Hereditary Cohesive Subgraphs Enumeration on Bipartite Graphs: The Power of Pivot-based Approaches

QIANGQIANG DAI, RONG-HUA LI, XIAOWEI YE, and MEIHAO LIAO, Beijing Institute of Technology, China
WEIPENG ZHANG, Tencent Technology (Shenzhen) Company Limited, China
GUOREN WANG, Beijing Institute of Technology, China

Finding cohesive subgraphs from a bipartite graph is a fundamental operator in bipartite graph analysis. In this paper, we focus on the problem of mining cohesive subgraphs from a bipartite graph that satisfy a hereditary property. Here a cohesive subgraph meets the hereditary property if all of its subgraphs satisfy the same property as itself. We show that several important cohesive subgraph models, such as maximal biclique and maximal k-biplex, satisfy the hereditary property. The problem of enumerating all maximal hereditary subgraphs was known to be NP-hard. To solve this problem, we first propose a novel and general pivot-based enumeration framework to efficiently enumerate all maximal hereditary subgraphs in a bipartite graph. Then, based on our general framework, we develop a new pivot-based algorithm with several pruning techniques to enumerate all maximal bicliques. We prove that the worst-case time complexity of our pivot-based maximal biclique enumeration algorithm is $O(m \times 2^{n/2})$ (or $O(m \times 1.414^n)$) which is near optimal since there exist up to $O(2^{n/2})$ maximal bicliques in a bipartite graph with n vertices and m edges. Moreover, we also show that our algorithm can achieve polynomial-delay time complexity with a slight modification. Third, on the basis of our general framework, we also devise a novel pivot-based algorithm with several non-trivial pruning techniques to enumerate maximal k-biplexes in a bipartite graph. Finally, we conduct extensive experiments using 11 real-world bipartite graphs to evaluate the proposed algorithms. The results show that our pivot-based solutions can achieve one order of magnitude (three orders of magnitude) faster than the state-of-the-art maximal biclique enumeration algorithms (maximal k-biplex enumeration algorithms).

CCS Concepts: • **Theory of computation → Backtracking**.

Additional Key Words and Phrases: hereditary cohesive subgraph, biclique, k-biplex, enumeration framework

ACM Reference Format:
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, Article 138 (June 2023), 26 pages. https://doi.org/10.1145/3589283

1 INTRODUCTION

Bipartite graphs are ubiquitous in real-world applications. In a bipartite graph, the vertices can be divided into two disjoint sets and each edge connects a vertex in one set to a vertex in the other set. Some representative examples of real-world bipartite graphs include user-item networks [43, 44], author-publication networks [22], and biological networks [23]. Real-world bipartite graphs often

Authors' addresses: Qiangqiang Dai, qiangd66@gmail.com; Rong-Hua Li, lironghuabit@126.com; Xiaowei Ye, yexiaowei@bit.edu.cn; Meihao Liao, mhliao@bit.edu.cn, Beijing Institute of Technology, Beijing, China; Weipeng Zhang, Tencent Technology (Shenzhen) Company Limited, Shenzhen, China, jackwpzhang@tencent.com; Guoren Wang, Beijing Institute of Technology, Beijing, China, wanggrbit@126.com.

2836-6573/2023/6-ART138 $15.00
https://doi.org/10.1145/3589283

contain cohesive subgraph structures which correspond to communities or densely-connected groups. Mining cohesive subgraphs from a bipartite graph is a fundamental operator in bipartite graph analysis which has been widely used in many applications, such as community detection [20, 22], online recommendation [17, 36], and biological network analysis [6, 42].

There exist many cohesive subgraph models in bipartite graphs. Notable examples include maximal biclique [1, 9, 14, 24, 28, 50], maximal k-biplex [40, 48, 49], (α, β)-core [7, 27], k-bitruss [45, 52], and quasi biclique [19, 30, 46]. Among them, the maximal biclique model, perhaps, is the most fundamental model, as all the other cohesive subgraph models can be considered as a *relaxed* biclique model.

Instead of focusing on a particular cohesive subgraph model, in this paper, we study a family of cohesive subgraph models in bipartite graphs that meet the hereditary property, namely hereditary cohesive subgraphs. Here a subgraph G' is called a hereditary subgraph if (1) G' satisfies a property $\mathcal{P}$ and (2) every induced subgraph of G' also meets the property $\mathcal{P}$. For convenience, we refer to a subgraph that meets a hereditary property $\mathcal{P}$ as a $\mathcal{P}$-subgraph. A $\mathcal{P}$-subgraph G' is a maximal $\mathcal{P}$-subgraph if there is no other $\mathcal{P}$-subgraph containing G'. Given a bipartite graph G, our goal is to enumerate all maximal $\mathcal{P}$-subgraphs from G. To our knowledge, such a maximal $\mathcal{P}$-subgraph enumeration problem has not been investigated before. We show that both the maximal biclique and maximal k-biplex are maximal $\mathcal{P}$-subgraphs, thus both the classic maximal biclique enumeration and maximal k-biplex enumeration problems are special instances of our problem.

Motivations. Practical solutions for maximal $\mathcal{P}$-subgraph enumeration can be applied to many applications, and two of them are summarized as follows.

Community detection in bipartite graphs. Detecting communities in a bipartite graph is an important graph analysis task [6, 20, 23, 36, 42]. We can use the hereditary cohesive subgraph to model communities in a bipartite graph. The communities detected by the hereditary cohesive subgraph model often exhibit strong robustness, since the removal of any subset of vertices from the community does not destroy the structural property. In effect, the classic maximal biclique and maximal k-biplex models have been widely used for community detection applications [20, 28, 40, 48] due to such a nice hereditary property and cohesive property. Thus, the solution for maximal $\mathcal{P}$-subgraph enumeration can provide a general framework for community detection in bipartite graphs, which captures a family of different community models.

Fraud detection in user-item networks. Consider an online user-item rating network (e.g., Amazon's user-product rating network), where the users can give ratings to the items. The item owners may wish to improve their items' ratings by hiring some fake users to frequently give high ratings to their items. Clearly, the set of fake users and the set of their rated items often form a densely-connected subgraph. Once again, we can use the hereditary cohesive subgraph model, such as maximal biclique or maximal k-biplex to detect such fake users [48]. As a result, the approaches to maximal $\mathcal{P}$-subgraph enumeration can also be used for identifying possible rating frauds in online user-item networks.

Although the significance of the maximal $\mathcal{P}$-subgraph enumeration problem, a practical solution for this problem is still lacking due to the intrinsic challenges of this problem. First, as indicated in [21], the problem of enumerating all maximal $\mathcal{P}$-subgraphs from a bipartite graph is NP-hard. Thus, there does not exist a polynomial algorithm to solve this problem unless NP = P. Second, since the hereditary property $\mathcal{P}$ is arbitrary (the enumeration algorithm for our problem should work for any hereditary property $\mathcal{P}$) and internal structure of the $\mathcal{P}$ is unclear, it is quite non-trivial to use such a property $\mathcal{P}$ to design an enumeration algorithm. Moreover, existing solutions for maximal biclique enumeration and maximal k-biplex enumeration also cannot be generalized to handle our problem. This is because different properties $\mathcal{P}$ give rise to different enumeration problems,

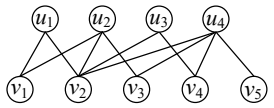


Fig. 1. Running example: a bipartite graph G

and enumeration techniques for a specified $\mathcal{P}$ are unlikely to work for every hereditary subgraph enumeration problem. For example, the techniques for maximal biclique enumeration are often hard to extend to solve the problem of maximal k-biplex enumeration [48], because maximal bicliques can be enumerated within the 2-hop neighborhood of each vertex, while maximal k-biplex does not share such a nice property. As a result, new techniques need to be developed to solve our problem.

Contributions. In this paper, we formulate and develop efficient algorithms to enumerate maximal hereditary subgraphs (maximal $\mathcal{P}$-subgraphs) on bipartite graphs, with special focus on enumerating maximal bicliques and maximal k-biplexes which are two representative maximal hereditary subgraphs on bipartite graphs. In summary, the main contributions of this paper are as follows.

A novel and general pivot-based framework. To solve the maximal $\mathcal{P}$-subgraph enumeration problem, we first devise a basic algorithm inspired by the classic set enumeration technique [5, 38]. Such a basic solution, however, may explore all possible subsets of vertices, thus resulting in many unnecessary computations. To improve the efficiency, we develop a novel and general pivot-based backtracking framework to enumerate all maximal $\mathcal{P}$-subgraphs. Our framework is based on the basic set enumeration technique together with a novel and carefully-designed pivoting strategy. With such a powerful pivoting technique, our framework can significantly prune redundant computations in the enumeration procedure. To our knowledge, the proposed framework is the first practical solution for enumerating all maximal $\mathcal{P}$-subgraphs in bipartite graphs. In addition, our framework is very general which can provide useful guidelines to design practical solutions for enumerating specified maximal $\mathcal{P}$-subgraphs (e.g., maximal biclique and maximal k-biplex).

New maximal biclique enumeration algorithms. We propose a new maximal biclique enumeration algorithm based on our general pivoting principle. The striking feature of our algorithm is that its worst-case time complexity is near optimal. Specifically, we prove that the time complexity of our pivot-based algorithm is $O(m \times 2^{n/2})$ (or $O(m \times 1.414^n)$) which is near optimal, because there exist up to $O(2^{n/2})$ maximal bicliques on a bipartite graph with n vertices and m edges [37]. Moreover, with a slight modification, our pivot-based algorithm can achieve polynomial-delay time complexity. In addition, we also present several non-trivial optimization techniques (including early termination and ordering techniques) to further improve the efficiency of our algorithm.

Novel maximal k-biplex enumeration algorithms. Based on our general pivoting framework, we also develop a novel pivot-based algorithm to enumerate all maximal k-biplexes. Note that the detailed implementation of our pivoting technique for maximal k-biplex enumeration is significantly different from that for maximal biclique enumeration, although both of them are based on our general pivoting principle. To our knowledge, this is the first pivot-based enumeration algorithm for maximal k-biplex enumeration. In addition, we also develop several non-trivial pruning techniques to further improve the efficiency of our algorithm when enumerating large maximal k-biplexes (with size no less than a given threshold).

Extensive experiments. We conduct extensive experiments to evaluate the efficiency and effectiveness of the proposed approaches using 11 real-world bipartite graphs. The results show that our pivot-based algorithms are one order of magnitude faster than the state-of-the-art algorithm for maximal biclique enumeration, and three orders of magnitude faster than the state-of-the-art algorithms for maximal k-biplex enumeration. In addition, two representative applications on fraud detection and community detection demonstrate the high effectiveness of our solutions. For reproducibility purposes, the source code of this work is available at https://github.com/qq-dai/BiHSE.

2 PROBLEM STATEMENT

Let $G = (U, V, E)$ be an undirected and unweighted bipartite graph with two disjoint vertex sets U and V and an edge set $E \subseteq U \times V$. Denote by $n = |U| + |V|$ and $m = |E|$ the number of vertices and edges in G, respectively. For a vertex $u \in U$, we define $N_u(G)$ as the set of neighbors of u in G, i.e., $N_u(G) = \{w \in V | (u, w) \in E\}$. The degree of a vertex $u \in U$ in G is denoted by $d_u(G) = |N_u(G)|$. Similar definitions are also applied for the vertices in V. Given a pair (A, B) of vertex sets with $A \subseteq U$ and $B \subseteq V$, we define $G(A, B) = (A, B, E')$ as the induced subgraph of G, where $E' = \{(u, v) \in E | u \in A, v \in B\}$. Denote by $\mathcal{P}$ a graph property. Then, the hereditary property for bipartite graphs is defined as follows.

Definition 1 (Hereditary property). Given a bipartite graph G and a random subgraph H of G with a graph property $\mathcal{P}$, $\mathcal{P}$ is said to be hereditary if every subgraph of H meets $\mathcal{P}$.

Based on Definition 1, the maximal $\mathcal{P}$-subgraph of a bipartite graph G is defined as follows.

Definition 2 (Maximal $\mathcal{P}$-subgraph). Given a bipartite graph G and a fixed hereditary property $\mathcal{P}$, a subgraph H of G is called a maximal $\mathcal{P}$-subgraph if 1) H meets the property $\mathcal{P}$, and 2) there is no other subgraph H' of G containing H and also meeting $\mathcal{P}$.

Two notable instances of maximal $\mathcal{P}$-subgraphs on bipartite graphs are maximal biclique [28] and maximal k-biplex [34], which have been widely studied in the literature [1, 9, 13, 14, 24, 48, 50]. Below, we give the formal definitions of these two models.

Definition 3 (Maximal biclique). Given a bipartite graph $G = (U, V, E)$, a subgraph $H = G(A, B)$ induced by a pair of vertex sets (A, B) is a maximal biclique of G if 1) $\forall u \in A, v \in B$, there is an edge $(u, v) \in E$, and 2) there is no other subgraph H' that contains H and satisfies 1).

Definition 4 (k-biplex). Given a bipartite graph $G = (U, V, E)$, a subgraph $H = G(A, B)$ induced by a pair of sets (A, B) in G is a k-biplex, if in the subgraph H, every vertex $u \in A$ has a degree no less than $|B| - k$, and every vertex $v \in B$ has a degree no less than $|A| - k$.

Definition 5 (Maximal k-biplex). A k-biplex $G(A, B)$ of G is called a maximal k-biplex if there is no other k-biplex $G(A', B')$ of G that contains $G(A, B)$.

The following lemma shows that both maximal biclique and maximal k-biplex are maximal $\mathcal{P}$-subgraphs. Due to space limitations, this paper omits the proofs that can be easily derived.

Lemma 1. Both biclique and k-biplex (for any integer k) meet the hereditary property.

Due to the nice hereditary property and the internal cohesiveness of these two hereditary subgraph instances, both maximal biclique and maximal k-biplex are widely used in bipartite graph analysis applications such as community detection [20, 28], fraud detection [48], and text mining [32, 39]. Thus, it is important to develop efficient algorithms to solve the problem of enumerating all maximal $\mathcal{P}$-subgraphs on bipartite graphs.

Problem definition. Given a bipartite graph G, our goal is to (1) develop a general framework to efficiently enumerate all maximal $\mathcal{P}$-subgraph of G for any fixed hereditary property $\mathcal{P}$, and (2) apply the proposed framework to enumerate all maximal bicliques and maximal k-biplexes on G respectively.

As shown by Yannakakis and Lewis [21], the problem of enumerating all maximal $\mathcal{P}$-subgraphs on G is NP-hard for any given hereditary property $\mathcal{P}$. To effectively address this problem, in this work, we develop a novel and general pivot-based enumeration framework, which can significantly prune enumeration space by a pivoting technique. In the following sections, we will first introduce our pivot-based framework, followed by two new solutions for enumerating all maximal bicliques and k-biplexes.

Algorithm 1: A basic set enumeration framework

Input: The bipartite graph G
Output: All maximal $\mathcal{P}$-subgraphs of G
1 $Enum(\emptyset, \emptyset, U, V, \emptyset, \emptyset)$;
2 **Function:** $Enum(R_U, R_V, C_U, C_V, X_U, X_V)$
3 **if** $C_U \cup C_V = \emptyset$ **then**
4 **if** $X_U \cup X_V = \emptyset$ **then** Output (R_U, R_V) as a result;
5 **return**;
6 **foreach** $w \in C_U$ **do**
7 $Branch(R_U, R_V, w, C_U \setminus \{w\}, C_V, X_U, X_V)$;
8 $C_U \leftarrow C_U \setminus \{w\}$; $X_U \leftarrow X_U \cup \{w\}$;
9 **foreach** $w \in C_V$ **do**
10 $Branch(R_V, R_U, w, C_V \setminus \{w\}, C_U, X_V, X_U)$;
11 $C_V \leftarrow C_V \setminus \{w\}$; $X_V \leftarrow X_V \cup \{w\}$;
12 **Function:** $Branch(R_U, R_V, w, C_U, C_V, X_U, X_V)$
13 Generate sets $C'_U \subseteq C_U, C'_V \subseteq C_V, X'_U \subseteq X_U$, and $X'_V \subseteq X_V$ such that $(R_U \cup \{w, u\}, R_V)$ and $(R_U \cup \{w\}, R_V \cup \{v\})$ are the partial results, where $u \in C'_U \cup X'_U$ and $v \in C'_V \cup X'_V$;
14 $Enum(R_U \cup \{w\}, R_V, C'_U, C'_V, X'_U, X'_V)$;

3 A GENERAL PIVOT-BASED FRAMEWORK

In this section, we first propose a basic set-enumeration technique to enumerate all maximal $\mathcal{P}$-subgraphs, based on which we then develop a general pivot-based enumeration framework.

3.1 The Basic Set Enumeration Framework

Our basic enumeration algorithm is inspired by the classic set enumeration technique [5, 38]. The key idea of our basic technique is that it makes use of a recursive approach to explore each subgraph of a given bipartite graph G, and then determines whether each explored subgraph is a valid maximal $\mathcal{P}$-subgraph. To achieve this, the algorithm needs to maintains three kinds of sets in each recursive call, called *the current partial result*, *candidate sets*, and *exclusion sets* respectively to direct the backtracking search. The detailed implementation is shown in Algorithm 1.

In Algorithm 1, it invokes the procedure *Enum* to enumerate all maximal $\mathcal{P}$-subgraphs (line 1), which requires six parameters: R_U, R_V, C_U, C_V, X_U, and X_V. Here the vertex sets (R_U, R_V) represent the current partial result, (C_U, C_V) are the candidate sets in which each vertex can be used to expand (R_U, R_V), and (X_U, X_V) are the exclusion sets containing vertices in (C_U, C_V) that have already been used to expand (R_U, R_V). Initially, both the sets (R_U, R_V) and (C_U, C_V) are set to empty, while (C_U, C_V) is set to (U, V) (line 1). Then, in each recursion, every vertex in (C_U, C_V) is used to expand the current partial result (R_U, R_V) (lines 6-11). When a vertex w in (C_U, C_V) is added into (R_U, R_V), the algorithm needs to update the candidate and exclusion sets (line 13). The algorithm further invokes a new sub-recursive call to enumerate the maximal results containing $(R_U \cup \{w\}, R_V)$ (or $(R_U, R_V \cup \{w\})$) (line 14). After processing the vertex w, w will be moved from C_U (or C_V) to X_U (or X_V) to avoid outputting non-maximal $\mathcal{P}$-subgraphs (line 8 and line 11). Specifically, if X_U (or X_V) is not empty, there must exist previously-processed vertices that can be used to expand the current (R_U, R_V). This implies that (R_U, R_V) cannot be the maximal result even if there is no vertex in (C_U, C_V). Thus, whenever both (C_U, C_V) and (X_U, X_V) are empty, the algorithm outputs the current (R_U, R_V) as a maximal $\mathcal{P}$-subgraph (lines 3-4).

It is easy to show that the worst-case time complexity of Algorithm 1 is $O(f(n) \times 2^n)$ ($f(n)$ denotes the time consumed to generate the candidate sets and exclusion sets), because the algorithm may traverse all possible subsets of the vertex set. For example, if G itself is a $\mathcal{P}$-subgraph, every possible subgraph of G will be explored by the algorithm. To improve Algorithm 1, we next propose a pivot-based enumeration framework that can prune many redundant computations.

3.2 Novel Pivot-based Enumeration Framework

The key to speeding up Algorithm 1 is to devise a pruning technique to reduce the unnecessary computations that produce non-maximal $\mathcal{P}$-subgraphs. However, it is quite non-trivial to achieve this, because the graph property $\mathcal{P}$ is arbitrary and the internal structure of the $\mathcal{P}$-subgraph is unclear. Our solution to achieve this is based on an in-depth analysis of Algorithm 1.

An in-depth analysis. Given a bipartite graph G and the candidate sets (C_U, C_V), suppose that (A, B) is a maximal $\mathcal{P}$-subgraph of G with $(A, B) \subseteq (C_U, C_V)$ (i.e, $A \subseteq C_U$ and $B \subseteq C_V$). In the top recursion of Algorithm 1, each vertex in (C_U, C_V) is used to expand the initial result $(\emptyset, \emptyset)$. It is easy to see that the result (A, B) can be detected by a recursive call that enumerates all maximal $\mathcal{P}$-subgraphs containing $u \in C_U$ if $u \in A$. When finishing this computation, the vertex u is removed from C_U and the vertices in $(C_U \setminus \{u\}, C_V)$ are further selected to continue the recursive calls. If the next selected vertex $v \in C_V$ satisfies $v \in B$, then all non-maximal $\mathcal{P}$-subgraphs contained in $(A \setminus \{u\}, B)$ can also be enumerated, thus causing many redundant computations. To reduce this, we can select another vertex v' in C_V with $v' \notin B$ to enumerate all maximal $\mathcal{P}$-subgraphs containing v' in $(C_U \setminus \{u\}, C_V)$. Since $v' \notin B$, each $\mathcal{P}$-subgraph (A', B') with $v' \in B'$ enumerated by such a sub-recursive call must be excluded in $(A \setminus \{u\}, B)$. Thus, we can avoid enumerating non-maximal $\mathcal{P}$-subgraphs contained in $(A \setminus \{v\}, B)$. Moreover, when all vertices in $(C_U \setminus (A \setminus \{u\}), C_V \setminus B)$ have been used to expand the initial result $(\emptyset, \emptyset)$, all the remaining vertices in the candidate sets are exactly $(A \setminus \{u\}, B)$, which does not contain any maximal $\mathcal{P}$-subgraph of G since (A, B) is a maximal $\mathcal{P}$-subgraph. As a consequence, we can terminate those sub-recursive calls. In other words, all vertices in $(A \setminus \{u\}, B)$ are not necessary to expand the current partial result. Based on this analysis, we present a general pivoting technique as described in the following theorem.

Theorem 3.1 (General Pivoting Rule). *Consider a recursion with six parameters $(R_U, R_V, C_U, C_V, X_U, X_V)$. Let $u \in C_U \cup X_U$ be a pivot vertex. Then, the vertices in $(P_U \subseteq C_U, P_V \subseteq C_V)$ can be skipped to expand (R_U, R_V) if and only if any result containing (R_U, R_V) but not u is not contained in $G(R_U \cup P_U, R_V \cup P_V)$.*

Note that a symmetrical pivot vertex v in $C_V \cup X_V$ can also be selected by Theorem 3.1. For simplicity, in the rest of this paper, we mainly discuss the pivot vertex which is selected from $C_U \cup X_U$, because the other pivot in $C_V \cup X_V$ can be obtained similarly. Then, based on the pivoting rule shown in Theorem 3.1, we can derive the following result, which provides guidelines for pivot-based branching.

Theorem 3.2. *Given a pivot vertex $u \in C_U \cup X_U$ and the skipping sets $(P_U \subseteq C_U, P_V \subseteq C_V)$, then any maximal $\mathcal{P}$-subgraph containing (R_U, R_V) must belong to one of the following three cases.*

(1) It contains the vertex u.
(2) It does not contain u, but contains at least one vertex in $C_V \setminus P_V$.
(3) It does not contain vertices in $\{u\} \cup C_V \setminus P_V$, but contains at least one vertex in $C_U \setminus (P_U \cup \{u\})$.

Implementation details. Algorithm 2 details our pivot-based enumeration framework. Note that Algorithm 2 is similar to Algorithm 1. The key difference is that Algorithm 2 uses the proposed pivoting technique (Theorem 3.1) to prune unnecessary sub-recursive calls. Specifically, the algorithm first selects two pivot vertices $u \in C_U \cup X_U$ and $v \in C_V \cup X_V$, and constructs two pairs of skipping sets (P_U, P_V) and (P'_U, P'_V) according to Theorem 3.1 (lines 6-7). When the size of $P'_U \cup P'_V$ is larger than that of $P_U \cup P_V$, the algorithm utilizes the skipping sets (P'_U, P'_V) to reduce enumeration branches (line 8), because in this case more recursive calls can be reduced by the pivot vertex v. Then, the algorithm recursively expands the current result (R_U, R_V) with each vertex in (C_U, C_V), but not in (P_U, P_V) (lines 9-14). The following theorem shows the correctness of Algorithm 2.

Algorithm 2: The pivot-based enumeration framework

Input: The bipartite graph G
Output: All maximal $\mathcal{P}$-subgraphs of G

$PivotEnum(\emptyset, \emptyset, U, V, \emptyset, \emptyset)$;
Function: $PivotEnum(R_U, R_V, C_U, C_V, X_U, X_V)$
  if $C_U \cup C_V = \emptyset$ **then**
    if $X_U \cup X_V = \emptyset$ **then** Output (R_U, R_V) as a result;
    return;
  Derive the skipping sets ($P_U \subseteq C_U, P_V \subseteq C_V$) by the pivot vertex u (Theorem 3.1) that is selected from $C_U \cup X_U$;
  Derive the skipping sets ($P'_U \subseteq C_U, P'_V \subseteq C_V$) by the pivot vertex v (Theorem 3.1) that is selected from $C_V \cup X_V$;
  if $|P'_U| + |P'_V| > |P_U| + |P_V|$ **then** $P_U \leftarrow P'_U; P_V \leftarrow P'_V$;
  foreach $w \in C_U \setminus P_U$ **do**
    $PivotBranch(R_U, R_V, w, C_U \setminus \{w\}, C_V, X_U, X_V)$;
    $C_U \leftarrow C_U \setminus \{w\}; X_U \leftarrow X_U \cup \{w\}$;
  foreach $w \in C_V \setminus P_V$ **do**
    $PivotBranch(R_V, R_U, w, C_V \setminus \{w\}, C_U, X_V, X_U)$;
    $C_V \leftarrow C_V \setminus \{w\}; X_V \leftarrow X_V \cup \{w\}$;
Function: $PivotBranch(R_U, R_V, w, C_U, C_V, X_U, X_V)$
  Generate sets $C'_U \subseteq C_U, C'_V \subseteq C_V, X'_U \subseteq X_U$, and $X'_V \subseteq X_V$ such that $(R_U \cup \{w, u\}, R_V)$ and $(R_U \cup \{w\}, R_V \cup \{v\})$ are the partial results, where $u \in C'_U \cup X'_U$ and $v \in C'_V \cup X'_V$;
  $PivotEnum(R_U \cup \{w\}, R_V, C'_U, C'_V, X'_U, X'_V)$;

Theorem 3.3. *Algorithm 2 correctly computes all maximal $\mathcal{P}$-subgraphs of a bipartite graph G.*

Clearly, the time complexity of Algorithm 2 can also be bounded by $O(f(n)2^n)$. However, compared to Algorithm 1, many unnecessary recursive calls can be pruned by Algorithm 2 with the pivoting technique, which results in much better performance. Moreover, such a pivot-based enumeration framework is very general which can guide us to design novel pivot-based algorithms for enumerating any maximal subgraph that satisfies the hereditary property. The following theorem shows the space complexity of Algorithm 2.

Theorem 3.4. *The space complexity of Algorithm 2 is $O(\triangle n + m)$, where $\triangle$ is the maximum size of the $\mathcal{P}$-subgraph in G.*

Proof. It is easy to see that the algorithm uses $O(n)$ space in each recursion, and the maximum depth of the recursions is bounded by $O(\triangle)$. Then, based on the depth-first backtrack search, we can derive that the space complexity of Algorithm 2 is $O(\triangle n + m)$. □

In the following sections, we will show how to use our general framework to devise new and efficient algorithms for enumerating maximal bicliques and maximal k-biplexes.

4 MAXIMAL BICLIQUE ENUMERATION

Recall that maximal biclique is a special instance of maximal $\mathcal{P}$-subgraph. In this section, we aim to develop new algorithms to enumerate all maximal bicliques based on the proposed pivoting framework. Below, we first briefly review existing maximal biclique enumeration techniques, and then present our solutions.

4.1 Overview of Existing Algorithms

The maximal biclique enumeration problem has been extensively studied in literature [1, 9, 13, 14, 24, 28, 50]. Note that all these existing solutions are fundamentally different from our proposed pivot-based framework (Algorithm 2). Specifically, all existing solutions are based on the following fact. Given a subset set $S \in U$, let $N(S)$ be the common neighbors of S in V, i.e., $N(S) = \cap_{u \in S} N_u(G)$. Then, if $N(S) \neq \emptyset$, $(N(N(S)), N(S))$ must be a maximal biclique. Therefore, all maximal bicliques can be enumerated by exploring the combinations of vertices on the one side [24, 50]. However,

unlike these algorithms, our pivot-based framework (Algorithm 2) considers both sides of vertices to expand the partial biclique.

To the best of our knowledge, the state-of-the-art algorithm for maximal biclique enumeration is developed in [9], which improves a so-called pivot-based approach presented in [1] with a 2-hop degree ordering optimization technique. Such a pivot-based approach is mainly based on a neighborhood dominating technique. Specifically, for two vertices u and u' in U, the vertex u' is dominated by u if $N_{u'}(G) \subseteq N_u(G)$. Then, if both u and u' are included in the candidate sets of a recursive call, the vertex u' is not necessary to be used to expand the current partial biclique. This technique [9], however, needs to compute the neighborhood domination relationships between every vertex and all its 2-hop neighbors which is often costly on large bipartite graphs. Moreover, real-world bipartite graphs do not necessarily contain too many *dominated* vertices, thus the pruning power of this technique may be poor on some real-world bipartite graphs (as indicated in our experiments). To address this problem, we will propose a novel and more efficient pivoting technique based on our general framework.

4.2 Pivot-based Maximal Biclique Enumeration

Following the general pivot-based framework (Algorithm 2), our pivot-based maximal biclique enumeration algorithm also admits six parameters $(R_U, R_V, C_U, C_V), X_U, X_V)$, where (R_U, R_V), (C_U, C_V), and (X_U, X_V) represents the current biclique, the candidate sets, and the exclusion sets respectively. Note that to apply our general framework to maximal biclique enumeration, the key is to determine the skipping sets P_U and P_V using the pivot vertex. Below, we first derive a lemma that can be used to determine P_U.

Lemma 2. If $u \in C_U \cup X_U$ is a pivot vertex, all vertices in $C_U \setminus \{u\}$ can be omitted to expand the current biclique (R_U, R_V).

According to Lemma 2, all vertices in C_V need to be used to expand the current biclique. However, this can still generate many redundant computations. For instance, suppose that (A, B) is a maximal biclique of G with $A = R_U \cup \{u\}$ and $R_V \subseteq B \subseteq (R_V \cup C_V)$, where u is a pivot vertex in C_U. After obtaining all maximal bicliques containing u, if we select any vertex $v \in B \setminus R_V$ to expand (R_U, R_V), a non-maximal biclique (R_U, B) is generated by the algorithm, incurring redundant computations. To overcome this issue, we also need to skip some vertices in C_V to enumerate all maximal bicliques (i.e., P_V is not empty), thus we present the following lemma.

Lemma 3. Given a pivot vertex $u \in C_U \cup X_U$ and the skipping sets $(P_U \subseteq C_U, P_V \subseteq C_V)$, if any biclique (A, B) with $R_U \subset A$ and $R_V \subseteq B \subseteq (R_V \cup P_V)$ can be enlarged by u, then there is no maximal biclique in G that excludes all vertices in $\{u\} \cup C_V \setminus P_V$ but includes at least one vertex in $P_U = C_U \setminus \{u\}$ (i.e., no maximal biclique in G belongs to the case (3) in Theorem 3.2).

Armed with Lemma 3, we can always set the skipping set P_U as $C_U \setminus \{u\}$ if u is the pivot vertex in $C_U \cup X_U$. Then, the remaining issue is to determine the skipping set P_V, satisfying that each biclique (A, B) with $R_V \subseteq B \subseteq (R_V \cup P_V)$ can be enlarged by u. Note that to achieve this, a prerequisite is that P_V must be included in the neighborhood of u. Otherwise, the biclique (A, B) cannot be extended by u, since there is a vertex $v \in B$ with $v \notin N_u(G)$. Based on this, we can obtain the following result.

Theorem 4.1 (Biclique Pivoting Rule). *Let $u \in C_U \cup X_U$ be a pivot vertex. The vertices in (P_U, P_V) can be skipped to expand (R_U, R_V), where $P_U = C_U \setminus \{u\}$ and $P_V = C_V \cap N_u(G)$.*

Based on Algorithm 2 and the pivoting rule established in Theorem 4.1, It is easy to devise a pivot-based algorithm for maximal biclique enumeration as outlined in Algorithm 3. Note that to achieve

Algorithm 3: Pivot-based maximal biclique enumeration

Input: The bipartite graph G
Output: All maximal bicliques of G

1 $BicliqueEnum(\emptyset, \emptyset, U, V, \emptyset, \emptyset)$;
2 **Function:** $BicliqueEnum(R_U, R_V, C_U, C_V, X_U, X_V)$
3 **if** $C_U \cup C_V = \emptyset$ **then**
4 **if** $X_V = \emptyset$ *and* $X_U = \emptyset$ **then** Output (R_U, R_V) as a result ;
5 **return**;
6 Select a pivot vertex u from $C_U \cup X_U$ that maximizes $|P_U| + |P_V|$, where $P_U = C_U \setminus \{u\}$ and $P_V = C_V \cap N_u(G)$;
7 Select a pivot vertex v from $C_V \cup X_V$ that maximizes $|P'_U| + |P'_V|$, where $P'_U = C_U \cap N_v(G)$ and $P'_V = C_V \setminus \{v\}$;
8 **if** $|P'_U| + |P'_V| > |P_U| + |P_V|$ **then** $P_U \leftarrow P'_U$; $P_V \leftarrow P'_V$;
9 **foreach** $u' \in C_U \setminus P_U$ **do**
10 $BicliqueBranch(R_U, R_V, u', C_U \setminus \{u'\}, C_V, X_U, X_V)$;
11 $C_U \leftarrow C_U \setminus \{u'\}$; $X_U \leftarrow X_U \cup \{u'\}$;
12 **foreach** $v' \in C_V \setminus P_V$ **do**
13 $BicliqueBranch(R_V, R_U, v', C_V \setminus \{v'\}, C_U, X_V, X_U)$;
14 $C_V \leftarrow C_V \setminus \{v'\}$; $X_V \leftarrow X_V \cup \{v'\}$;
15 **Function:** $BicliqueBranch(R_U, R_V, u, C_U, C_V, X_U, X_V)$
16 $C'_V \leftarrow C_V \cap N_u(G)$; $X'_V \leftarrow X_V \cap N_u(G)$;
17 $BicliqueEnum(R_U \cup \{u\}, R_V, C_U, C'_V, X_U, X'_V)$;

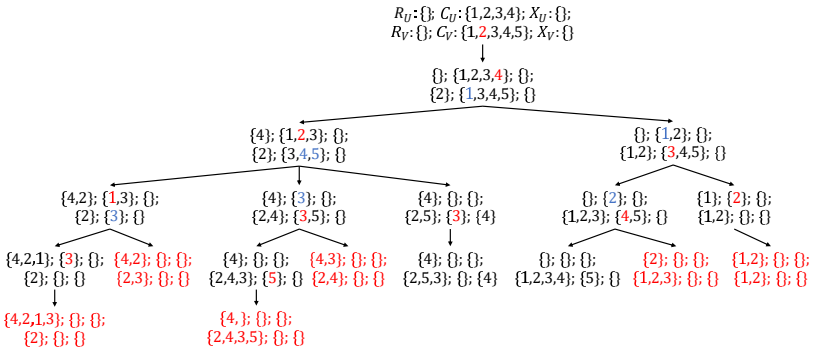


Fig. 2. The enumeration tree of the pivot-based maximal biclique enumeration algorithm (red vertices are the pivot vertices, red branches are maximal bicliques, and blue vertices are the non-neighbors of pivot vertices).

good performance, Algorithm 3 always selects a pivot vertex u (v) from $C_U \cup X_U$ ($C_V \cup X_V$) such that $|P_U| + |P_V|$ is maximum (lines 6-7). The following example illustrates the idea of Algorithm 3.

Example 1. Consider the bipartite graph G shown in Fig. 1. Let R_U, R_V, C_U, C_V, X_U, and X_V be the six parameters invoked by Algorithm 3. Initially, the sets R_U, R_V, X_U, and X_V are set to empty, and the sets C_U and C_V are set to U and V, respectively. In the first recursion, the vertex $v_2 \in C_V$ is selected as the pivot vertex, and thus only vertex v_2 is used to expand the current biclique $(\emptyset, \emptyset)$ based on the pivoting rule (Theorem 4.1). Then, in the recursion with the current biclique $(\emptyset, \{v_2\})$, a pivot vertex $u_4 \in C_U$ is further selected; and we can obtain that the vertices $(\{u_4\}, \{v_1\})$ are used to expand $(\emptyset, \{v_2\})$. If u_4 is pushed into $(\emptyset, \{v_2\})$, the similar pivoting rule will be used to enumerate all maximal bicliques that contains $(\{u_4\}, \{v_2\})$. After that, the vertex v_1 is pushed into $(\emptyset, \{v_2\})$ to continue the recursions. Finally, the algorithm terminates if no vertex is left to expand the current biclique in each recursion. The complete enumeration tree of Algorithm 3 is shown in Fig. 2.

By Theorem 3.3 and Theorem 4.1, it is easy to show that Algorithm 3 correctly enumerates all maximal bicliques. The following theorem shows the time complexity of Algorithm 3.

Theorem 4.2. *The worst-case time complexity of Algorithm 3 is $O(m \times 2^{n/2})$ (or $O(m \times 1.414^n)$).*

Proof. Denote by $T(n)$ the maximum number of recursive calls of *BicliqueEnum* $(\emptyset, \emptyset, U, V, \emptyset, \emptyset)$, where $n = |C_U| + |C_V|$. Suppose that $k = \min\{\overline{d}_u(C_V), \overline{d}_v(C_U)\}$, where $u \in C_U \cup X_U$ ($v \in C_V \cup X_V$) and $\overline{d}_u(C_V) = C_V \setminus N_u(G)$ ($\overline{d}_v(C_U) = C_U \setminus N_v(G)$). Next, we analyze recurrence relations of $T(n)$.

Assume that the optimal pivot vertex u is selected from $C_U \cup X_U$. We then have $k = \overline{d}_u(C_V)$ and $\overline{d}_v(C_U) \geq k$ for every $v \in C_V \setminus N_u(G)$. Based on the pivoting technique (Theorem 4.1), only the vertex u and each vertex in $C_V \setminus N_u(G)$ is used to expand (R_U, R_V) in the recursions. When vertex u is pushed into (R_U, R_V), we notice that the size of the corresponding candidate sets is at most $n-k-1$, since u has k non-neighbors in C_V. Moreover, when pushing the i-th vertex in $C_V \setminus N_u(G)$ into (R_U, R_V), the size of its corresponding candidate sets is at most $n-k-i$ since $\overline{d}_v(C_U) \geq k$ for every $v \in C_V \setminus N_u(G)$. Thus, the following recurrence relations of $T(n)$ can be obtained:

$$T(n) \leq T(n-k-1) + \sum_{i=1}^{k} T(n-k-i), \text{ if } u \in C_U, \tag{1}$$

$$T(n) \leq \sum_{i=1}^{k} T(n-k-i), \text{ if } u \in X_U. \tag{2}$$

Clearly, the worst-case bound of Eq. (2) is smaller than that of Eq. (1). Thus, in the following, we mainly analyze the worst-case bound of Eq. (1) by considering four different cases.

Case $k = 0$: In this case, $T(n) = T(n-1)$. Obviously, the size of $T(n)$ is a constant.

Case $k = 1$: By Eq. (1), we have: $T(n) \leq T(n-1-1) + T(n-1-1) = 2T(n-2)$. It is also easy to derive that $T(n)$ is bounded by $O(2^{n/2})$.

Case $k = 2$: In this case, we have: $T(n) \leq T(n-2-1) + \sum_{i=1}^{2} T(n-2-i) = T(n-3) + T(n-3) + T(n-4)$. As proved in [16], for a linear recursion function of $F(n) = \sum_{i=1}^{j} F(n-a_i)$, $F(n)$ is bounded by $O(\alpha^n)$, where α is the maximum real root of the equation $x^n - \sum_{i=1}^{j} x^{n-a_i} = 0$. By setting $a_1 = a_2 = 3$ and $a_3 = 4$, $T(n)$ can be bounded by $O(\alpha^n)$. Here α is the maximum real root of the equation $x^{n-4}(x^4 - x - x - 1) = 0$ which can be easily shown that $\alpha < \sqrt{2}$. As a result, we have $T(n) \leq 2^{n/2}$.

Case $k \geq 3$: In this case, we have: $T(n) \leq T(n-k-1) + \sum_{i=1}^{k} T(n-k-1) = (k+1)T(n-k-1) = (k+1)^{\frac{n}{k+1}} T(n - (k+1) \times \frac{n}{k+1})$. To bound $T(n)$, it is sufficient to derive a bound for $(k+1)^{\frac{n}{k+1}}$. Note that $f(x) = x^{\frac{n}{x}}$ is a monotonically decreasing function with $x \geq 4$. Since $k+1 \geq 4$, we have $(k+1)^{\frac{n}{k+1}} \geq 4^{\frac{n}{4}}$. As a result, we have $T(n) = O(4^{n/4}) = O(2^{n/2})$.

Putting it all together, the size of the enumeration tree of Algorithm 3 is bounded by $O(2^{n/2})$ (or $O(1.414^n)$). It is easy to show that Algorithm 3 takes at most $O(m)$ time in each recursion, thus the total time complexity of Algorithm 3 is bounded by $O(m \times 2^{n/2})$. □

By Theorem 3.4, it is easy to verify that the space complexity of Algorithm 3 is bounded by $O(\triangle_{clq} n + m)$, where $\triangle_{clq}$ is the maximum size of the biclique in G. Note that since there may exist $O(2^{n/2})$ maximal bicliques in a bipartite graph as shown in [37], the worst-case time complexity of any algorithm to enumerate all maximal bicliques must be no less than $O(\tau \times 2^{n/2})$ where τ is the average size of all maximal bicliques. As a result, the worst-case time complexity of Algorithm 3 is near optimal (only up to a factor m/τ).

A polynomial-delay implementation. We find that Algorithm 3 can also achieve polynomial-delay time complexity with a slight modification. The key idea is that: when a vertex $u \in C_U$ is added into the current partial biclique (R_U, R_V), we then check if the conditions $N(R_U \cup \{u\}) = R_V \cup C'_V$ and $N(R_V \cup C'_V) = R_U \cup \{u\}$ are satisfied; if so, the current biclique $(R_U \cup \{u\}, R_V \cup C'_V)$ is definitely maximal and can be output as a result. To avoid redundant results, some additional modifications are required: 1) remove lines 4-5 of Algorithm 3; 2) check $C_V \neq \emptyset$ before line 17; and 3) remove all vertices in C'_V that have no neighbor in $C_U \setminus \{u\}$. The detailed procedure is shown in Algorithm 4.

Algorithm 4: A polynomial-delay algorithm of pivot-based maximal biclique enumeration

Input: The bipartite graph G
Output: All maximal bicliques of G
1 $BicliqueEnum(\emptyset, \emptyset, U, V, \emptyset, \emptyset)$;
2 **Function:** $BicliqueEnum(R_U, R_V, C_U, C_V, X_U, X_V)$
3 **if** $C_U \cup C_V = \emptyset$ **then return** ;
4 Lines 7-15 of Algorithm 3 ;
5 **Function:** $BicliqueBranch(R_U, R_V, u, C_U, C_V, X_U, X_V)$
6 **if** $C_V \neq \emptyset$ **then**
7 $C'_V \leftarrow C_V \cap N_u(G)$; $X'_V \leftarrow X_V \cap N_u(G)$;
8 **if** $N(R_U \cup \{u\}) = R_V \cup C'_V$ *and* $N(R_V \cup C'_V) = R_U \cup \{u\}$ **then** Output $(R_U \cup \{u\}, R_V \cup C'_V)$ as a result ;
9 $C'_V = \{w \in C'_V | N_w(G) \cap C_U \setminus \{u\} \neq \emptyset\}$;
10 $BicliqueEnum(R_U \cup \{u\}, R_V, C_U, C'_V, X_U, X'_V)$;

Theorem 4.3. *Algorithm 4 is a polynomial-delay algorithm with $O(nm\triangle_{clq})$ delay.*

Proof. First, we prove that the first maximal biclique can be output in polynomial time. Denote by u (in C_U) the vertex used to expand the initial biclique $(\emptyset, \emptyset)$. In the recursion with current biclique $(\{u\}, \emptyset)$, we can see that $(N(C_V), C_V)$ is a maximal biclique, where $C_V = N_u(G)$. Clearly, any vertex $u' \in N(C_V) \setminus \{u\}$ can be the pivot vertex, and only the pivot vertex is used to expand $(\{u\}, \emptyset)$, because $\overline{N}_{u'}(C_V) = \emptyset$. When all vertices in $N(C_V)$ have been expanded into $(\{u\}, \emptyset)$, the first maximal biclique $(N(C_V), C_V)$ is obtained, which consumes at most $O(m\triangle_{clq})$ time. Second, we show that when a maximal biclique is output, the next maximal biclique can be determined in polynomial time ($O(nm\triangle_{clq})$). Given a recursion with the partial result (R_U, R_V) satisfying that $(R_U, R_V \cup C_V)$ is a maximal biclique, we can derive that the pivot vertex $u \in C_U$ and vertices in $\overline{N}_u(C_V)$ are used to expand (R_U, R_V) by Theorem 4.1. When a pivot vertex u is added to (R_U, R_V), $(R_U \cup N(N_u(C_V)), R_V \cup N_u(C_V))$ is clearly a maximal biclique. Based on the previous analysis, such a maximal clique can be found in $O(m\triangle_{clq})$ time. Similarly, we note that another maximal biclique $(R_U \cup N_v(C_U), R_V \cup N(N_v(C_U)))$ can be obtained if the first vertex $v \in \overline{N}_u(C_V)$ is added to (R_U, R_V). Since $v \notin N_u(G)$ for any $v \in \overline{N}_u(C_V)$, any maximal biclique generated by the recursion with $(R_U \cup \{u\}, R_V)$ is different from that generated by the recursion with $(R_U, R_V \cup \{v\})$. All that remains is to show that the new maximal biclique can be found in polynomial time when using the vertices in $\overline{N}_u(C_V) \setminus \{v\}$ to expand (R_U, R_V). Denote by $v' \in \overline{N}_u(C_V) \setminus \{v\}$ the next vertex used to expand (R_U, R_V). If $N_{v'}(C_U) \subseteq N_v(C_U)$, the recursion with $(R_U, R_V \cup \{v'\})$ can be terminated because the vertex $v \in X_V$ is the pivot vertex. Otherwise, $(R_U \cup N_{v'}(C_U), R_V \cup N(N_{v'}(C_U)))$ is output as a maximal biclique. In the worst case, all vertices in $\overline{N}_u(C_V) \setminus \{v\}$ may satisfy $N_{v'}(C_U) \subseteq N_v(C_U)$. Thus, the time to find a new maximal biclique is bounded by $O(nm\triangle_{clq})$, since $|\overline{N}_u(C_V)| < n$ and the maximum depth of any recursion is bounded by $O(\triangle_{clq})$. This completes the proof. □

Discussions. Note that pivot-based algorithms are widely used in enumerating maximal cliques in traditional graphs [15, 33, 41]. The key idea of these pivot-based algorithms is that they skip all neighbor vertices of the pivot vertex to expand the current clique based on the fact that any maximal clique either contains a vertex v or a non-neighbor of v. However, such a fact no longer holds for maximal biclique enumeration. This is because in maximal biclique enumeration, there exist two different candidate sets C_U and C_V. We need to consider skipping both the neighbor vertices and non-neighbor vertices to expand the current biclique. Inspired by existing pivot-based algorithms for maximal clique enumeration [41], we develop a new and general pivoting technique for enumerating hereditary subgraphs in bipartite graphs, based on which a novel pivoting technique for maximal biclique enumeration is proposed. Our pivot-based maximal biclique enumeration technique can skip all vertices in $C_U \setminus \{u\}$ and $N_u(C_V)$, where $u \in C_U$ is a pivot vertex. In addition,

the time complexity of our algorithm depends on $O(2^{n/2})$ which is also different from the traditional pivot-based maximal clique enumeration algorithms that depend on $O(3^{n/3})$.

4.3 Optimization Techniques

In this subsection, we propose several optimization techniques to further improve the efficiency. The first one is an early termination technique to reduce the maximum depth of recursions and the other one is an ordering technique to reduce the maximum size of the initial candidate sets.

Early termination. The early termination trick is based on the fact that if there is no edge in the subgraph $G(C_U, C_V)$, then there are at most two possible maximal bicliques $(R_U \cup C_U, R_V)$ and $(R_U, R_V \cup C_V)$ to be explored. This is because both $(R_V \cup C_V) \subseteq N(R_U)$ and $(R_U \cup C_U) \subseteq N(R_V)$ always hold; and if a vertex on one side of the candidate sets is used to expand (R_U, R_V), then the other side of the candidate sets will be empty.

We can slightly modify Algorithm 3 to implement such an early termination trick. Specifically, let $u \in C_U$ be the optimal pivot vertex selected by Algorithm 3 (lines 7-9). If $P_V = \emptyset$ which means that there is no edge in $G(C_U, C_V)$, the current recursive call can be terminated. In this case, we only need to determine whether $(R_U \cup C_U, R_V)$ is a maximal biclique by checking $X_U = \emptyset$ and $X_V \cap N(C_U) = \emptyset$; and determine whether $(R_U, R_V \cup C_V)$ is a maximal biclique by verifying $X_V = \emptyset$ and $X_U \cap N(C_V) = \emptyset$.

Ordering optimization. As indicated in [1, 9], the performance of maximal biclique enumeration algorithms is often sensitive to the vertex ordering. Inspired by this, we can also use the ordering technique to further improve the performance of our algorithm.

Given an ordered vertex set $O = \{u_1, u_2, ..., u_n\}$ of all vertices in $U \cup V$, we define $O^{\geq u_i}$ as the set of vertices in O with ranks higher than u_i; and $O^{<u_i}$ is defined similarly. Let $N^2_{u_i}(G)$ be the set of vertices in G whose distance from $u_i \in O$ is 2, i.e., $N^2_{u_i}(G) = \{u_j \in O | j \neq i, N_{u_i}(G) \cap N_{u_j}(G) \neq \emptyset\}$. Note that in bipartite graphs, $N^2_{u_i}(G)$ does not contain any neighbor of u_i. Given a vertex u_i in O, we denote by $G^+_{u_i}$ the subgraph of G induced by $O^{\geq u_i} \cap N^2_{u_i}(G)$ and $N_{u_i}(G)$. It is easy to show that for any maximal biclique (A, B) of G, there must exist a subgraph $G^+_{u_i}$ that contains all vertices in $(A \setminus \{u_i\}, B)$. Here the vertex u_i can be only contained in $O \cap U$, as the diameter of (A, B) is 2 and $G^+_{u_i}$ contains all neighbor of u_i and all vertices in $N^2_{u_i}(G)$ with ranks higher than u_i. Thus, to enumerate all maximal bicliques in G, it is sufficient to enumerate all maximal bicliques in each $G^+_{u_i}$ ($u_i \in O \cap U$). In this work, we make use of two ordering techniques, including the classic degeneracy ordering on graphs [15, 25] and the 2-hop degree ordering developed in [9]. Note that to obtain the degeneracy ordering, we can treat the bipartite graph as a traditional graph, and then iteratively remove the smallest-degree vertex from the bipartite graph to generate the degeneracy ordering. Such an iteratively-peeling procedure can be implemented in $O(m + n)$ time [2, 15].

4.4 Enumerating Large Maximal Bicliques

In practical applications, we may be more interested in finding large maximal bicliques with size no less than a given threshold, since small maximal bicliques typically have low practical value [1, 28]. More specifically, we aim to identify every maximal biclique (A, B) that satisfies $|A| \geq q$ and $|B| \geq q$, where q is a given threshold.

Note that our algorithm can be easily adapted to enumerate such size-constraint maximal bicliques. Moreover, we can also use the size-constraint to further prune unnecessary computations. Below, we first introduce the definition of (α, β)-core, which was originally proposed in [7, 27].

Definition 6 ([7]). Given a bipartite graph G, an (α, β)-core is a maximal subgraph $H = (U_H, V_H, E_H)$ of G such that each vertex u in U_H has a degree no less than α in H and each vertex v in V_H has a degree no less than β in H.

By Definition 6, it is easy to show that every maximal biclique (A, B) that satisfies $|A| \geq q$ and $|B| \geq q$ must be contained in the (q, q)-core of G. Thus, we can first compute the (q, q)-core of the bipartite graph, and then enumerate all size-constraint maximal bicliques in the (q, q)-core, instead of in the original bipartite graph. Such a (q, q)-core can significantly prune a large number of unpromising vertices. In addition, in our pivot-based algorithm, we need to compute the optimal pivot vertex (line 7 of Algorithm 3) which means that the degree of every vertex in $C_U \cup C_V$ must be computed. Based on this degree information, we can prune the vertices in C_U (C_V) if their degrees are less than $q - |R_V|$ ($q - |R_U|$) in $G(C_U, C_V)$ to further improve the efficiency of our algorithms.

5 MAXIMAL k-BIPLEX ENUMERATION

In this section, we focus on the problem of enumerating all maximal k-biplexes of a given bipartite graph G, where a maximal k-biplex is another instance of maximal $\mathcal{P}$-subgraph. Such a maximal k-biplex enumeration problem has also been extensively studied in recent years [40, 48, 49]. To the best of our knowledge, there are two state-of-the-art solutions for solving this problem. The first one is developed in [49], which proposes a prefix tree based recursive algorithm to detect whether each possible combination of vertices of G can form a maximal k-biplex. Such an enumeration method may explore all the possible combinations of vertices, thus it cannot handle large bipartite graphs. The second one is a polynomial-delay algorithm proposed in [48], which is based on the reverse search framework [10]. Such an algorithm, however, needs to store all detected maximal k-biplexes in the main memory to guide the reverse search procedure, which prohibits it to handle large bipartite graphs. To overcome the limitations of existing solutions, we propose a novel algorithm to enumerate all maximal k-biplexes based on our general pivot-based enumeration framework.

5.1 Pivot-based Maximal k-Biplex Enumeration

To use our general pivot-based enumeration framework for maximal k-biplex enumeration, the key is to derive the skipping sets (P_U, P_V) based on the pivot vertex as stated in Theorem 3.1. Suppose that the pivot vertex u is selected from $C_U \cup X_U$. Given a vertex $v \in U$, we denote by $\overline{N}_v(B)$ the set of non-neighbors of v in B and define $\overline{d}_v(B)$ as $|\overline{N}_v(B)|$. Below, we detail our solutions.

Constructing the skipping set P_V. We find that the skipping set $P_V \subseteq C_V$ can be easily derived, as shown in Lemma 4.

Lemma 4. Let $u \in C_U \cup X_U$ be a pivot vertex in a recursion to enumerate all maximal k-biplexes that contain (R_U, R_V). Then, the skipping set P_V can be set as $C_V \cap N_u(G)$.

Constructing the skipping set P_U. We note that constructing the skipping set P_U for maximal k-biplex enumeration is quite nontrivial. Similar ideas to construct P_U for maximal biclique enumeration cannot be used for maximal k-biplex enumeration. Specifically, for maximal biclique enumeration, only the pivot vertex u (if $u \in C_U$) and a subset of vertices on the opposite side (i.e., C_V) are considered to be used to expand the partial biclique. Such a nice property, however, no longer holds for maximal k-biplex enumeration. For instance, consider a bipartite graph G shown in Fig. 1. Let $(R_U, R_V) = (\{u_1, u_4\}, \{v_1\})$ and $(C_U, C_V) = (\{u_2, u_3\}, \{v_2, v_3, v_4, v_5\})$ be the current k-biplex and the candidate sets used to expand (R_U, R_V), respectively. Suppose that $k = 1$ and $v_3 \in C_V$ is selected as the pivot vertex. Then, it is easy to see that $(\{u_1, u_4\}, \{v_1, v_2, v_5\})$ is a maximal 1-biplex of G. Thus, when selecting a pivot vertex v from $C_V \cup X_V$, there still exist some maximal k-biplexes containing (R_U, R_V) in the subgraph of G induced by R_U and $R_V \cup C_V \setminus \{v\}$. This example suggests that P_U cannot be simply set to $C_U \setminus \{u\}$ if $u \in C_U \cup X_U$ is selected as the pivot.

For any maximal k-biplex (A, B) containing (R_U, R_V) in the subgraph of G that is induced by $R_U \cup C_U \setminus \{u\}$ and R_V, we notice that there exists an approach to determine whether it is also a maximal k-biplex in G. Specifically, if (A, B) is also maximal in G, there must exist a *conflict*

Algorithm 5: The pivot-based maximal k-biplex algorithm

Input: The bipartite graph G and a parameter k
Output: All maximal k-biplexes of G

$BiplexEnum(\emptyset, \emptyset, U, V, \emptyset, \emptyset)$;
Function: $BiplexEnum(R_U, R_V, C_U, C_V, X_U, X_V)$
  if $C_U \cup C_V = \emptyset$ **then**
    if $X_V \cup X_U = \emptyset$ **then** Output (R_U, R_V) as a result;
    return;
  Select a pivot $u \in C_U \cup X_U$ and obtain the skipping sets $(P_U = \{w \in C_U | \overline{N}_u(R_V) \subseteq N_w(G)\}, P_V = C_V \cap N_u(G))$;
  Select a pivot $v \in C_V \cup X_V$ and obtain the skipping sets $(P'_U = C_U \cap N_v(G), P'_V = \{w \in C_V | \overline{N}_v(R_U) \subseteq N_w(G)\})$;
  if $|P'_U| + |P'_V| > |P_U| + |P_V|$ **then** $P_U \leftarrow P'_U$; $P_V \leftarrow P'_V$;
  foreach $w \in C_U \setminus P_U$ **do**
    $BiplexBranch(R_U, R_V, w, C_U \setminus \{w\}, C_V, X_U, X_V)$;
    $C_U \leftarrow C_U \setminus \{w\}$; $X_U \leftarrow X_U \cup \{w\}$;
  foreach $w \in C_V \setminus P_V$ **do**
    $BiplexBranch(R_V, R_U, w, C_V \setminus \{w\}, C_U, X_V, X_U)$;
    $C_V \leftarrow C_V \setminus \{w\}$; $X_V \leftarrow X_V \cup \{w\}$;
Function: $BiplexBranch(R_U, R_V, u, C_U, C_V, X_U, X_V)$
  $(C'_U, C'_V) \leftarrow BiPlexUpdates(R_U, R_V, u, C_U, C_V)$;
  $(X'_U, X'_V) \leftarrow BiPlexUpdates(R_U, R_V, u, X_U, X_V)$;
  $BiplexEnum(R_U \cup \{u\}, R_V, C'_U, C'_V, X'_U, X'_V)$;

Algorithm 6: $BiPlexUpdates(R_U, R_V, u, C_U, C_V)$

$C'_U \leftarrow C_U \setminus \{u\}$; $C'_V \leftarrow C_V \cap N_u(G)$;
foreach $v \in C_V \setminus N_u(G)$ **do**
  if $\overline{d}_v(R_U \cup \{u\}) \leq k \wedge \overline{d}_u(R_V \cup \{v\}) \leq k$ **then** $C'_V \leftarrow C'_V \cup \{v\}$;
foreach $v \in R_V \setminus N_u(G)$ **do**
  if $\overline{d}_v(R_U \cup \{u\}) = k$ **then** $C'_U \leftarrow C'_U \cap N_v(G)$;
return (C'_U, C'_V);

vertex in $\overline{N}_u(R_V)$ that prohibits the pivot vertex u to enlarge (A, B). Based on this analysis, we can derive the following lemma, which is used to compute maximal k-biplexes of G in the subgraph $G(R_U \cup C_U \setminus \{u\}, R_V \cup P_V)$.

Lemma 5. Given a pivot vertex $u \in C_U \cup X_U$ and a skipping set $P_V \subseteq C_V$, if a k-biplex (A, B) that contains (R_U, R_V) but not u and $C_V \setminus P_V$ is maximal in G (the maximal k-biplex of G belongs to the case (iii) in Theorem 3.2), there must exist a vertex $v' \in R_V \setminus N_u(G)$ with $\overline{d}_{v'}(A) = k$. Otherwise, (A, B) is definitely not maximal in G.

By Lemma 5, the problem of generating the skipping set P_U is equivalent to find a subset $P_U \subseteq C_U$ such that for any k-biplex (A, B) containing (R_U, R_V) in $G(R_U \cup P_U, R_V \cup P_V)$, $\overline{d}_{v'}(A) < k$ holds for each $v' \in R_V \setminus N_u(G)$. Then, the following lemma shows how to derive the skipping set P_U.

Lemma 6. Given a pivot vertex $u \in C_U \cup X_U$ and the skipping set $P_V \subseteq C_V$ in a recursion to enumerate all maximal k-biplexes that contain (R_U, R_V), then the skipping set P_U can be set as $\{w \in C_U | \overline{N}_u(R_V) \subseteq N_w(G)\}$.

Armed with Lemma 4 and Lemma 6, we can obtain the following pivoting rule for maximal k-biplex enumeration.

Theorem 5.1 (Biplex Pivoting Rule). *Let $u \in C_U \cup X_U$ be a pivot vertex in a recursion. The vertices in (P_U, P_V) can be skipped to expand (R_U, R_V), where $P_U = \{w \in C_U | \overline{N}_u(R_V) \subseteq N_w(G), w \neq u\}$ and $P_V = C_V \cap N_u(G)$.*

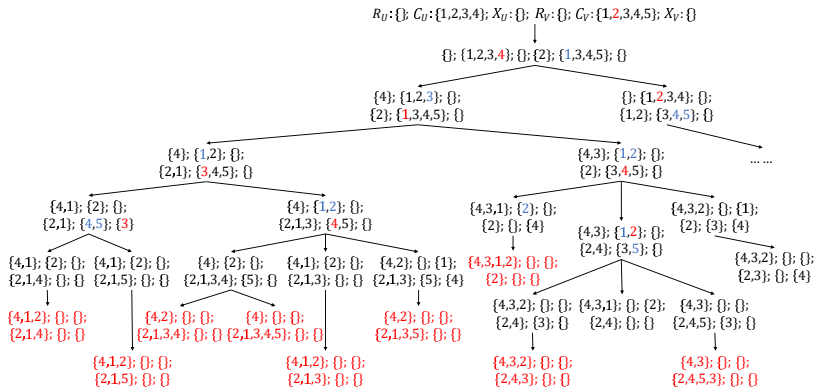


Fig. 3. The enumeration tree of pivot-based maximal biplex enumeration algorithm ($k = 1$).

Implementation details. With Theorem 5.1, we can implement our pivot-based maximal k-biplex enumeration algorithm which is outlined in Algorithm 5. In Algorithm 5, the sets (R_U, R_V), (C_U, C_V), and (X_U, X_V) are the current k-biplex of G, the candidate sets, and the exclusion sets, respectively. The algorithm first makes use of Theorem 5.1 to generate the optimal skipping sets (P_U, P_V) (lines 6-8). Then, each vertex contained in (C_U, C_V) but excluded in (P_U, P_V) is used to expand (R_U, R_V) (lines 9-14) to continue sub-recursive calls. If both the candidate sets and the exclusion sets are empty, the current k-biplex (R_U, R_V) is maximal in G (lines 3-4).

Note that when a vertex is used to expand the current k-biplex, Algorithm 5 invokes Algorithm 6 to update the candidate and exclusion sets (lines 16-17 of Algorithm 5). Specifically, when a vertex $u \in C_U$ is pushed into (R_U, R_V), every vertex in C_V should keep the invariance that it has at most k non-neighbors in $R_U \cup \{u\}$ (lines 1-3 of Algorithm 6). If there is a subset S in $R_V \setminus N_u(G)$ that has k non-neighbors in $R_U \cup \{u\}$, then every vertex in $C_U \setminus \{u\}$ must be the common neighbors of S, i.e., $C_U \setminus \{u\} \subseteq N(S)$ (lines 4-5 of Algorithm 6). The solution for updating the exclusion sets (X_U, X_V) is similar, thus we omit the details. Below, we give an example to illustrate the idea of Algorithm 5.

Example 2. Reconsider the bipartite graph G in Fig. 1 and assume that $k = 1$. Algorithm 5 first initializes the sets R_U, R_V, X_U, and X_V to empty, and the sets C_U and C_V to U and V, respectively. Based on the biplex pivoting rule (Theorem 5.1), the vertex $v_2 \in C_V$ is selected as the pivot vertex in the first recursion, and it can be obtained that only vertex v_2 is used to expand the current k-biplex $(\emptyset, \emptyset)$. Then, in the recursion with the current k-biplex $(\emptyset, \{v_2\})$, a new pivot vertex $u_4 \in C_U$ can be selected, and the vertices $(\{u_4\}, \{v_1\})$ can be obtained to expand $(\emptyset, \{v_2\})$. If u_4 is pushed into $(\emptyset, \{v_2\})$, the similar pivoting rule will be used to enumerate all maximal k-biplex containing $(\{u_4\}, \{v_2\})$. After that, the vertex v_1 is pushed into $(\emptyset, \{v_2\})$ to continue the recursions. Finally, the algorithm terminates if no vertex is left to expand the current k-biplex in each recursion. The complete enumeration tree of Algorithm 5 is illustrated in Fig. 3.

The following theorem shows that such an updating algorithm takes $O(kn)$ time.

Theorem 5.2. *The time complexity of Algorithm 6 is $O(kn)$.*

By Theorem 3.4, we can derive that the space complexity of Algorithm 5 is bounded by $O(\triangle_{plex} n + m)$, where $\triangle_{plex}$ is the maximum size of the biclique in G. By Theorem 5.1 and Theorem 3.3, we can also conclude that Algorithm 5 correctly enumerates all maximal k-biplex of G.

The worst-case time complexity of Algorithm 5 is $O(n^2 2^n)$, as there are at most 2^n enumeration branches. Nevertheless, it works well for real-world bipartite graphs (as evidenced in our experiments) due to the proposed powerful pivot-based pruning technique. Moreover, similar to maximal biclique enumeration, we can also use the ordering techniques, such as the degeneracy ordering [15, 25] and the 2-hop degree ordering [9]), to further improve the efficiency of Algorithm 5.

Algorithm 7: Size-constraint maximal k-biplex algorithm

Input: The bipartite graph G and two parameters k and q
Output: All size constraint maximal k-biplexes of G

```
H ← reduce the graph G by the (q − k, q − k)-core;
O ← ordering all vertices in U_H ∪ V_H;
foreach u_i ∈ O, s.t. u_i ∈ U do
    C_U ← O^{>u_i} ∩ N^2_{u_i}(H); C_V ← ∪_{u∈C_U∪{u_i}} N_u(H); X_U ← O^{<u_i} ∩ N^2_{u_i}(H); X_V ← ∅;
    Reduce (C_U, C_V) and (X_U, X_V) by Lemma 9;
    BiplexEnum({u_i}, ∅, C_U, C_V, X_U, X_V);
```

Remark. We note that it seems difficult to derive a tighter time complexity for Algorithm 5. This is because the size of the skipping set P_U is very hard to bound; and the size of candidate sets cannot be reduced if u is used to expand the current partial k-biplex, because all non-neighbors of u can still be included in the candidate sets. We leave the problem of deriving a tighter time complexity for Algorithm 5 as an interesting open question.

Discussion. We note that an existing solution used to find the maximum k-biplex [47] can also be adapted to enumerate all maximal k-biplexes based on the so-called symmetric-BK branching technique [47]. Specifically, this technique first selects a pivot vertex u from $R_U \cup C_U$ (or from $R_V \cup C_V$). If $\overline{d}_u(R_U \cup C_U) > k$, then it generates at most $k+2$ subbranches to enumerate all maximal k-biplexes. Here each subbranch B_i is $\langle R_i = R_U \cup R_V \cup \{u_1, ..., u_{i-1}\}, C_i = C_U \cup C_V \setminus \{u_1, ..., u_i\}, X_i = X_U \cup X_V \setminus \{u_i\}\rangle$, where $u_1 = u$ and $u_i \in \overline{N}_u(C_V)$ $(i > 1)$. With the restriction on the number of subbranches for each recursive call, we can use a similar proof given in [47] to show that the recurrence inequation $T(n) \leq \sum_{i=1}^{k+2} T(n-i)$ holds for such an adapted algorithm, where $T(n)$ denotes the number of recursive calls of the algorithm. Then, based on the theoretical result in [16], we can derive that the worst-case time complexity of this adapted algorithm depends on $O(\gamma_k^n)$, where $\gamma_k = 1.839$, 1.928, and 1.966, for $k = 1$, 2, and 3 respectively. Note that the time complexity of this adaption is worse than that of the original algorithm proposed in [47] for detecting the maximum k-biplex. The reason is that the number of maximal k-biplexes on a bipartite graph can be extremely large and the problem of finding one maximum k-biplex is often much easier than the problem of enumerating all maximal k-biplexes. Moreover, we find that such an adapted algorithm is not very efficient for enumerating all maximal k-biplexes, as it involves many unnecessary computations. Specifically, in this algorithm, B_1 can generate all maximal k-biplexes that have been detected by each B_i $(i > 1)$ based on the result that $R_1 \subset R_i$ and $C_i \subset C_1$ for each $i > 1$, thus many redundant results will be explored by this algorithm. Indeed, as shown in our experiments (see Sec. 6.3), the performance of such an adapted algorithm is significantly worse than our pivot-based solutions, although its worst-case time complexity is lower than ours.

5.2 Enumerating Large Maximal k-Biplexes

In this subsection section, we focus on the problem of enumerating large maximal k-biplexes as small maximal k-biplexes are often no practical use in real-world applications. More importantly, we prove that large maximal k-biplexes have small diameters, thus they are often more cohesive compared to the small maximal k-biplexes.

Lemma 7. For any maximal k-biplex (A, B) of G, the diameter of $G(A, B)$ is no larger than 3 if $|A| \geq 2k+1$ and $|B| \geq 2k+1$.

According to Lemma 7, we can see that every maximal k-biplex (A, B) of G with $|A| \geq 2k+1$ and $|B| \geq 2k+1$, $G(A, B)$ must be densely connected since its diameter is no larger than 3. Note that the constraints of $|A| \geq 2k+1$ and $|B| \geq 2k+1$ are relatively mild, since k is often not very large (e.g.,

$k \leq 5$). In the following, we focus mainly on the problem of enumerating all maximal k-biplexes of G whose size on each side is no less than $q \geq 2k+1$, i.e., output each maximal k-biplex (A, B) of G with $|A| \geq q$, $|B| \geq q$, and $q \geq 2k+1$.

Pruning techniques. Similar to the size-constraint maximal biclique enumeration problem, we can also make use of the (α, β)-core to prune unpromising vertices for size-constraint maximal k-biplex enumeration.

For each k-biplex (A, B) of G with $|A| \geq q$ and $|B| \geq q$, it is easy to see that the smallest degree of vertices in $G(A, B)$ is at least $q-k$, which means that every k-biplex must be contained in the $(q-k, q-k)$-core.

Lemma 8. Any maximal k-biplex (A, B) of bipartite graph G with $|A| \geq q$ and $|B| \geq q$ must be included in the $(q-k, q-k)$-core of G.

By Lemma 8, all vertices that are not contained in the $(q-k, q-k)$-core can be safely pruned from G when enumerating maximal k-biplexes (A, B) with $|A| \geq q$ and $|B| \geq q$. In addition, we also derive a more effective pruning rule to further reduce the unpromising vertices.

Lemma 9. Given any k-biplex (A, B) of G with $|A| \geq q$ and $|B| \geq q$, for each pair of vertices u_1 and u_2 in A and each pair of vertices v_1 and v_2 in B, we have $|N_{u_1}(B) \cap N_{u_2}(B)| \geq q-2k$ and $|N_{v_1}(A) \cap N_{v_2}(A)| \geq q-2k$.

When enumerating the size-constraint maximal k-biplexes that contain a specific vertex $u \in U$, we can further remove the vertices in the candidate sets and exclusion sets that conflict with u according to the condition shown in Lemma 9.

Size-constraint maximal k-biplex enumeration. Algorithm 7 outlines the size-constraint maximal k-biplex enumeration algorithm which is integrated with the pruning rules developed in Lemma 8 and Lemma 9. Specifically, Algorithm 7 first applies the (α, β)-core reduction technique (Lemma 8) to remove the unnecessary vertices in G (line 1). Then, the algorithm uses the ordering technique (degeneracy ordering or 2-hop degree ordering) to enumerate all maximal k-biplexes in the remaining graph (lines 2-6). With a specific ordered vertex set O, each vertex $u_i \in O$ with $u_i \in U$ is selected to enumerate all size-constraint maximal k-biplexes containing u_i. Note that the corresponding candidate sets (C_U, C_V) and exclusion sets (X_U, X_V) are initialized with vertices whose distance from u_i is no larger than 3 by Lemma 7 (line 4). After that, the algorithm leverages Lemma 9 to further reduce the size of (C_U, C_V) and (X_U, X_V) (line 5). Finally, the pivot-based recursive procedure is invoked to enumerate all the maximal k-biplexes (line 6).

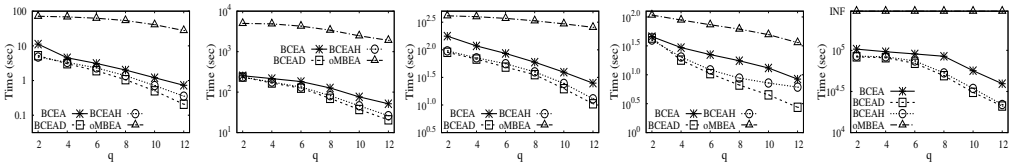
6 EXPERIMENTS

6.1 Experimental Setup

Different algorithms. We implement four algorithms to enumerate maximal bicliques: BCEA, BCEAD, BCEAH, and BCDelay. Here BCEA is Algorithm 3; BCEAD and BCEAH denote Algorithm 3 with the degeneracy ordering and 2-hop degree ordering optimization respectively; and BCDelay is Algorithm 4 with a polynomial-delay property. All BCEA, BCEAD, BCEAH, and BCDelay are equipped with the early termination trick. To enumerate size-constraint maximal bicliques, all pruning techniques developed in Section 4.4 are used for BCEA, BCEAD, and BCEAH. Note that BCDelay cannot guarantee polynomial-delay for enumerating size-constraint maximal bicliques, since it is difficult to determine whether there is a biclique with size no less than q in a bipartite graph. Thus, we exclude BCDelay when enumerating size-constraint maximal bicliques. We compare our algorithms with the state-of-the-art maximal biclique enumeration algorithm developed in [9], namely oMBEA. Note that there are several other maximal biclique enumeration algorithms

Table 1. Real-world graph datasets.

Datasets	$\|U\|$	$\|V\|$	$\|E\|$	$d_{1\max}$	$d_{2\max}$
Crime	829	551	1,476	25	18
Ucforum	899	522	7,089	99	126
Fjwiki	612	1,922	12,382	360	48
Escorts	16,730	6,624	50,632	125	305
YouTube	94,238	30,087	293,360	1,035	7,591
BkCrossing	105,278	340,523	1,149,739	13,601	2,502
Cite	22,715	731,769	2,411,819	189,292	1,264
Dbtropes	64,415	87,678	3,232,134	6507	12400
IMDB	303,617	896,302	3,782,463	1,334	1,590
Twitter	175,214	530,418	4,664,605	968	19,805
DBLP	1,953,085	5,624,219	12,282,059	1,386	287



(a) YouTube (b) BkCrossing (c) Cite (d) IMDB (e) Dbtropes

Fig. 4. Runtime of various algorithms for enumerating large maximal bicliques on large bipartite graphs.

[1, 14, 50], but all of them are much worse than oMBEA [9]. Therefore, we only use oMBEA as the baseline in our experiments.

We also implement four maximal k-biplex enumeration algorithms, called BPEA, BPEAD, BPEAH, and symBK, where BPEA is Algorithm 5, BCEAD and BCEAH denote Algorithm 5 with the degeneracy ordering and 2-hop degree ordering optimization respectively, symBK is an adaptation of the algorithm proposed in [47] which is originally devised to detect the maximum k-biplex. To enumerate size-constraint maximal k-biplexes, all pruning techniques proposed in Section 5.2 are used for BPEA, BPEAD, BPEAH, and symBK. We compare our algorithms with two state-of-the-art maximal k-biplex algorithms, called iTrav [48] and iMB [49]. All the algorithms, including our algorithms and baselines, are implemented in C++, and tested on a PC with one 2.2 GHz CPU and 64GB memory running CentOS.

Datasets. In our experiments, we use 11 real-world bipartite graphs for performance evaluation. The detailed statistics of datasets are summarized in Table 1, where the columns $d_{1\max}$ and $d_{2\max}$ are the maximum degree of vertices in U and V, respectively. Note that the first 4 small real-world bipartite graphs in Table 1 are used to evaluate the performance of different algorithms for enumerating all maximal k-biplexes. Since the number of small-size maximal k-biplexes is often very large, most of algorithms cannot handle large graphs when enumerating all maximal k-biplexes. The remaining 7 large real-world bipartite graphs in Table 1 are used to test the performance of algorithms for enumerating all (and size-constraint) maximal bicliques and the size-constraint maximal k-biplexes. All datasets are downloaded from (http://www.konect.cc/networks).

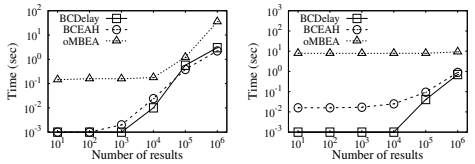
Parameters. When enumerating size-constraint maximal bicliques, the integer threshold parameter q is selected from 2 to 12. In enumerating maximal k-biplexes, there are two parameters k and q (the size-constraint threshold). We choose k from 1 to 4, with a default value of 1; and we select q from 10 to 20 for all datasets except DBLP (for which q is selected from 6 to 10, because DBLP does not contain very large k-biplexes).

6.2 Efficiency of Maximal Biclique Enumeration

Exp-1: Results of enumerating all maximal cliques. Table 2 shows the runtime of oMBEA, BCEA, BCEAD, BCEAH, and BCDelay for enumerating all maximal bicliques on different large real-world bipartite graphs. From Table 2, we can see that the proposed pivot-based algorithm

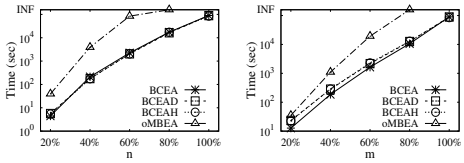
Table 2. Runtime of various algorithms for enumerating all maximal bicliques on large graphs (in seconds).

Datasets	oMBEA	BCEA	BCEAD	BCEAH	BCDelay
YouTube	74.59	17.44	7.77	**5.89**	8.27
BkCrossing	4739.6	314.63	**307.29**	310.85	337.52
Cite	414.83	265.67	95.21	**91.53**	113.65
Dbtropes	>20d	116710.13	88851.47	85111.11	**80617.47**
IMDB	111.04	**51.15**	58.59	54.81	59.57
Twitter	135.73	424.21	240.87	**107.89**	193.72
DBLP	26.73	15.83	13.52	14.7	**11.74**



(a) YouTube (b) DBLP

Fig. 5. Delay testing for maximal biclique enumeration algorithms.



(a) varying vertices (b) varying edges

Fig. 6. Scalability for maximal biclique enumeration algorithms.

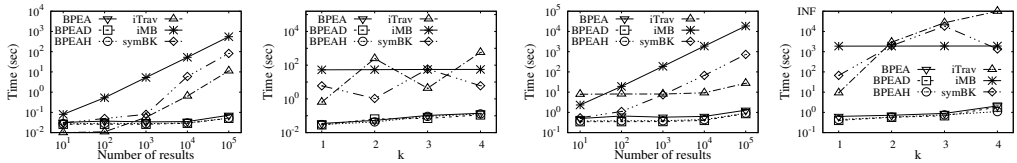
BCEAH consistently outperforms the state-of-the-art algorithm oMBEA. On most datasets, our algorithms can achieve one order of magnitude faster than oMBEA. For instance, our algorithms take at most 337 seconds on BkCrossing, while oMBEA consumes 4739.6 seconds. The reasons are two-fold. First, the proposed pivoting technique can dramatically prune unnecessary search branches, because it only expands the current partial biclique with the pivot vertex and the non-neighbor vertices on the opposite side. Second, the proposed pivoting technique is computationally efficient (it simply selects the vertex with the largest degree as the pivot), which can be computed in linear time. In contrast, the dominating technique used in oMBEA needs to compute the common neighbors for each pair of vertices in the candidate set, which is much more expensive. As a result, our pivot-based solutions can substantially outperform oMBEA. In addition, when comparing our algorithms BCEA, BCEAD, BCEAH, and BCDelay, we notice that there is a relatively-small gap in time spent on most datasets, which indicates that the proposed pivoting techniques play the crucial role in reducing unnecessary computations.

Exp-2: Results of enumerating large maximal bicliques. Note that oMBEA was originally designed to enumerate all maximal bicliques. For a fair comparison, all the pruning techniques developed in Section 4.4 for enumerating size-constraint maximal bicliques are also integrated in oMBEA. Fig. 4 shows the runtime of different algorithms on all large bipartite graphs with varying q, where "INF" denotes that the algorithm cannot terminate within 24 hours. From Fig. 4, we observe that the performance of BCEAD consistently outperforms that of oMBEA with varying q. Moreover, with the increase of the parameter q, the speedup rate of BCEAD compared to oMBEA increases dramatically. The reason behind this may be that as q increases, the pruning technique (i.e., (α, β)-core) can produce a denser induced subgraph. This would decrease the effectiveness of the neighborhood domination based pruning techniques used in oMBEA. In addition, we can see that BCEAD is slightly better than BCEAH; and BCEA still significantly outperforms oMBEA. The results further demonstrate the high efficiency of the proposed pivoting techniques compared to the state-of-the-art algorithm.

Exp-3: Delay testing for enumerating maximal bicliques. Fig. 5 shows the time delay of algorithms BCDelay, BCEAH, and oMBEA on two representative datasets with varying the number of returned maximal bicliques. The results for the other algorithms (BCEA and BCEAD) and other datasets are consistent. As can be seen, the time delay for each algorithm is very short. Moreover, similar to the previous results, our algorithms (BCDelay and BCEAH) significantly outperform oMBEA when outputting a fixed number of results. For example, on YouTube, even

Table 3. Runtime of various algorithms for enumerating all maximal k-biplexes with $k = 1$ (in seconds).

Datasets	iTrav	iMB	symBK	BPEA	BPEAD	BPEAH
Crime	167.784	47.60	12.72	4.09	4.13	**3.90**
Ucforum	16763.21	18899.93	3334.75	36.37	**35.11**	35.17
Fjwiki	71161.72	79687.91	3739.50	86.78	**69.29**	78.70
Escorts	INF	INF	INF	19098.19	**15816.21**	16266.08



(a) Twitter ($k = 1$) (b) Twitter (output 10^4 results) (c) DBLP ($k = 1$) (d) DBLP (output 10^4 results)

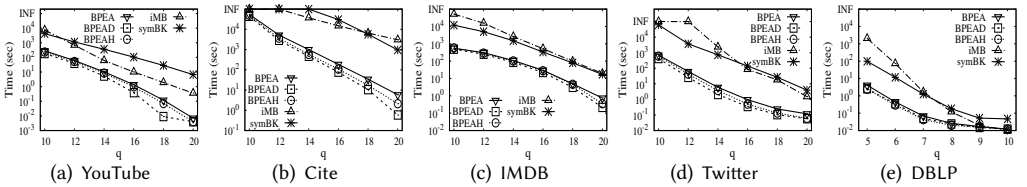
Fig. 7. Delay testing for biplex enumeration algorithms.

when outputting 10^6 results, the time costs of BCDelay, BCEAH, and oMBEA are 3.0, 2.22, and 35.4 seconds respectively. These results indicate that our algorithms are very efficient when the applications only require to output a fixed number of results. In addition, we can also see that BCDelay outperforms BCEAH if the number of returned results is not very large (e.g., $\leq 10^4$), but it may be slightly worse than BCEAH when the number of returned results is large (Fig. 5(a)). The reasons are twofold. First, BCDelay can output results in non-leaf nodes of the enumeration tree, whereas BCEAH only outputs results in leaf nodes. Second, BCDelay consumes a litter more time than BCEAH in each recursion, because BCDelay additionally needs to check the maximality of $(R_U, R_V \cup C_V)$ (or $(R_U \cup C_U, R_V)$) in each recursion.

Exp-4: Scalability testing. To evaluate the scalability of the proposed algorithms, we randomly sample 20-80% vertices and edges from the dataset Dbtropes to produce 8 subgraphs with different scales (similar results on the other datasets can also be observed). Fig. 6 shows the runtime of BCEA, BCEAD, BCEAH, and oMBEA for listing all maximal bicliques on each sampled subgraph. As can be seen, the runtime of our algorithms increases smoothly as the size of the graph increases; and the state-of-the-art oMBEA always shows the worst performance with varying parameters. Moreover, as the number of vertices or edges increases, the runtime of oMBEA also increases much faster compared to our pivot-based algorithms BCEA, BCEAD, and BCEAH. This result suggests that our algorithms exhibit good scalability in processing large bipartite graphs.

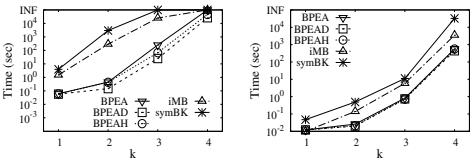
6.3 Efficiency of k-Biplex Enumerations

Exp-5: Results of enumerating all maximal k-biplexes. In this experiment, we test the performance of various algorithms for enumerating all maximal k-biplexes. Table 3 shows the runtime of BPEA, BPEAD, BPEAH, iTrav, iMB, and symBK on 4 small bipartite graphs (first 4 datasets in Table 1) with $k = 1$. Note that enumerating all maximal k-biplexes on large bipartite graphs is very costly for all algorithms, so we only use large bipartite graphs (the remaining datasets in Table 1) to evaluate the performance of different algorithms for enumerating size-constraint maximal k-biplexes. As can be seen, the algorithms we developed, BPEA, BPEAD, and BPEAH, are several orders of magnitude faster than the state-of-the-art algorithms iTrav, iMB, and symBK. For example, on Ucforum, iTrav, iMB and symBK take 16763.21, 18899.93, and 3334.75 seconds to enumerate all maximal 1-biplexes respectively, however, our algorithms consume at most 36.37 seconds. This result indicates that the proposed pivoting technique is very effective to prune unnecessary branches in enumerating all maximal k-biplexes. Moreover, we observe that BPEAD is usually faster than BPEAH. The reason may be that the k-biplex is no longer included in the subgraph induced by the 2-hop neighbors of the vertices, thus reducing the pruning performance of the 2-hop degree ordering.



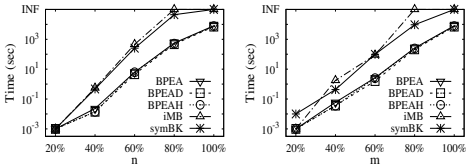
(a) YouTube (b) Cite (c) IMDB (d) Twitter (e) DBLP

Fig. 8. Runtime of various algorithms for enumerating large k-biplexes with varying q ($k = 1$).



(a) Twitter ($q = 20$) (b) DBLP ($q = 10$)

Fig. 9. Runtime of various algorithms with varying k.



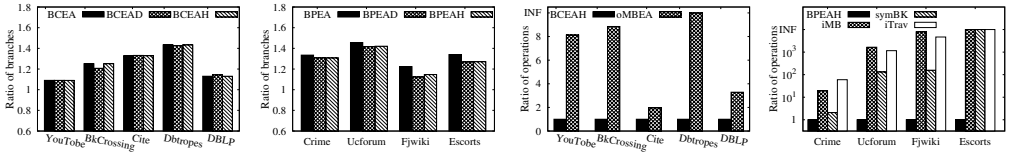
(a) varying vertices (b) varying edges

Fig. 10. Scalability testing for maximal k-biplex enumeration ($k = 1, q = 8$).

Exp-6: Delay testing. We also compare the delay times of different algorithms with iTrav, where iTrav is a polynomial delay algorithm using a reverse search technique. Fig. 7 shows the runtime of each algorithm on Twitter and DBLP with varying k and the number of returned maximal k-biplexes. The results on other datasets are consistent. As can be seen, our pivot-based algorithms, BPEA, BPEAD, and BPEAH shows better delay results compared to iTrav, iMB, and symBK, although our algorithms do not guarantee a polynomial delay property in theory. This result indicates that our pivot-based algorithms can achieve very good delay performance in practice. Moreover, the runtime of iTrav usually increases dramatically as the number of returned k-biplexes increases or the value of k increases. For instance, on Twitter, if $k = 1$, iTrav takes 0.642 seconds and 11.174 seconds to return 10^4 and 10^5 results, respectively. However, with the same parameter settings, BPEAD only takes 0.029 and 0.051 seconds to finish the computation respectively, which further confirms the efficiency of the proposed pivot-based algorithms.

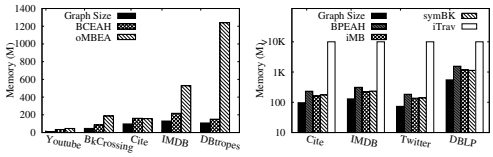
Exp-7: Results of enumerating large maximal k-biplexes. In this experiment, we evaluate the performance of various algorithms for enumerating size-constraint maximal k-biplexes on real-world large graphs. Fig. 8 and Fig. 9 show the runtime of BPEA, BPEAD, BPEAH, iMB, and symBK on real-world large graphs with varying q and k. Note that Fig. 9 only shows the results on Twitter and DBLP, and similar results can also be observed on the other datasets. Moreover, because of the reverse search technique, iTrav still needs to enumerate all maximal k-biplexes to obtain size-constraint maximal k-biplexes, which is very costly for large graphs. Thus, we do not present the runtime of iTrav in this experiment. As can be seen, our algorithms BPEA, BPEAD, and BPEAH consistently outperform the baselines (iMB and symBK) on all datasets. More specifically, under most parameter settings, BPEAD can be orders of magnitude faster than iMB and symBK. For example, on Cite, when $q = 20$, our algorithm BPEAD takes only 0.58 seconds, while symBK and iMB consume 959.72 and 5,228 seconds, respectively. These results further demonstrate the high efficiency of the proposed pivot-based enumeration algorithms.

Exp-8 Scalability testing. Here we evaluate the scalability of BPEA, BPEAD, and BPEAH. To this end, we generate 8 subgraphs by randomly sampling 20-80% of vertices or edges on Twitter. Then, we run our algorithms on these subgraphs. Fig. 10 shows the runtime of various algorithms on each subgraph. As can be seen, the runtime of our algorithms increases smoothly as the scale of the bipartite graph increases, and the baseline algorithms iMB and symBK are always worse than our algorithms under all parameter settings. With the number of vertices or edges increasing, the runtime of iMB and symBK also increases much faster compared to our BPEA, BPEAD, and



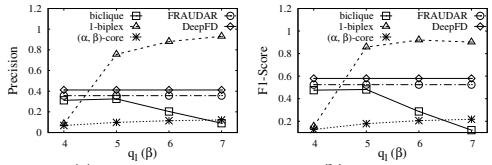
(a) biclique enumerations (b) 1-biplex enumerations (c) biclique enumerations (d) 1-biplex enumerations

Fig. 11. Pruning performance testing for pivoting techniques.



(a) biclique enumerations (b) k-biplex enumerations

Fig. 12. Memory usage for each algorithm



(a) Precision (b) $F1$-Score

Fig. 13. Accuracy results for fraud detection.

BPEAH algorithms. This result indicates that our algorithms can achieve good scalability when processing large bipartite graphs.

6.4 Further Performance Studies

Exp-9: Pruning performance testing. To study the pruning performance of our algorithms, we evaluate the ratios of the number of branches for our algorithms to the number of maximal results and the ratios of the number of atomic calculations for each baseline algorithm to the number of atomic calculations of our algorithms. Here the number of atomic calculations of an algorithm denotes the total number of arithmetic and logical operations involved in the algorithm. The results on different datasets are shown in Fig. 11. Note that in Fig. 11(c-d) we only report the results for BCEAD and BPEAD, since the results for our other algorithms are consistent. As shown in Figs. 11(a-b), the number of branches in all our pivot-based algorithms is slightly larger than the number of maximal results, even when using different ordering optimizations. This result suggests that our pivoting technique is very powerful to prune unnecessary computations. From Figs. 11(c-d), we can see that the number of atomic calculations in our algorithms can be one order of magnitude less than the state-of-the-art maximal biclique enumeration algorithm oMBEA and up to three orders of magnitude less than the state-of-the-art maximal k-biplex enumeration algorithms iMB, iTrav and symBK, which further confirm the high efficiency of proposed pivot-based algorithms.

Exp-10: Memory usages. In this experiment, we evaluate the memory usage of maximal biclique and maximal k-biplex enumeration algorithms on different datasets. The results are shown in Fig. 12. Note that we only show the space overheads of BCEAH and BPEAH, since our algorithms with different ordering techniques consume almost the same amount of space. As can be seen, the space consumption of BCEAH and BPEAH is linearly w.r.t. the size of the bipartite graph. However, some of existing algorithms may consume a large number of space. Specifically, oMBEA consumes an order of magnitude more space than the input graph size. This is because oMBEA needs to store the local subgraph for each recursive subproblem. iTrav also uses too much memory on all datasets, since it needs to store all detected maximal biplexes in the main memory. These results indicate that our algorithms are space-efficient, which also confirms the space complexity analysis in Secs. 3-5.

6.5 Case Studies

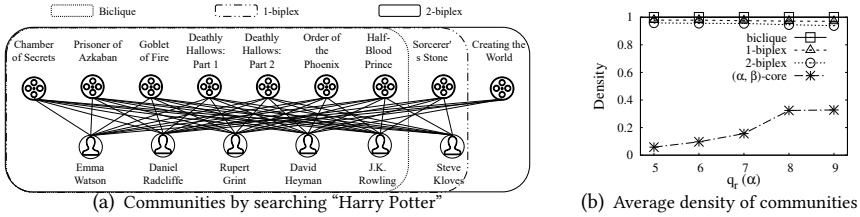
Exp-11: Fraud Detection. In this case study, we evaluate the effectiveness of the maximal biclique and maximal k-biplex models for fraud detection [18]. Same as [18], we consider a camouflage attack on "Amazon Fashion" (https://nijianmo.github.io/amazon/index.html), which contains 883,636 reviews on 186,637 products by 749,233 users, and also injected with 100 fake users, 100 fake

products, 2K fake reviews, and 2K camouflage reviews. Each fake review is randomly generated between all pairs of fake users and fake products, and camouflage reviews are randomly linked to real users (fake users) to fake products (real products), which coincides with a real camouflage attack as shown in [18]. We use the maximal biclique and maximal k-biplex to detect fraud users and products in this dataset. We compare our methods with three baselines. The first baseline is the (α, β)-core [7], which is also a cohesive subgraph model in bipartite graphs. The second baseline is the widely-used graph-based fraud detection algorithm FRAUDAR [18]. The third baseline is the state-of-the-art deep-learning based algorithm DeepFD [43]. The source code of FRAUDAR and DeepFD is publicly available, and in this experiment we use the default parameter settings as used in [18, 43]. Denote by q_l and q_r the size of the left side (product-side) and right side (user-side) vertices, respectively. Fig. 13 shows the precision and $F1$-Score ($\frac{2*precision*recall}{precision+recall}$) results with varying q_l and β, where $\alpha = 3$ and $q_r = 3$ for biclique and $q_r = 4$ for k-biplex. We can see that when q_r is small ($q_r \leq 5$), the accuracy results (both precision and $F1$-Score) detected by the maximal biclique are comparable to FRAUDAR and DeepFD, while both the precision and $F1$-Score results decrease rapidly as q_r grows to $q_r > 5$. This is because the structural constraint of the maximal biclique model is too strong which results in no biclique available when q_r is large. For 1-biplex, the accuracy results are different. Specifically, when q_r is small, the accuracy is low, but the accuracy results increase dramatically with the increase of q_r. This is because the 1-biplexes with small size are very sparse, and thus many results detected by k-biplex are not desirable. In addition, we can see that the accuracy of (α, β)-core is always very low, which is not very well for fraud detection. The accuracy of FRAUDAR and DeepFD is worse than that of maximal k-biplex model. In conclusion, the two hereditary subgraph models we focus on in this paper work well for fraud detection.

Exp-12: Community detection. Here we further conduct a case study to investigate the effectiveness of the maximal biclique and maximal k-biplex models for community detection. We use the IMDB dataset (https://datasets.imdbws.com) in this case study. which collects primary information of each film/TV production on IMDB website. We generate a bipartite graph by connecting each movie with its principal casts/crews, and the selected movies must meet the requirements of being released after 2000 and having an average rating of no less than 6.5. In this experiment, we first present the communities of "Harry Potter" detected by different cohesive subgraph models, whose results are shown in Fig. 14(a). As can be seen, the studied models can detect meaningful communities; and each detected community is highly relevant, but with slight differences. Specifically, the community detected by maximal biclique misses a movie "Sorcerer's Stone". This is because the dataset includes only a portion of principal casts for each movie, and the principal casts of "Sorcerer's Stone" in the original dataset do not include "Emma Watson", causing the result to be ignored. However, by maximal k-biplex model, such a missed result can be identified. This result suggests that the maximal k-biplex model can be used as an enhancement of the maximal biclique model in community detection. To further evaluate the quality of communities detected by different cohesive subgraph models, we also test the average density of communities detected by each cohesive subgraph model with varying q_l and α. Fig. 14(b) shows the results, where the default values of q_l and β are 5. As expected, the density of the communities detected by biclique and k-biplex models are very high. The (α, β)-core model, however, has very low densities with varying α. These results further confirm that the maximal biclique and maximal k-biplex models are indeed very effective for detecting densely-connected communities in bipartite graphs.

7 RELATED WORKS

Cohesive subgraph mining on bipartite graphs. Finding cohesive subgraphs from a bipartite graph has been recognized as an important problem in bipartite graph analysis. In addition to the maximal biclique model [1, 9, 14, 24, 28, 50] and the maximal k-biplex model [40, 48, 49], many



(a) Communities by searching "Harry Potter" (b) Average density of communities

Fig. 14. Results for community detection.

other interesting cohesive subgraph models have also been investigated, such as (α, β)-core [7, 27], k-bitruss [45, 52], and quasi biclique [19, 30, 46]. Specifically, (α, β)-core [7] is a subgraph in which the minimum degree of vertices on each side is limited by a certain threshold Liu et al. [27] presented an index-based algorithm for querying the (α, β)-core communities in optimal time. In [52], the authors proposed a k-bitruss model in bipartite graphs such that each edge is contained in at least k butterflies (i.e., 2×2 bicliques); and efficient online index algorithms, as well as peeling algorithms for computing k-bitruss, were developed in [45]. Quasi biclique is a relaxed biclique model which requires the edge density of the subgraph $H = (H_U, H_V, H_E)$ no less than a threshold $\gamma \in [0, 1)$ [19] or the minimum degree no less than $\gamma \cdot |H_U|$ ($\gamma \cdot |H_V|$) [30]. All these cohesive subgraph models do not satisfy the hereditary property, thus their solutions cannot be directly used for solving our problem.

Cohesive subgraph enumerations on traditional graphs. There are a large number of studies that focus on the cohesive subgraph enumeration problem on traditional graphs (non-bipartite graphs). Notable examples include maximal clique enumeration [5, 8, 11, 15, 33, 41], maximal k-plex enumeration [4, 12, 26, 51], s-clique enumeration [3, 31], and γ-quasi clique enumeration [29, 35]. All existing enumeration algorithms can be roughly classified into two categories. The first category is the output-sensitive algorithms, in which the total time complexity of the algorithm is polynomial with respect to (w.r.t.) the size of the outputs [3, 4, 8, 10, 11]. However, the practical performance of these output-sensitive solutions is often very poor; and they typically cannot be used to handle large real-world graphs. Another category of algorithms, which do not necessary to be output-sensitive, have exponential time complexity w.r.t. the number of vertices in the worst-case, but they are often very efficient when handling real-world sparse graphs due to some carefully-designed pruning techniques [12, 15, 15, 33, 41, 51]. All these existing algorithms are mainly tailored to non-bipartite graphs. It is quite non-trivial to extend these techniques to handle bipartite graphs, as the structures of cohesive subgraphs on bipartite graphs are very different from that on non-bipartite graphs.

8 CONCLUSION

In this paper, we investigate the problem of enumerating all maximal subgraphs on bipartite graphs that satisfy the hereditary property. To solve this problem, we develop a general enumeration framework which utilizes a novel and carefully-designed pivoting principle. Based on this general framework, we then propose new pivoting techniques to enumerate all maximal bicliques and maximal k-biplexes in bipartite graphs. We show that the time complexity of our maximal biclique enumeration algorithm is near optimal. For practical applications, some optimization techniques are further developed to enumerate size-constraint maximal bicliques and maximal k-biplexes. Finally, we conduct extensive experiments to evaluate the proposed algorithms; and the results demonstrate the efficiency, scalability, and effectiveness of the proposed solutions.

ACKNOWLEDGMENTS

This work was partially supported by (i) National Key Research and Development Program of China 2020AAA0108503, (ii) NSFC Grants U2241211, 62072034, U1809206 and (iii) CCF-Huawei Populus Grove Fund. Rong-Hua Li is the corresponding author of this paper.

REFERENCES

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

[2] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR* cs.DS/0310049 (2003).

[3] Rachel Behar and Sara Cohen. 2018. Finding All Maximal Connected s-Cliques in Social Networks. In *EDBT*. 61–72.

[4] Devora Berlowitz, Sara Cohen, and Benny Kimelfeld. 2015. Efficient Enumeration of Maximal k-Plexes. In *SIGMOD*. 431–444.

[5] Coenraad Bron and Joep Kerbosch. 1973. Finding All Cliques of an Undirected Graph (Algorithm 457). *Commun. ACM* 16, 9 (1973), 575–576.

[6] 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.

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

[8] Lijun Chang, Jeffrey Xu Yu, and Lu Qin. 2013. Fast Maximal Cliques Enumeration in Sparse Graphs. *Algorithmica* 66, 1 (2013), 173–186.

[9] 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.

[10] Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. 2008. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *J. Comput. Syst. Sci.* 74, 7 (2008), 1147–1159.

[11] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. 2016. Sublinear-Space Bounded-Delay Enumeration for Massive Network Analytics: Maximal Cliques. In *ICALP*, Vol. 55. 148:1–148:15.

[12] Alessio Conte, Tiziano De Matteis, Daniele De Sensi, Roberto Grossi, Andrea Marino, and Luca Versari. 2018. D2K: Scalable Community Detection in Massive Networks via Small-Diameter k-Plexes. In *KDD*. 1272–1281.

[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] David Eppstein, Maarten Löffler, and Darren Strash. 2010. Listing All Maximal Cliques in Sparse Graphs in Near-Optimal Time. In *ISAAC*, Vol. 6506. 403–414.

[16] Fedor V. Fomin and Dieter Kratsch. 2010. *Exact Exponential Algorithms*. Springer.

[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] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. FRAUDAR: Bounding Graph Fraud in the Face of Camouflage. In *KDD*. 895–904.

[19] 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.

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

[21] John M. Lewis and Mihalis Yannakakis. 1980. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *J. Comput. Syst. Sci.* 20, 2 (1980), 219–230.

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

[23] 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.

[24] 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.

[25] Rong-Hua Li, Qiushuo Song, Xiaokui Xiao, Lu Qin, Guoren Wang, Jeffrey Xu Yu, and Rui Mao. 2022. I/O-Efficient Algorithms for Degeneracy Computation on Massive Networks. *IEEE Trans. Knowl. Data Eng.* 34, 7 (2022), 3335–3348.

[26] Don R. Lick and Arthur T. White. 1970. k-Degenerate graphs. *Canadian Journal of Mathematics* 22, 5 (1970), 1082–1096.

[27] 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.

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

[29] Guimei Liu and Limsoon Wong. 2008. Effective Pruning Techniques for Mining Quasi-Cliques. In *ECML/PKDD*, Vol. 5212. 33–49.

[30] Xiaowen Liu, Jinyan Li, and Lusheng Wang. 2008. Quasi-bicliques: Complexity and Binding Pairs. In *COCOON 2008*, Vol. 5092. 255–264.
[31] John W. Moon and Leo Moser. 1965. On cliques in graphs. *Israel journal of Mathematics* 3, 1 (1965), 23–28.
[32] Azam Sheikh Muhammad, Peter Damaschke, and Olof Mogren. 2016. Summarizing Online User Reviews Using Bicliques. In *SOFSEM*, Vol. 9587. 569–579.
[33] Kevin A. Naudé. 2016. Refined pivot selection for maximal clique enumeration in graphs. *Theor. Comput. Sci.* 613 (2016), 28–37.
[34] René Peeters. 2003. The maximum edge biclique problem is NP-complete. *Discret. Appl. Math.* 131, 3 (2003), 651–654.
[35] Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *KDD*. 228–238.
[36] Ardian Kristanto Poernomo and Vivekanand Gopalkrishnan. 2009. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *KDD*. 697–706.
[37] Erich Prisner. 2000. Bicliques in Graphs I: Bounds on Their Number. *Comb.* 20, 1 (2000), 109–117.
[38] Ron Rymon. 1992. Search through Systematic Set Enumeration. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*. 539–550.
[39] 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.
[40] 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.
[41] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.* 363, 1 (2006), 28–42.
[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] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. 2018. Deep Structure Learning for Fraud Detection. In *ICDM*. 567–576.
[44] 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.
[45] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient Bitruss Decomposition for Large-scale Bipartite Graphs. In *ICDE*. 661–672.
[46] Lusheng Wang. 2010. Near Optimal Solutions for Maximum Quasi-bicliques. In *COCOON*, Vol. 6196. 409–418.
[47] Kaiqiang Yu and Cheng Long. 2022. Maximum k-Biplex Search on Bipartite Graphs: A Symmetric-BK Branching Approach. *CoRR* abs/2208.13207 (2022).
[48] Kaiqiang Yu, Cheng Long, Shengxin Liu, and Da Yan. 2022. Efficient Algorithms for Maximal k-Biplex Enumeration. In *SIGMOD*. 860–873.
[49] Kaiqiang Yu, Cheng Long, Deepak P, and Tanmoy Chakraborty. 2021. On Efficient Large Maximal Biplex Discovery. *IEEE Trans. Knowl. Data Eng.* (2021).
[50] 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), 110.
[51] Yi Zhou, Jingwei Xu, Zhenyu Guo, Mingyu Xiao, and Yan Jin. 2020. Enumerating Maximal k-Plexes with Worst-Case Time Guarantee. In *AAAI*. 2442–2449.
[52] Zhaonian Zou. 2016. Bitruss Decomposition of Bipartite Graphs. In *DASFAA*, Vol. 9643. 218–233.

Received October 2022; revised January 2023; accepted February 2023