On Efficient Large Maximal Biplex Discovery

Kaiqiang Yu, Cheng Long, Deepak P and Tanmoy Chakraborty

Abstract—Cohesive subgraph discovery is an important problem in bipartite graph mining. In this paper, we focus on one kind of cohesive structure, called k-biplex, where each vertex of one side is disconnected from at most k vertices of the other side. We consider the large maximal k-biplex enumeration problem which is to list all those maximal k-biplexes with the number of vertices at each side at least a non-negative integer \( \theta \). This formulation, we observe, has various applications and targets to find non-redundant results by excluding non-maximal ones. Existing approaches suffer from massive redundant computations and can only run on small and moderate datasets. Towards improving scalability, we propose an efficient tree-based algorithm with two advanced strategies and powerful pruning techniques. Experimental results on real and synthetic datasets show the superiority of our algorithm over existing approaches.

Index Terms—Bipartite graph, large maximal biplex, graph mining, maximal subgraph enumeration.

1 INTRODUCTION

Many real-world applications involve two types of entities and can be modeled as bipartite graphs. Some examples include customer-item graphs [1], author-paper graphs [2] and gene co-expression graphs [3]. These applications often require to discover cohesive subgraphs for dense group detection. One type of cohesive subgraph structure on bipartite graphs is k-biplex, which has been extensively studied [4], [5], [6]. A k-biplex is a subgraph of a bipartite graph, induced by two sets of vertices each from one side of the graph, such that each vertex in one set is disconnected from at most k vertices in the other set. There usually exist numerous k-biplexes; yet the majority of them are of small size and not interesting. Therefore, we focus on those k-biplexes with the sizes of the vertex sets at least a given non-negative integer threshold \( \theta \) [4]. We call these k-biplexes large maximal k-biplexes and the problem of finding all large maximal k-biplexes in a bipartite graph the large maximal k-biplex enumeration problem. For example, in Fig. 1(a), the induced subgraph \( H(\{u_1, u_2, u_3\}, \{v_1, v_2, v_3, v_4\}) \) is a large 2-biplex w.r.t. \( \theta=3 \), but not a maximal one since there is a superset \( H(\{u_1, u_2, u_3, u_4\}, \{v_1, v_2, v_3, v_4\}) \) which is also a 2-biplex. The problem has various applications such as collaborative users detection [7], fraudulent reviewer group detection [8], product recommendation [9] and protein network analysis [10].

Existing Methods and Challenges. A simple but powerful technique for enumeration problems is binary partition or backtracking. Intuitively, it recursively partitions the search space based on certain vertices, yielding all solutions including those vertices and all excluding them [11]. The authors in [4] followed this framework and developed an algorithm which we denote by Enum. To be specific, Enum constructs prefix trees to navigate the search space and employs backtracking and branch-and-bound techniques to prune subtrees that cannot yield any large maximal k-biplexes. The pruning techniques employed therein improve its effectiveness with increasing \( \theta \), and are not effective for very low values of \( \theta \). Moreover, its framework with the simple enumeration and partition leads to massive redundant computations, which limits the practicality and scalability. For example, all k-biplexes satisfying the size constraints but not maximal would be enumerated once and then be discarded during the procedure of maximality checking. Unfortunately, there exist numerous such k-biplexes necessitating much wasteful computation.

A k-biplex in a bipartite graph can be converted to a \((k + 1)\)-plex in a general graph by graph inflation, which adds edges to each pair of vertices at the same side. Although there exist efficient maximal k-plex enumeration algorithms [12], [13], they suffer from the following practical and scalable issues. (1) Graph inflation yields a dense graph with many edges. Even for the sparse bipartite graph \( G(L, R, E) \) with no edges and \(|L|=|R|=n\), the yielded graph \( \hat{G} \) would have a density \( d=(n^2-n)/2n^2 \) nearly 50%. (2) The resultant \( k \)-plexes (and their corresponding k-biplexes) may not satisfy the size constraint (imposed by \( \theta \)). While we can post-process the results to filter out small ones, this would be inefficient due to numerous small k-biplexes. We compare with such a baseline in experiments, and show that it incurs a significant amount of redundant computations.

Contributions. To overcome these challenges, we propose an efficient tree-based algorithm called iMB with two advanced strategies and powerful pruning techniques. Basically, we build upon insights from binary partition and backtracking techniques. Let \( L \) and \( R \) be the sets of vertices at the left and right sides of a bipartite graph, respectively. We first select a primary enumeration vertex set \( L \) and then iteratively explore its subsets in a depth-first way on a prefix tree based on its power set. For each fixed subset \( X \) of \( L \), we then enumerate possible subsets \( Y \) of \( R \) that can form large maximal k-biplexes with \( X \). To enhance this framework, we first propose an early stopping strategy to prune those branches that only generate non-maximal k-biplexes. In addition, we employ the advanced vertex expansion strategy to extend from non-maximal k-biplexes to maximal ones, which highly reduces the redundant computation. Moreover, we propose some pruning techniques to improve its scalability. Finally, we evaluate our algorithms on both real and synthetic datasets, and the results demonstrate iMB’s superiority over existing approaches.
2 Problem Definition

In this paper, we consider an undirected, unweighted bipartite graph $G(L, R, E)$ with two disjoint vertex sets $L, R$ and an edge set $E$. We define $H(X, Y)$ to be an induced bipartite subgraph, where $X \subseteq L$ and $Y \subseteq R$, and define $N_R(v)$ for $v \in L$ to be the set of vertices that are adjacent to $v$ in $R$. In addition, we define $\delta(v, R) = |N_R(v)|$. We now outline cohesive bipartite subgraph structures exploited in this paper.

**Definition 1 (k-biplex).** A $k$-biplex $B(X, Y)$ is an induced bipartite subgraph of bipartite graph $G(L, R)$ such that

- $|Y| - \delta(u, Y) \leq k, \forall u \in X$ and
- $|X| - \delta(v, X) \leq k, \forall v \in Y$,

where $k \geq 0$ is a non-negative integer, $X \subseteq L$ and $Y \subseteq R$.

Note that the threshold parameter $k$ represents the number of missing edges that each vertex can tolerate. Clearly, $k$-biplexes w.r.t. $k = 0$ reduce to classical bicliques. We then show the importance of an important property of $k$-biplexes.

**Property 1 (Hereditary property).** If $H(X, Y)$ is a $k$-biplex, any subgraph $H'(X', Y')$ with $X' \subseteq X, Y' \subseteq Y$ is a $k$-biplex.

**Proof:** Based on the definition, we have $|X| - \delta(v, X) \leq k, \forall v \in Y'$. We can then imply $|X'| - \delta(v, X') \leq |X'| + |X| - \delta(v, X') \leq k$, since $|X| - \delta(v, X') \leq k$. Similarly, we can obtain $|Y'| - \delta(u, Y') \leq k, \forall u \in X'$. □

We focus on maximal $k$-biplexes since they, by definition, encompass all non-maximal ones.

**Definition 2 (Maximal k-biplex).** A $k$-biplex $B(X, Y)$ of the bipartite graph $G$ is maximal if and only if it is not contained by any $k$-biplex $B'(X', Y')$ with $X' \subseteq X$ and $Y' \subseteq Y$.

In this paper, we use $MB$ to denote a maximal $k$-biplex. There usually exist numerous $MB$s; yet the majority of them are of small sizes and not interesting since they are generated by the simple combinations of disconnected vertices. Therefore, we propose to investigate those large $MB$s with the sizes of the vertex sets at least a given integer threshold $\theta$, i.e., $|X| \geq \theta, |Y| \geq \theta$ and $\theta \geq 0$. We call these $MB$s large $MB$s. We formalize the problem exploited in this paper as follows.

**Problem 1 (Large Maximal k-biplex Enumeration).** Given a bipartite graph $G(L, R, E)$, two non-negative integers $k$ and $\theta$, the Large Maximal k-biplex Enumeration problem aims to list all maximal $k$-biplexes $B(X, Y)$ whose sizes of the vertex sets are at least $\theta$, i.e., all large $MB$s.

We first derive a powerful pruning technique (called CorePruning) by exploiting the following relationship between $k$-biplexes and $k$-cores. Here, $k$-core is a subgraph with each vertex having degree at least $k$.

**Lemma 1.** A large $k$-biplex w.r.t. $\theta$ is a $(\theta - k)$-core subgraph.

**Proof:** Let $H(X, Y)$ be a $k$-biplex w.r.t. $\theta$. We have $|X| - k \leq \delta(v, X), \forall v \in Y$. Since $|X| \geq \theta$, we imply $\delta(v, X) \geq \theta - k$. Similarly, we can obtain $\delta(u, Y) \geq \theta - k, \forall u \in X$. Therefore, all vertices have degree at least $\theta - k$. □

For example, $G'$ in Fig. 1(a), which is a large 0-biplex w.r.t. $\theta=3$, is a 3-core. Based on Lemma 1, we can first conduct a core decomposition, which computes all cores in linear time by iteratively removing the vertex with the smallest degree, and then prune all vertices not in any $(\theta - k)$-core. Note that this pruning technique is not specific to any algorithm. It can also be used to boost existing algorithms such as Enum [3] and D2K [13] as a pre-processing step, which will be studied in our experiments.

3 Proposed Method: IMB

We first present a basic algorithm in Section 3.1 and then enhance the algorithm with two advanced techniques in Sections 3.2 and 3.3. We call the resulting algorithm IMB (efficient large Maximal k-Biplex enumeration).

3.1 Overview of Basic Framework

Our solution is based on a general tree-based framework which follows the classical paradigm of binary partition and backtracking [11]. The intuition is to enumerate all combinations of vertex subsets $X \subseteq L$ and $Y \subseteq R$, and iteratively check the maximality for each of them. Basically, it starts with selecting a primary enumeration vertex set (w.l.o.g. $L$) and exploring, in a depth-first way, its subsets on a prefix tree. The prefix tree is based on $L$’s power set w.r.t. certain vertex order. Each node of the prefix tree represents a subset of $L$, denoted by $X$. We use “node” to denote “prefix tree node” in this paper. For each node $X$, it then enumerates the subsets of $R$, which are also maintained in a prefix tree, and finds potential large $MB$s. For instance, Fig. 1(b) shows the prefix tree constructed for $L$ of graph $G(L, R)$ shown in Fig. 1(a). All examples in this paper are executed based on a certain order of vertex index. W.l.o.g, we assume $L = \{v_1, v_2, v_3\}$ and $R = \{u_1, u_2, u_3, u_4, u_5\}$ follow the order of $u_1, u_2, u_3, u_4, u_5$, respectively. With this order, the algorithm would process the nodes of the prefix tree in the order of $X_0, X_1, ..., X_{15}$ shown in Fig. 1(b).

Specifically, the basic framework is shown in Alg. 1 and executes as follows. For a given bipartite graph $G(L, R)$ shown in Fig. 1(a) and parameters $\theta=1, k=1$, it starts with the first tree node $X_0=\emptyset$, and expands $X_0$ by iteratively adding each vertex $u \in cand_p(X_0)$ to $X_0$ (Line 1-2). Here, $cand_p(X)$ denotes the set of vertices $u$ that $X \cup \{u\}$ has not been explored before, i.e., $cand_p(X_0)=\{u_i\}_{i=1}^4$. It adds $u_1$ to $X_0$ and then arrives at a new node $X_1=\{u_1\}$. We have $cand_p(X_1)=\{u_i\}_{i=2}^4$. Since $X_1$ is fixed, it refines the search space $R$ based on the fact that vertices with the degree less than $|X_1| - k$ cannot be part of any large $MB$ with $X_1$. Formally, it defines the refined search space as $\Phi(X) = \{v \in R | \delta(v, X) \geq |X| - k\}$ (Line 3). Note that $\Phi(X)$ is nested, i.e., $\Phi(X \cup \{u\}) \subseteq \Phi(X)$. We have $\Phi(X_1)=\{v_i\}_{i=1}^4$. It then enumerates all combinations $(X_1, Y_1)$, $Y_1 \subseteq \Phi(X_1)$ and checks the maximality for each of them. Following this, it checks the size constraints and repeats the same procedure for extending $X_1$ to the next node (Line 9-10). Note that if $|X_1| + |cand_p(X_1)| < \theta$ or $|\Phi(X_1)| < \theta$, then the node $X_1$ does not need to be further extended, since $cand_p(X_1)$ and $\Phi(X_1)$ are non-increasing. Otherwise, it extends $X_1$ with a recursive call. After finishing the exploration of $X_1$, it adds $u$ to $cand_g(X_1)$ which is employed to avoid redundant computations (Line 11). Here, $cand_g(X)$ is the set of vertices $w \in L$ that $X \cup \{w\}$ has
been explored before. For example, we have \( cand_q(X_k) = \{u_2\} \) in Fig. 1(b), since \( X_3 = X_6 \cup \{u_2\} \) has already been processed.

Here, we elaborate implementation of the aforementioned procedure of listing MBs (Line 7). Basically, it can be done by enumerating all subsets \( Y_i \subseteq \Phi(X) \) and checking whether \( (X', Y_i) \) is a MB. To speedup this procedure, we can maintain all subsets \( Y_i \) of \( \Phi(X) \) and check the maximality for each subset \( Y_i \) by enumerating all subsets \( \mathcal{M}_B \). Based on the above lemma, we can refine \( \Phi(X) \) by removing those vertices in set \( T \). Since we need to construct the similar prefix tree over \( \Phi(X) \), refining \( \Phi(X) \) leads to a smaller search space. Intuitively, the smaller the size of the vertex set \( \Phi(X) \), the more likely it satisfies the early stop conditions. For example, after removing \( T \) from \( \Phi(X) \), we may have \( |\Phi(X)| < \theta \) or \( \Phi(X) \not\subseteq \mathcal{N}_R(u) \), which leads to a pruning operation for \( X \). We use the following example to illustrate the strategy.

\section*{3.2 Early Stopping Strategy}

In this section, we propose to prune some branches early in the prefix tree instead of checking the redundancy for all MBs at the end as existing methods do \cite{14}.

**Lemma 2.** Given a subset \( X \subseteq L \), if there exists a vertex \( u \in \text{cand}_q(X) \) that \( \Phi(X) \subseteq \mathcal{N}_R(u) \), then for any subset \( S \subseteq \text{cand}_p(X) \) and \( Y \subseteq \Phi(X) \), \( H(X \cup S, Y) \) is not a MB.

\textbf{Proof:} It can be easily proved by contradiction. Suppose that \( (X \cup S, Y) \) is a MB. Based on \( \Phi(X) \subseteq \mathcal{N}_R(u) \) and \( Y \subseteq \Phi(X) \), we imply \( Y \subseteq \mathcal{N}_R(u) \). Furthermore, we have \( |Y| - \delta(u, Y) = 0 \leq k \) and \( |X'| - \delta(v, X') \leq k \), \( \forall v \in Y, X' = X \cup S \cup \{u\} \). Thus, we can form a larger \( k \)-biplex by adding \( u \) to \( X \cup S \), which leads to a contradiction.

Lemma 2 essentially says that if there exists an expanded vertex whose neighbors form a superset of \( \Phi(X) \), then any expansion of the current \( X \) will not yield a MB. Therefore, we can stop early without expanding some nodes and those in the subtrees rooted at these nodes. Note that \( \text{Enum} \) would enumerate all those non-maximal \( k \)-biplexes.

\textbf{Example 1.} Suppose \( k = 1 \) and \( \theta = 1 \) for Fig. 1(a). We consider the current node \( X_0 \) of the prefix tree in Fig. 1(b). Since \( X_1 \) and its subtrees have been processed, \( u_1 \) will be added to \( \text{cand}_q(X_0) \). We can stop at \( X_9 \) due to the fact that \( \Phi(X_9) = \{v_1\} \subseteq \mathcal{N}_Y(u_1) \). The next node \( X_{13} \) will then be processed and \( u_2 \) will be added to \( \text{cand}_q(X_{13}) \).

It should be noted that the effectiveness of the early stopping strategy based on Lemma 2 depends on the size of \( \Phi(X) \), i.e., the smaller the size of \( \Phi(X) \), the better the strategy. Therefore, we further develop a technique to refine \( \Phi(X) \). We first define a set \( \Gamma = \{Y_i \subseteq \Phi(X) | H(X \cup S, Y) \text{ is a MB w.r.t. subgraph } (X, \Phi(X)) \} \) and a refined set \( \tilde{\Gamma} = \{Y_i \subseteq \Gamma | v \in \text{cand}_q(X), Y_i \subseteq \mathcal{N}_R(u) \} \), where \( X \subseteq L \) denotes a prefix tree node.

**Lemma 3.** Let \( X \subseteq L \) denote a prefix tree node. Then any MB \( B(X \cup S, Y) \) formed by subset \( S \subseteq \text{cand}_p(X) \) and \( Y \subseteq \Phi(X) \) does not include any vertex in the set \( T = \{v \in \Phi(X) | \forall v_i \in \text{cand}_q(X), Y_i \subseteq \mathcal{N}_R(u) \} \).

**Proof:** We prove it by contradiction. Suppose we have a MB \( B(X \cup S, Y) \) formed by subset \( S \subseteq \text{cand}_p(X) \) and \( Y \subseteq \Phi(X) \) that contains nonempty vertex set \( T \subseteq T \). Based on the property of \( k \)-bip lexes, we can remove \( S \) from \( B(X \cup S, Y) \), and the resulting subgraph \( H(X, Y) \) is still a \( k \)-biplex. Then we can imply \( Y \subseteq Y_i \), \( \forall Y_i \in \Gamma \). Since \( T \subseteq Y_i \), we can obtain \( Y \not\subseteq \tilde{\Gamma} \). Otherwise, \( T \) cannot be included in \( Y_i \) due to the fact that \( \exists Y_i \in \tilde{\Gamma}, T \not\subseteq Y_i \). Therefore, \( Y \) must be refined from \( \Gamma \), and we thus imply \( \exists u \in \text{cand}_q(X), Y \subseteq \mathcal{N}_R(u) \). We can obtain that \( (X \cup S, Y) \) is not a MB, which leads to a contradiction.

Fig. 2. Pruning techniques illustrated (vertices in current \( X \cup \Phi(X) \), \( \text{cand}_p(X) \) and \( \text{cand}_q(X) \) are black, white and gray, respectively).

**3.3 Advanced Node Expansion**

While the early stopping strategy reduces redundant non-maximal solution enumeration and checking, its simple expansion strategy still incurs some unnecessary computations. We propose an advanced node expansion strategy for better efficiency. Intuitively, it expands the tree node \( X \) by adding a group of vertices at each step instead of only one vertex, which would help to find the next MBs sooner.

**Lemma 4.** Given a subset \( X \subseteq L \), if there exists a vertex \( u \in \text{cand}_p(X) \) that \( \Phi(X) \subseteq \mathcal{N}_R(u) \), then for any \( k \)-biplex \( H(X \cup S, Y) \) formed by \( S \subseteq \text{cand}_p(X) \) and \( Y \subseteq \Phi(X) \), we have that \( H(X \cup S \cup \{u\}, Y) \) is also a \( k \)-biplex.

**Proof:** It could be proved similarly as Lemma 2.
Lemma 4 enables it to add such vertices \( u \in \text{cand}_p(X) \) that \( \Phi(X) \subseteq N_G(u) \) to \( X \), since all MBs generated by this branch must include \( u \). Therefore, it helps to prune some redundant extensions for the current node \( X \) by removing such vertices from \( \text{cand}_p(X) \). In fact, it would be enough to only require the vertex \( u \) to cover all vertices of \( Y \) in \( \overline{\Gamma} \) instead of \( \Gamma \). This is because all \( k \)-biplexes \( H(X, Y_i) \) formed by subsets \( Y_i \subseteq \overline{\Gamma} \setminus \Gamma \) are contained in previous MBs and will lead to a duplication no matter whether \( \mathcal{N}_R(u) \) include such subsets or not.

**Example 3.** Suppose \( k = 1 \) and \( \theta = 1 \) for Fig. 1(a). We consider current node \( X_1 = \{u_1\} \) with \( \Phi(X_1) = \{v_1, v_2, v_3, v_4\} \) and \( \text{cand}_p(X_1) = \{u_2, u_3, u_4\} \). Based on Lemma 4, we can remove \( u_3 \) from \( \text{cand}_p(X_1) \) and add it to \( X_1 \) such that \( X_1 = \{u_1, u_3\} \). Note that the next node is \( X_2 = \{u_1, u_3, u_2\} \) instead of \( X_2 = \{u_1, u_2\} \). Moreover, \( X_2, X_3, X_6 \) and \( X_7 \) will be pruned since \( u_3 \) has been removed from \( \text{cand}_p(X_1) \).

To further prune unnecessary branches, we propose to refine \( \text{cand}_p(X) \). Unfortunately, several \( k \)-biplexes \( H(X, Y_i) \) with the same \( X \) are highly coupled with each other, for which we only maintain \( \Phi(X) \) for each node \( X \) instead of all subsets \( Y_i \). It is hard to build an equivalent condition for the candidate set only based on \( \Phi(X) \), which means there are still some vertices in \( \text{cand}_p(X) \) that cannot form any MB with \( X \). We then develop a strong condition for \( \text{cand}_p(X) \) based on all subsets \( Y_i \in \overline{\Gamma} \).

**Lemma 5.** Let \( X \subseteq L \) denote a prefix tree node. If for any \( Y_i \in \overline{\Gamma} \), the vertex \( u \in \text{cand}_p(X) \) cannot be added to \( X \) to form a \( k \)-biplex \( H(X \cup \{u\}, Y_i') \) with \( Y_i' \subseteq Y_i \) (sufficient condition \( A \)), then any MB \( B(X \cup S, Y) \) formed by \( S \subseteq \text{cand}_p(X) \) and \( Y \subseteq \Phi(X) \) does not include \( u \).

**Proof:** We prove it by contradiction. Suppose there exists a MB \( B(X \cup S, Y) \) formed by \( S \subseteq \text{cand}_p(X) \) and \( Y \subseteq \Phi(X) \) which includes \( u \). We can deduce that \( u \in S \) and \( H(X, Y) \) is a \( k \)-biplex. We can further obtain that \( Y \subseteq Y_i, \forall Y_i \in \Gamma \). Two different cases for \( Y \) need to be further exploited as follows.

- \( i) \) \( Y_i \subseteq \overline{\Gamma} \): In this case, there exists a vertex \( u \in \text{cand}_q(X) \) that \( Y \subseteq N_G(w) \). Therefore, a larger MB \( B(X \cup S \setminus \{Y\}, Y) \) can be formed, which leads to a contradiction to the assumption.
- \( ii) \) \( Y_i \in \overline{\Gamma} \): In this case, we can obtain a larger \( k \)-biplex \( H(X \cup \{u\}, Y) \) by removing \( S \setminus \{u\} \) from the MB \( B \), which contradicts to the condition. We thus finish the proof.

Lemma 5 indicates that we can remove a vertex \( u \) from the candidate set \( \text{cand}_p(X) \), if \( u \) satisfies some conditions. Intuitively, it turns to check all \( Y_i \in \overline{\Gamma} \) instead of \( \Phi(X) \), by which we can derive a tighter bound. In the following, we develop an equivalent condition which is easy to check.

**Lemma 6.** Given a large \( k \)-biplex \( H(X, Y) \) w.r.t. \( \theta \), a vertex \( u \) can be added to \( X \) to form a large \( k \)-biplex \( H(X \cup \{u\}, Y') \) with \( Y' \subseteq Y \) (condition \( \neg A \)) if and only if \( \delta(u, Y) \geq \theta - k \) and \( |\Phi(X \cup \{u\})| \geq \theta \).

**Proof:** We prove it by two directions, and start with “\( \Rightarrow \)”.

If \( H(X \cup \{u\}, Y') \) with \( Y' \subseteq Y \) is a \( k \)-biplex, we can obviously obtain \( |\Phi(X \cup \{u\})| \geq |Y'| \geq \theta \) and \( \delta(u, Y) \geq \delta(u, Y') \geq |Y'| - k \geq \theta - k \).

For the other direction “\( \Leftarrow \)”, we consider two different cases.

- \( i) \) \( \delta(u, Y) \geq \theta \): In this case, we prove it by showing that \( (X \cup \{u\}, N_Y(u)) \) is a \( k \)-biplex. Since \( \delta(u, Y) \geq \theta \) and \( H(X', Y) \) is an \( k \)-biplex, we can deduce that \( (X, N_Y(u)) \) is also a \( k \)-biplex. Then \( u \) can be added to \( X \) to form the desired \( k \)-biplex \( (X \cup \{u\}, N_Y(u)) \), since \( \delta(u, Y) - \delta(u, Y') = 0 \) and \( \forall v \in N_Y(u), |X \cup \{u\}| = |X| - \delta(v, X \cup \{u\}) \leq k. \)

**4 Experiments**

We evaluate our algorithms on seven real-world datasets (http://konect.uni-koblenz.de/networks/) taken from various domains (see Table 1). In addition, we use synthetic Erdos-REyni (ER). Specifically, we first generate a fixed number of vertices and then randomly add edges between pairs of vertices. We then compare the proposed algorithm \( iMB \) with two existing methods – \( Enum \) [4] and \( D2K \) [13]. \( Enum \) accepts two parameters, \( k \) and \( \theta \), and lists all large MBs, i.e., maximal \( k \)-biplexes with the size at each side at least \( \theta \). \( D2K \) accepts two parameters \( k' \) and \( \theta' \) and lists all \( k' \)-biplexes with the size at least \( \theta' \). For \( D2K \), we set \( k' = k+1 \) and \( \theta' = 2 \theta \) and apply a post-processing step to filter out those MBs that are not large (i.e., with the size at one side smaller than \( \theta \)). For fair comparison, no parallelism is enabled. Moreover, we denote these algorithms with the pre-processing procedure \( CorePruning \) as \( iMB^+ \), \( Enum^+ \) and \( D2K^+ \). The algorithms are implemented in C++, and all experiments are executed on a Linux machine with a 2.66GHz processor and 32GB of memory. The source codes are available at https://personal.ntu.edu.sg/cf,long/code/iMB.zip.

**# of MBs.** We show the results of the effect of varying \( \theta \) and \( k \) on the number of large MBs in Fig. 3. The results that are out of the y-range are not shown. Generally, the number of MBs exponentially decreases or increases as \( \theta \) or \( k \) increases, respectively. Note that

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clfat</td>
<td>Miscellaneous</td>
<td>400</td>
<td>802</td>
</tr>
<tr>
<td>Opsahl</td>
<td>Authorship</td>
<td>38,741</td>
<td>58,595</td>
</tr>
<tr>
<td>Writer</td>
<td>Affiliation</td>
<td>135,569</td>
<td>144,340</td>
</tr>
<tr>
<td>Actors</td>
<td>Affiliation</td>
<td>520,223</td>
<td>1,470,404</td>
</tr>
<tr>
<td>IMDB</td>
<td>Communication</td>
<td>1,324,748</td>
<td>3,782,463</td>
</tr>
<tr>
<td>DBLP</td>
<td>Authorship</td>
<td>5,425,963</td>
<td>8,649,016</td>
</tr>
<tr>
<td>Google</td>
<td>Hyperlink</td>
<td>20,200,070</td>
<td>14,693,125</td>
</tr>
</tbody>
</table>
we use the indexes located at the top of Fig. 3(a) as the x-axis for IMDB. Based on these results, we can determine the meaningful settings of \(\theta\) and \(k\) for different datasets. We set \(\theta\) to be 5, 6, 8, 8, 10, 10, 19 on Writer, Cfat, Actor, Opsahl, DBLP, Google, IMDB respectively, and \(k\) to be 1 on all datasets by default.

**Comparison to D2K+.** D2K+ cannot handle large datasets within limited time (120h) due to the graph inflation. We compare with D2K+ on two small datasets in Figs. 5(a-b), where iMB+ is faster than D2K+. This is because D2K+ executes in a dense graph, and needs to enumerate some maximal \(k\)-biplexes with size smaller than \(\theta\), although they are finally discarded.

**Performance on biclique setting (\(k=0\)).** Since 0-biplexes are bicliques, we compare with maximal biclique enumeration algorithms, LCM [13] and iMBEA [14], in Table 2. We omit the dataset “Google”, since all algorithms cannot finish within a reasonable time. While strategies based on lemma 3 are ineffective at this setting, iMB outperforms LCM and iMBEA on most datasets. These results verify the effectiveness of the pruning techniques that are newly introduced in this paper.

**4.2 Evaluation on Synthetic Datasets**

We evaluate the scalability of iMB+ on synthetic graphs of varying numbers of vertices and densities. We report the running time of returning the first 1000 large MBs and denote INF as 24 hours.

**Varying vertices.** We vary the scale of graphs from \(10^2\) vertices and \(10^3\) edges to \(10^6\) vertices and \(10^7\) edges. Fig. 7(a) shows the running time of iMB+ and Enum+ on different graphs. We observe that the running time increases as the size of the graph grows, since the larger the size of the graph, the larger the search space. Since we return a fixed number of MBs, the running time increases with the increase of \(\theta\). Moreover, it is clear that iMB+ is more scalable than Enum+.

**Varying edges.** We generate synthetic graphs with \(10^5\) vertices, and vary the number of edges from \(10^4\) to \(10^7\). The running times of iMB+ and Enum+ are presented in Fig. 7(b). In general, the running time increases as the graph becomes denser, and iMB+ outperforms Enum+ over all settings. Moreover, the speedup of iMB+ over Enum+ increases as the graph becomes denser. Intuitively, the increasing degree of each vertex makes the pruning techniques employed in Enum+ perform worse and even fail.

**5 Related Work**

In the literature, there are two kinds of work related to the large maximal \(k\)-biplex enumeration problem.

**Bipartite graph.** Various cohesive structures in bipartite graph have received much interest in recent research such as \((\alpha, \beta)\)-core [16], bitruss [17], biclique [14] and \(k\)-biplex [4]. However, those studies on \((\alpha, \beta)\)-core and bitruss cannot be adopted to our problem, since their structure definitions are inherently different. Recent studies on maximal biclique and \(k\)-biplex enumeration problem employ binary partition or backtracking techniques, and they are not scalable due to massive redundant computations.

**Maximal biclique enumeration/ frequent itemset mining.** Our problem is a generalization of the maximal biclique enumeration [14] and closed frequent itemset mining problems [15]. Specifically, when \(k=0\), \(k\)-biplexes reduce to bicliques, which correspond to the closed frequent itemsets and the transactions that involve the itemsets. However, the unique challenge, induced by the fact that the number of \(k\)-biplexes exponentially increases as \(k\) grows, as shown in Fig. 5, incurs huge redundancies for large \(k\) during the
enumeration and makes the problem intrinsically harder to solve. Existing techniques, e.g., LCM [15] and iMBEA [14], cannot be directly adapted to handle the cases $k > 0$.

6 CONCLUSION

In this paper, we study the large maximal $k$-biplex enumeration problem which is fundamental to many bipartite graph applications. With an eye on improving scalability of large maximal $k$-biplex enumeration, we develop an efficient tree-based algorithm with two advanced strategies and powerful pruning techniques. Extensive experiments on both real and synthetic graphs demonstrate that our algorithms outperform existing methods by up to four orders of magnitude.

Acknowledgement This research is supported by the Nanyang Technological University Start-Up Grant from the College of Engineering under Grant M4082302 and by the MHRD (India) under the SPARC programme project #P620.

REFERENCES