEBS-NHE is 5.41 and 8.57 seconds, respectively, while MBC-NHE is at least 589 times slower than MBC with the same parameters. This is because the efficiency of MBC largely relies on the progressive bounding technique, which requires a good initial solution for pruning. These results demonstrate the robustness of our maximum biclique search algorithm, as its performance does not heavily depend on the quality of the results of the heuristic search. This further lowers the barrier to adoption of the proposed algorithm in real-world applications.

In addition, we evaluated the effectiveness of the proposed heuristic method for approximating the maximum biclique in a bipartite graph G. The experimental results, presented in Table 5, report



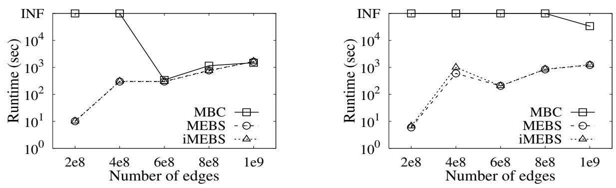
(a) Epinions ($\tau_U = \tau_V = 3$) (b) Epinions ($\tau_U = \tau_V = 7$) (c) LiveJournal ($\tau_U = \tau_V = 3$) (d) LiveJournal ($\tau_U = \tau_V = 7$)

Figure 12: Scalability testing of various algorithms under different constrains

performance on six representative bipartite graphs under varying values of τ_U and τ_V. When both τ_U and τ_V are set to small values, the heuristic probably produces results that match those obtained by the exact algorithm across the majority of test cases. These findings confirm that our heuristic approach, while simple, is effective in identifying high-quality near-maximum bicliques in bipartite graphs.

Exp-5: Efficiency of the progressive bounding. We then evaluated the efficiency of the progressive bounding technique integrated into iMEBS. Fig. 10 shows the results for the algorithms MBC, iMEBS, MBC-NPB, and iMEBS-NPB across 5 representative bipartite graphs with varying τ_U (τ_V). Here, MBC-NPB and iMEBS-NPB represent the versions of MBC and iMEBS, respectively, that do not employ the progressive bounding technique. The results demonstrate that the runtime of all algorithms increases when excluding the progressive bounding technique. However, even without this technique, our algorithms still outperform the SOTA MBC algorithm. For instance, on the EachMovie dataset with $\tau_U = \tau_V = 3$, the runtime of our iMEBS-NPB (which excludes the progressive bounding technique) is only 86.73 seconds, whereas MBC takes 8588.06 seconds, which is 99 times slower than iMEBS-NPB. Additionally, we observe that MBC heavily relies on the progressive bounding technique, as its performance degrades sharply without it. In contrast, the impact of excluding progressive bounding on iMEBS is less pronounced. For example, on the Sualize and EachMovie datasets, iMEBS-NPB performs at most 2.08 times slower than iMEBS across varying τ_U (and τ_V) values from 3 to 13. These results further highlight the powerful pruning capabilities of the proposed techniques in efficiently finding the maximum biclique.

Exp-6: Efficiency of iMEBS with different density threshold. We evaluated the runtime of iMEBS as a function of the density threshold δ. Fig 11 reports results on four representative bipartite



(a) ($\tau_U = \tau_V = 3$) (b) ($\tau_U = \tau_V = 7$)

Figure 13: Runtime of each algorithm on large sparse synthetic bipartite graphs

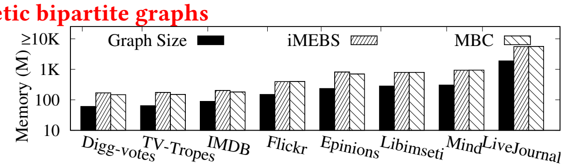
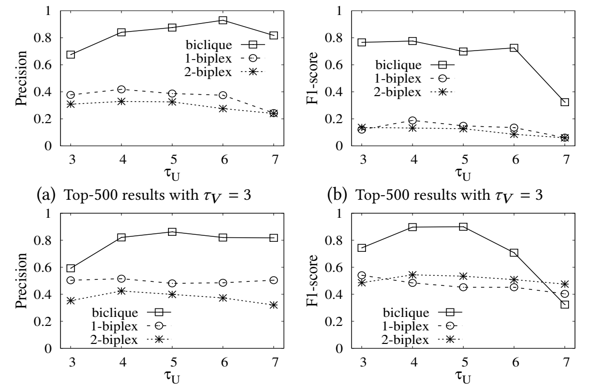


Figure 14: Memory usages of iMEBS.

graphs, with δ varying from 0 to 1. Note that when $\delta = 0$, iMEBS reduces to the pure cover-based algorithm (Algorithm 4), whereas $\delta = 1$ recovers MEBS. On dense graphs, the runtime of iMEBS remains essentially constant for $\delta \leq 0.4$, but increases sharply once $\delta \geq 0.6$. For example, on the EachMovie dataset with $\tau_U = \tau_V = 3$, the runtime grows from at most 55.3 seconds at $\delta = 0.4$ to at least 132.1 seconds when $\delta \geq 0.6$. Moreover, as the size constraints τ_U and τ_V increase, the sensitivity of runtime to δ becomes even more pronounced. This effect arises because larger τ_U and τ_V yield denser bipartite graphs and a smaller maximum biclique, which degrades the pruning efficiency of MEBS but benefits the cover-based approach. Hence, iMEBS performs well for large τ_U (and τ_V). Conversely, for very small δ, iMEBS may be inefficient on some graphs, since the cover-based approach operates on the complement graph and is less effective when few edges are present. Based on these results, we adopt $\delta = 0.4$ as the default threshold, which yields robust performance across both sparse and dense bipartite graphs.

Exp-7: Scalability testing. This experiment assessed the scalability of each algorithm by generating eight subgraphs through vertex sampling (20–80%) from the Epinions and LiveJournal datasets. Fig. 12 reports the runtime of our algorithms (iMEBS and MEBS) and the SOTA benchmark (MBC) under constraints $\tau_U = \tau_V = 3$ and 7. The results from other datasets show consistent trends. Our findings reveal that iMEBS scales smoothly with increasing graph size, maintaining efficiency across both sparse and dense bipartite graphs, while MBC proves inefficient on larger graphs regardless of density. For example, on a 40% vertex-sampled subgraph from the LiveJournal dataset, when $\tau_U = \tau_V = 7$, iMEBS and MEBS complete computations within 54.36 seconds, while MBC fails to terminate within 24 hours. Interestingly, MBC shows reduced runtime as the graph size increases. This counterintuitive behavior is attributed to the presence of large bicliques in the original LiveJournal dataset, which enable aggressive pruning. However, when smaller subgraphs are derived from LiveJournal, the lack of such large bicliques reduces pruning effectiveness, leading to higher computational overhead. These results emphasize the robust scalability of our algorithms, particularly iMEBS, in handling large-scale real-world bipartite graphs across a variety of density regimes.

Exp-7: Efficiency on very large synthetic bipartite graphs. To further investigate the scalability of our proposed algorithms, we generate a set of synthetic bipartite graphs whose degrees follow a power-law distribution, with the number of vertices ranging from 2.76×10^7 to 2.76×10^8 and the number of edges varying from 1×10^8 to 1×10^9. We evaluate the runtime performance of various



(a) Top-500 results with $\tau_V = 3$ (b) Top-500 results with $\tau_V = 3$

(c) Top-5000 results with $\tau_V = 3$ (d) Top-5000 results with $\tau_V = 3$

Figure 15: Fraud detection results on "Prime Pantry"

algorithms on these graphs under two size constraint settings: $\tau_U = \tau_V = 3$ and $\tau_U = \tau_V = 7$. As shown in Fig. 13, the runtime of our algorithms increases only modestly as the graph size grows, under both constraint settings, which confirms the time efficiency of our methods on large-scale bipartite graphs. Moreover, when τ_U and τ_V increases, our algorithms (MEBS and iMEBS) typically have minimal impact on the runtime, while the runtime of the state-of-the-art algorithm MBC increases significantly. For instance, the runtime of MBC at $\tau_U = \tau_V = 7$ is at least 60 times longer than at $\tau_U = \tau_V = 3$, whereas our algorithms demonstrate a factor of at most 1.13 times longer. This contrast highlights the strength of our branching strategies and upper-bound pruning techniques. Overall, the results further demonstrate that our methods are robust for processing very large bipartite graphs.

Exp-8: Memory overhead. This experiment evaluates the memory usage of our algorithm iMEBS and existing algorithm MBC on eight large bipartite graphs, with results presented in Fig. 14. We omit the memory usage results of MEBS, as it is implemented using the same data structures as iMEBS and thus exhibit comparable memory consumption. As shown, the memory usage of iMEBS remains only slightly larger than the graph size itself, and is comparable to that of MBC. This efficiency stems from the fact that, beyond storing the full bipartite graph, our algorithms rely solely on a few arrays of size n and an index of size m to recognize the connection relations of vertices for the branch-and-bound process. These results demonstrate that our approach is space-efficient and well-suited for processing large-scale real-world bipartite graphs.

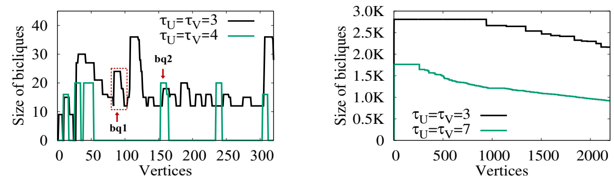
6.3 Effective testing

Exp-9: fraud detection via maximum bicliques. Here we evaluate the effectiveness of the studied maximum biclique model in a real-world fraud detection scenario. Specifically, we simulate a camouflage attack on the "Prime Pantry" dataset from the Amazon review corpus (https://nijianmo.github.io/amazon/index.html), which includes 447,399 reviews over 10,814 items by 247,659 users. To simulate fraudulent behavior, we inject 200 fake items, 1,000 fake users, 25,000 fake reviews, and 25,000 camouflage reviews into the dataset. Fake (camouflage) reviews are randomly generated by connecting fake users to fake (real) items [4, 5, 51]. To detect fraud, we extract the top-r maximum bicliques from the augmented user-item bipartite graph. The top-r maximum bicliques refer to the r bicliques with the highest edge counts. For comparison, we include the k-biplex model (with $k = 1$ and 2) as a baseline, which is a widely-used cohesive subgraph model. Fig. 15 reports the precision and F1-score of the detected fake users and fake items using the top-500 and top-5000 results, under varying τ_U on the item side. We denote the biclique method as biclique, and the k-biplex models as 1-biplex and 2-biplex. Due to the high computational cost of algorithms for enumerating maximum k-biplexes [12], we constrain the runtime of each algorithm to 3 hours and report the currently discovered top-r results. The results demonstrate that biclique consistently achieves higher precision and F1-score than k-biplex models when $\tau_U \leq 6$. This advantage stems from the nature of the planted fake user groups, which form small and densely connected subgraphs, favoring the stricter connectivity of the biclique model. In contrast, k-biplexes are more relaxed and tend to include many irrelevant nodes. Moreover, we observe that even a small number of maximum bicliques can yield very high detection accuracy. For example, with just the top-500 bicliques, the precision of biclique exceeds 80%, whereas the k-biplex models fail to surpass 60% precision, even with top-5000 results. These findings highlight the superiority of maximum bicliques for fraud detection and affirm that our studied maximum biclique search problem is both practically relevant and highly effective in real-world applications.

Exp-10: Cohesive subgraph visualization. In this experiment, we investigate the application of maximum biclique search to Cohesive Subgraph Visualization (CSV), a technique that projects nodes and edges into a multi-dimensional space such that densely clustered regions in the visualization correspond to cohesive subgraphs [50]. The biclique-based CSV approach is straightforward: starting from a randomly selected unvisited vertex, it performs a depth-first traversal, at each step plotting the vertex associated with the biclique of the largest size among those encountered so far. This method offers the following desirable properties [50]: (1) it identifies all bicliques and all connected subgraphs of size larger than $\tau_U \times \tau_V$; and (2) it captures all subgraphs with density exceeding γ and size greater than $\tau_U \times \tau_V$. We conduct experiments on two representative bipartite graphs, Actor-Movie, derived from IMDB data (https://datasets.imdbws.com), and Sualize, to evaluate the effectiveness of biclique-based CSV. Fig. 16 presents visualization results under varying thresholds τ_U and τ_V. To enhance readability, we display only the first 300 and 2000 vertices for Actor-Movie and Sualize, respectively, although the overall patterns are consistent with larger vertex sets. The visualizations reveal distinct cohesion characteristics: in Actor-Movie, dense regions appear dispersed and relatively independent, while in Sualize, they exhibit continuity with a gradual decline in density. Notably, sequences of consecutive vertices associated with large bicliques frequently indicate the presence of dense subgraphs in G. For example, Fig.17 depicts a subgraph corresponding to the region "bq1" in Fig.16(a), where significant variation in biclique sizes between adjacent vertices suggests "bq1" is a dense cluster connected to other regions via green vertices serving as bridges. In contrast, region "bq2" (when $\tau_U = \tau_V = 4$) forms an isolated cohesive subgraph within Actor-Movie. These results highlight the effectiveness of maximum biclique search for CSV and underscore its utility in visualizing and analyzing the structural cohesiveness of large-scale bipartite graphs.

7 RELATED WORKS

Maximum biclique search and its variants. The maximum biclique search problem is typically framed in three main forms: the



(a) Actor-Movies (b) Sualize

Figure 16: Cohesive subgraph visualization of bipartite graphs using the maximum biclique search

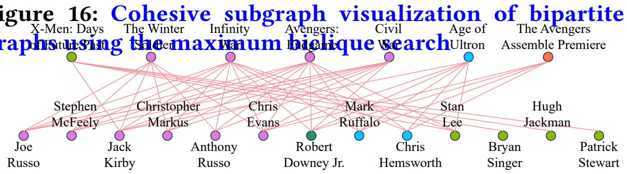


Figure 17: The cohesive subgraph "bq1" in Fig. 16, where green vertices act as bridges connecting other dense subgraphs

maximum edge biclique (MEB) problem, the maximum vertex biclique (MVB) problem, and the maximum balanced biclique (MBB) problem. The MEB problem focuses on finding a complete subgraph on bipartite graphs with the maximum number of edges, which problem is known to be NP-complete [35]. To solve this problem, researchers have developed a variety of algorithms [2, 31, 38, 40], which include integer programming [2, 40], Monte Carlo [38], and branch-and-bound techniques [31]. Recently, Wang et al. [48] studied the personalized MEB search problem, which aims finding a MEB that contains a query vertex q under the size constrains τ_U and τ_V. The MVB problem, which seeks the largest complete subgraph by vertex count, can be solved in polynomial time for bipartite graphs, often through reductions to maximum matching or flow-based techniques [16]. The MBBP problem aims to find a largest biclique where the two disjoint vertex sets are of equal size. This problem is also NP-hard [16]. Recent studies have proposed efficient exact algorithms for solving the MBB problem in large bipartite graphs [9, 34, 56], such as the symmetry-breaking technique introduced in [34], the branch-and-bound approach with upper bound propagation in [56], and the enumeration algorithm with worst-time guarantees in [9]. In addition, the inapproximability of the MBB problem has been explored, showing that near-optimal solutions cannot be found in polynomial time [33]. Some heuristic approaches for the MBB problem have also been wildly studied, including node deletion-based methods [3, 52], tube search-based methods [55], and swap-based approaches [27].

We also note that some maximum SAT-based approaches have been successfully applied to the maximum clique problem [23, 24]. However, to the best of our knowledge, these techniques have not yet been extended to address the maximum biclique problem in bipartite graphs. This remains a challenging task for several reasons. First, the objective of the maximum biclique problem is to identify the biclique with the largest number of edges, whereas existing SAT-based solvers are primarily designed to maximize the number of vertices. Second, the development of efficient upper-bounding strategies, which are critical to the performance of SAT-based approaches, remains an open problem in the context of maximum biclique search. It is still unclear how to adapt such upper-bounding techniques effectively for this setting. Despite these challenges, we recognize SAT-based formulations as a promising research direction that may yield valuable insights for solving the maximum biclique problem in the future.

Maximal biclique enumeration. The problem of enumerating all maximal bicliques has also been extensively studied [1, 10, 11, 13, 14, 26, 29, 53]. Early works are typically based on BFS methods [26, 29], which explore the power set of one vertex partition but often suffer from high computational overhead. To improve efficiency, subsequent researches have largely focused on DFS-based methods. For example, Zhang et al. [53] proposed an iterative approach by leveraging the structural property that $G(N_{N_A(G)}(G), N_A(G))$ forms a maximal biclique for any vertex set A. Das et al. [14] enhanced the performance by employing vertex-ordering to break the problem into smaller subproblems. Abidi et al. [1] introduced a pivot-based method to prune redundant branches during traversal, and Chen et al. [10] developed a polynomial-delay approach that integrates vertex ordering with domination strategies for better computational efficiency. More recently, Dai et al. [11] proposed a new pivot-based framework achieving both near-optimal worst-case time complexity and polynomial-delay guarantees. However, applying these maximal biclique enumeration methods to the maximum biclique search problem remains inefficient, as they cannot prune unnecessary maximal bicliques. To address this, we propose novel branching rules and upper-bound techniques that significantly improve the practical efficiency and worst-case time complexity of the maximum biclique search.

8 CONCLUSION

In this paper, we investigate the problem of finding the maximum edge biclique in bipartite graphs, a problem that has garnered significant attention in recent years. Given the theoretical and practical limitations of existing methods, we propose several novel and efficient solutions. Specifically, we develop two new algorithms based on a pivot-based branching rule and a vertex-cover-based branching rule, achieving time complexities of $O(m \cdot 1.348^n)$ and $O(m \cdot 1.381^n)$, respectively, both of which outperform SOTA approaches. To further enhance efficiency, we introduce a series of branch reduction and upper-bound techniques for both algorithms. Additionally, to leverage the strengths of each approach, we design a hybrid algorithm that separately processes the sparse and dense regions of the graph, integrating heuristic strategies and an improved progressive bounding technique. Finally, we conduct extensive experiments on 12 real-world bipartite graphs to illustrate the efficiency and scalability of the proposed algorithms.

REFERENCES

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

[2] Vicente Acuña, Carlos Eduardo Ferreira, Alexandre S. Freire, and Eduardo Moreno. 2014. Solving the maximum edge biclique packing problem on unbalanced bipartite graphs. *Discret. Appl. Math.* 164 (2014), 2–12.

[3] Ahmad A Al-Yamani, Sundarkumar Ramsundar, and Dhiraj K Pradhan. 2007. A defect tolerance scheme for nanotechnology circuits. *IEEE Trans. Circuits Syst. I Regul. Pap.* 54, 11 (2007), 2402–2409.

[4] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Seyed-Mehdi-Reza Beheshti, Elisa Bertino, and Norman Foo. 2013. Collusion Detection in Online Rating Systems. In *APWeb*, Vol. 7808. 196–207.

[5] 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.

[6] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. *Computer networks* 33, 1-6 (2000), 309–320.

[7] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guojie Li, and Runsheng Chen. 2003. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic Acids Research* 31, 9 (05 2003), 2443–2450.

[8] Monika Cerinsek and Vladimir Batagelj. 2015. Generalized two-mode cores. *Soc. Networks* 42 (2015), 80–87.

[9] 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.

[10] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2022. Efficient Maximal Biclique Enumeration for Large Sparse Bipartite Graphs. *Proc. VLDB Endow.* 15, 8 (2022), 1559–1571.

[11] Qiangqiang Dai, Rong-Hua Li, Xiaowei Ye, Meihao Liao, Weipeng Zhang, and Guoren Wang. 2023. Hereditary Cohesive Subgraphs Enumeration on Bipartite Graphs: The Power of Pivot-based Approaches. *Proc. ACM Manag. Data* 1, 2 (2023), 138:1–138:26.

[12] Qiangqiang Dai, Rong-Hua Li, Donghang Cui, Meihao Liao, Yu-Xuan Qiu, and Guoren Wang. 2024. Efficient Maximal Biplex Enumerations with Improved Worst-Case Time Guarantee. *Proc. ACM Manag. Data* 2, 3 (2024), 135:1–135:26.

[13] Peter Damaschke. 2014. Enumerating maximal bicliques in bipartite graphs with favorable degree sequences. *Inf. Process. Lett.* 114, 6 (2014), 317–321.

[14] Apurba Das and Srikanta Tirthapura. 2019. Shared-Memory Parallel Maximal Biclique Enumeration. In *HiPC*. 34–43.

[15] Kemal Eren, Mehmet Deveci, Onur Küçüktunç, and Ümit V Çatalyürek. 2013. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in bioinformatics* 14, 3 (2013), 279–292.

[16] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., USA.

[17] Stephan Günnemann, Emmanuel Müller, Sebastian Raubach, and Thomas Seidl. 2011. Flexible Fault Tolerant Subspace Clustering for Data with Missing Values. In *ICDM*. 231–240.

[18] Dmitry I. Ignatov. 2019. Preliminary Results on Mixed Integer Programming for Searching Maximum Quasi-Bicliques and Large Dense Biclusters. In *ICFCA*, Vol. 2378. 28–32.

[19] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. 1999. Trawling the web for emerging cyber-communities. *Computer networks* 31, 11-16 (1999), 1481–1493.

[20] Sune Lehmann, Martin Schwartz, and Lars Kai Hansen. 2008. Biclique communities. *Phys. Rev. E* 78 (2008), 016108. Issue 1.

[21] Sune Lehmann, Martin Schwartz, and Lars Kai Hansen. 2008. Biclique communities. *Physical review E* 78, 1 (2008), 016108.

[22] Michael Ley. 2002. The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. In *SPIRE*, Vol. 2476. 1–10.

[23] Chu-Min Li, Zhiwen Fang, and Ke Xu. 2013. Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem. In *ICTAI*. 939–946.

[24] Chu Min Li and Zhe Quan. 2010. An Efficient Branch-and-Bound Algorithm Based on MaxSAT for the Maximum Clique Problem. In *AAAI*. 128–133.

[25] Haiquan Li, Jinyan Li, and Limsoon Wong. 2006. Discovering motif pairs at interaction sites from protein sequences on a proteome-wide scale. *Bioinform.* 22, 8 (2006), 989–996.

[26] Jinyan Li, Guimei Liu, Haiquan Li, and Limsoon Wong. 2007. Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms. *IEEE Trans. Knowl. Data Eng.* 19, 12 (2007), 1625–1637.

[27] Mingjie Li, Jin-Kao Hao, and Qinghua Wu. 2020. General swap-based multiple neighborhood adaptive search for the maximum balanced biclique problem. *Comput. Oper. Res.* 119 (2020), 104922.

[28] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2020. Efficient (α, β)-core computation in bipartite graphs. *VLDB J.* 29, 5 (2020), 1075–1099.

[29] Guimei Liu, Kelvin Sim, and Jinyan Li. 2006. Efficient Mining of Large Maximal Bicliques. In *DaWaK*, Vol. 4081. 437–448.

[30] Xiaowen Liu, Jinyan Li, and Lusheng Wang. 2008. Quasi-bicliques: Complexity and Binding Pairs. In *COCOON 2008*, Vol. 5092. 255–264.

[31] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2020. Maximum Biclique Search at Billion Scale. *Proc. VLDB Endow.* 13, 9 (2020), 1359–1372.

[32] Fragkiskos D. Malliaros, Christos Giatsidis, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. 2020. The core decomposition of networks: theory, algorithms and applications. *VLDB J.* 29, 1 (2020), 61–92.

[33] Pasin Manurangsi. 2017. Inapproximability of Maximum Edge Biclique, Maximum Balanced Biclique and Minimum k-Cut from the Small Set Expansion Hypothesis. In *ICALP (LIPIcs, Vol. 80)*. 79:1–79:14.

[34] Ciaran McCreesh and Patrick Prosser. 2014. An exact branch and bound algorithm with symmetry breaking for the maximum balanced induced biclique problem. In *CPAIOR*. 226–234.

[35] René Peeters. 2003. The maximum edge biclique problem is NP-complete. *Discret. Appl. Math.* 131, 3 (2003), 651–654.

[36] Ardian Kristanto Poernomo and Vivekanand Gopalkrishnan. 2009. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *KDD*. 697–706.

[37] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. 2006. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics* 22, 9 (2006), 1122–1129.

[38] Eran Shaham, Honghai Yu, and Xiaoli Li. 2016. On finding the maximum edge biclique in a bipartite graph: a subspace clustering approach. In *SIAM*. 315–323.

[39] Kelvin Sim, Jinyan Li, Vivekanand Gopalkrishnan, and Guimei Liu. 2009. Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market

and protein networks. *Stat. Anal. Data Min.* 2, 4 (2009), 255–273.
[40] Melih Sözdinler and Can C. Özturan. 2018. Finding Maximum Edge Biclique in Bipartite Networks by Integer Programming. In *CSE*. 132–137.
[41] Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. 2017. Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, Vol. 36. 201–225.
[42] Oliver Voggenreiter, Stefan Bleuler, and Wilhelm Gruissem. 2012. Exact biclustering algorithm for the analysis of large gene expression data sets. *BMC Bioinform.* 13, S-18 (2012), A10.
[43] Oliver Voggenreiter, Stefan Bleuler, and Wilhelm Gruissem. 2012. Exact biclustering algorithm for the analysis of large gene expression data sets. *BMC bioinformatics* 13, Suppl 18 (2012), A10.
[44] Jose L. Walteros and Austin Buchanan. 2020. Why Is Maximum Clique Often Easy in Practice? *Oper. Res.* 68, 6 (2020), 1866–1895.
[45] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. 2018. Deep Structure Learning for Fraud Detection. In *ICDM*. 567–576.
[46] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*. 501–508.
[47] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient Bitruss Decomposition for Large-scale Bipartite Graphs. In *ICDE*. 661–672.
[48] Kai Wang, Wenjie Zhang, Xuemin Lin, Lu Qin, and Alexander Zhou. 2022. Efficient Personalized Maximum Biclique Search. In *ICDE*. 498–511.
[49] Lusheng Wang. 2010. Near Optimal Solutions for Maximum Quasi-bicliques. In *COCOON*, Vol. 6196. 409–418.
[50] Nan Wang, Srinivasan Parthasarathy, Kian-Lee Tan, and Anthony K. H. Tung. 2008. CSV: visualizing and mining cohesive subgraphs. In *SIGMOD*, Jason Tsong-Li Wang (Ed.). 445–458.
[51] Kaiqiang Yu, Cheng Long, Shengxin Liu, and Da Yan. 2022. Efficient Algorithms for Maximal k-Biplex Enumeration. In *SIGMOD*. 860–873.
[52] Bo Yuan and Bin Li. 2014. A fast extraction algorithm for defect-free subcrossbar in nanoelectronic crossbar. *ACM J. Emerg. Technol. Comput. Syst (JETC)* 10, 3 (2014), 1–19.
[53] 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. *BMC Bioinform.* 15 (2014), 1–18.
[54] Feng Zhao and Anthony K. H. Tung. 2012. Large Scale Cohesive Subgraphs Discovery for Social Network Visual Analysis. *Proc. VLDB Endow.* 6, 2 (2012), 85–96.
[55] Yi Zhou and Jin-Kao Hao. 2019. Tabu search with graph reduction for finding maximum balanced bicliques in bipartite graphs. *Eng. Appl. Artif. Intell.* 77 (2019), 86–97.
[56] Yi Zhou, André Rossi, and Jin-Kao Hao. 2018. Towards effective exact methods for the maximum balanced biclique problem in bipartite graphs. *Eur. J. Oper. Res.* 269, 3 (2018), 834–843.
[57] Zhaonian Zou. 2016. Bitruss Decomposition of Bipartite Graphs. In *DASFAA*, Vol. 9643. 218–233.