Queries to the Author

When you submit your corrections, please either annotate the IEEE Proof PDF or send a list of corrections. Do not send new source files as we do not reconvert them at this production stage.

Authors: Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections. Your article has been peer reviewed, accepted as final, and sent in to IEEE. No text changes have been made to the main part of the article as dictated by the editorial level of service for your publication.

Per IEEE policy, one complimentary proof will be sent to only the Corresponding Author. The Corresponding Author is responsible for uploading one set of corrected proofs to the IEEE Author Gateway

Q1. Please provide the page range for Ref. [15].

Thanks for your carefully checking, the page range for Ref. [15] is pp. 1-11.

Q2. Please check the Fig. 1(b). It looks different with the one, model1.pdf , in the final materials provided by us. Some symbols and subscripts are missed, as indicated by the red lines in Fig. 1(b).

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 θ. 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 θ [4]. We call these k-biplexes *large 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. 1a, 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 collusive 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 θ, and are not effective for very low values of θ. 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 $\widehat{G}$ would have a density $d = (n^2 - n)/(2n^2 - n)$ nearly 50 percent. (2) The resultant k-plexes (and their corresponding k-biplexes) may not satisfy the size constraint (imposed by θ). 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 $\mathcal{B}(X, Y)$ is an induced bipartite subgraph of bipartite graph $G(L, R)$ such that*

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

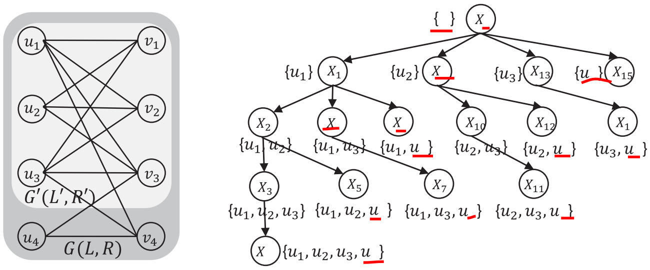
where $k \geq 0$ is a non-negative integer, $X \subseteq L$ and $Y \subseteq R$.

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 an important property of k-biplexes.

- *Kaiqiang Yu and Cheng Long are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore. E-mail: kaiqiang002@e.ntu.edu.sg, c.long@ntu.edu.sg.*
- *Deepak P is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN Belfast, U.K. E-mail: deepaksp@acm.org.*
- *Tanmoy Chakraborty is with the Department of Computer Science and Engineering, Indraprastha Institute of Information Technology, Delhi (IIIT-D), Delhi 110020, India. E-mail: tanmoy@iiitd.ac.in.*

Manuscript received 8 Apr. 2020; revised 14 Apr. 2021; accepted 26 Apr. 2021. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Cheng Long.)
Recommended for acceptance by Y. Zhang.
Digital Object Identifier no. 10.1109/TKDE.2021.3077071

Q2. Fig. 1(b) looks different with the one, model1.pdf , in the final materials provided by us. Some symbols and subscripts are missed, as indicated by the red lines in Fig. 1(b).



(a) Example graph. (b) Prefix tree over L.

Fig. 1. Examples to illustrate different strategies.

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 $|X| - \delta(v,X) \le k, \forall v \in Y'$. We can then imply $|X'| - \delta(v,X') \le |X'| + |X \setminus X'| - \delta(v,X') - \delta(v, X \setminus X') \le k$, since $|X \setminus X'| \ge \delta(v, X \setminus X')$. Similarly, we can obtain $|Y'| - \delta(u,Y') \le k, \forall u \in X'$. □

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

Definition 2 (Maximal k-biplex). *A k-biplex $\mathcal{B}(X,Y)$ of the bipartite graph G is maximal if and only if it is not contained by any k-biplex $\mathcal{B}'(X',Y')$ 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 θ, i.e., $|X| \ge \theta$, $|Y| \ge \theta$ and $\theta \ge 0$. We call these MBs *large MBs*. We formalize the problem exploited in this paper as follow [4].

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

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. θ is a $(\theta - k)$-core subgraph.*

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

For example, G' in Fig. 1a, 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* [4] 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. 1b shows the prefix tree constructed for L of graph $G(L,R)$ shown in Fig. 1a. All examples in this paper are executed based on a certain order of vertex index. W.l.o.g, we assume $L = \{u_1, \ldots, u_{|L|}\}$ and $R = \{v_1, \ldots, v_{|R|}\}$ follow the order of $u_1, u_2, \ldots, u_{|L|}$ and $v_1, v_2, \ldots, v_{|R|}$, respectively. With this order, the algorithm would process the nodes of the prefix tree in the order of $X_0, X_1, \ldots, X_{15}$ shown in Fig. 1b.

Specifically, the basic framework is shown in Algorithm 1 and executes as follows. For a given bipartite graph $G(L,R)$ shown in Fig. 1a 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 $\Phi(X) = \{v \in R | \delta(v,X) \ge |X| - k\}$ (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_i), $Y_i \subseteq \Phi(X_1)$ 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_q*(X_1) which is employed to avoid redundant computations (Line 11). Here, *cand_q*(X) is the set of vertices $w \in L$ that $X \cup \{w\}$ has been explored before. For example, we have *cand_q*$(X_6) = \{u_2\}$ in Fig. 1b, since $X_3 = X_6 \cup \{u_2\}$ has already been processed.

Algorithm 1. Algorithm Framework: `iMB-Basic`

Input: Bipartite graph $G(L,R,E)$, error tolerance k, minimum size θ, current node X, candidate set *cand_p*(X), exploited set *cand_q*(X), extension set $\Phi(X)$;
Output: All *MB*s $\mathcal{B}(X,Y)$ w.r.t. k and θ;
1: **for** $u \in$ *cand_p*(X) **do**
2: $X' \leftarrow X \cup \{u\}$, *cand_p*$(X) \leftarrow$ *cand_p*$(X) - \{u\}$;
3: $\Phi(X') \leftarrow \{v \in \Phi(X) | \delta(v,X') \ge |X'| - k\}$;
4: Apply the early stop strategy (Lemma 2);
5: Apply the advanced node expansion (Lemma 4);
6: **if** $|X'| \ge \theta$ **then**
7: List all *MB*s $\{(X',Y_i)\}, Y_i \subseteq \Phi(X')$ and $|Y_i| \ge \theta$;
8: Apply pruning strategies (Lemma 3 and Lemma 6);
9: **if** $|X'| + |$*cand_p*$(X')| \ge \theta$ *AND* $|\Phi(X')| \ge \theta$ **then**
10: *iMB-Basic*$(G, k, \theta, X',$ *cand_p*$(X'),$ *cand_q*$(X), \Phi(X'))$;
11: *cand_q*$(X) \leftarrow$ *cand_q*$(X) \cup u$;

Here, we elaborate implementation of the aforementioned procedure of listing *MB*s (Line 7). Basically, it can be done by enumerating all subsets Y_i of $\Phi(X')$ and checking whether (X', Y_i) is a *MB*. To speedup this procedure, we can maintain all subsets Y_i with a prefix tree w.r.t. a certain order, and then explore each subset Y_i in a depth-first way. According to the hereditary property, we can stop the extension and backtrack if the current subset Y_i cannot form a *MB* with X'. In addition, we can check the maximality for each subgraph $H(X', Y_i)$ by iteratively checking whether there exists a vertex u not in H such that $(X' \cup \{u\}, Y_i)$ or $(X', Y_i \cup \{u\})$ forms a *MB*.

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 *MB*s at the end as existing methods do [14].

Lemma 2. *Given a subset* $X \subseteq L$, *if there exists a vertex* $u \in$ cand_q(X) *that* $\Phi(X) \subseteq N_R(u)$, *then for any subset* $S \subseteq$ cand_p(X) *and* $Y \subseteq \Phi(X)$, $H(X \bigcup S, Y)$ *is not a* MB.

Proof. It can be easily proved by contradiction. Suppose that $(X \bigcup S, Y)$ is a *MB*. Based on $\Phi(X) \subseteq N_R(u)$ and $Y \subseteq \Phi(X)$, we imply $Y \subseteq N_R(u)$. Furthermore, we have $|Y| - \delta(u, Y) = 0 \leq k$ and $|X'| - \delta(v, X') \leq k, \forall v \in Y$, $X' = X \bigcup S \bigcup \{u\}$. Thus, we can form a larger k-biplex by adding u to $X \bigcup S$, 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 *Enum* would enumerate all those non-maximal k-biplexes.

Example 1. Suppose k = 1 and θ = 1 for Fig. 1a. We consider the current node X_9 of the prefix tree in Fig. 1b. Since X_1 and its subtrees have been processed, u_1 will be added to *cand_q*(X_9). We can stop at X_9 due to the fact that $\Phi(X_9) = \{v_i\}_{i=1}^{4} \subseteq N_Y(u_1) = \{v_i\}_{i=1}^{4}$. The next node X_{13} will then be processed and u_2 will be added to *cand_q*(X_{13}).

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 $\Gamma = \{Y_i \subseteq \Phi(X) | H(X, Y_i)$ is a *MB* w.r.t. subgraph $(X, \Phi(X))\}$ and a refined set $\widehat{\Gamma} = \{Y_i \in \Gamma | \forall u \in$*cand_q*$(X),\ Y_i \subsetneq N_R(u)\}$, where $X \subseteq L$ denotes a prefix tree node.

Lemma 3. *Let* $X \subseteq L$ *denote a prefix tree node. Then any* MB $\mathcal{B}(X \bigcup S, Y)$ *formed by subset* $S \subseteq$ cand_p(X) *and* $Y \subseteq \Phi(X)$ *does not include any vertex in the set* $T = \{v \in \Phi(X) | \forall Y_i \in \widehat{\Gamma}, v \notin Y_i\}$.

Proof. We prove it by contradiction. Suppose we have a *MB* $\mathcal{B}(X \bigcup S, Y)$ formed by subset $S \subseteq$ *cand_p*(X) and $Y \subseteq \Phi(X)$ that contains nonempty vertex set $T' \subseteq T$. Based on the property of k-biplexes, we can remove S from $\mathcal{B}(X \bigcup S, Y)$, and the resulting subgraph $H(X, Y)$ is still a k-biplex. Then we can imply $Y \subseteq Y_i$, where $Y_i \in \Gamma$. Since $T' \subseteq Y$, we can obtain $Y \notin \widehat{\Gamma}$. Otherwise, T' cannot be included in T due to the fact that $\exists Y \in \widehat{\Gamma}$, $T' \subseteq Y$. Therefore, Y must be refined from $\Gamma \backslash \widehat{\Gamma}$, and we thus imply $\exists u \in$ *cand_q*(X), $Y \subseteq N_R(u)$. We can obtain that $(X \bigcup S, Y)$ is not a *MB*, which leads to a contradiction. □

Since (X, Y_i), $Y_i \in \Gamma \setminus \widehat{\Gamma}$, is non-maximal, we can obtain $\widehat{\Gamma}$ and T when checking the maximality of the left side X which needs to check whether vertices in *cand_q*(X), cover Y_i. 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 that it satisfies the early stop conditions. For example, after removing T from $\Phi(X)$, we may have $|\Phi(X)| < \theta$ or $\Phi(X) \subseteq N_R(u)$, which leads to a pruning operation for X. We use the following example to illustrate the strategy.

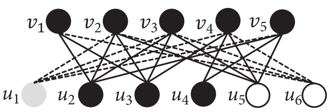


Fig. 2. Pruning techniques illustrated (vertices in current $X \cup \Phi(X)$, *cand_p*(X) and *cand_q*(X) are black, white and gray, respectively).

Example 2. Suppose $k = 1$ and $\theta = 3$ in Fig. 2. We consider current node $X = \{u_2, u_3, u_4\}$ and the corresponding vertex set *cand_q*$(X) = \{u_1\}$. We cannot stop the expansion of this node, since the early stopping condition is not satisfied because $v_1 \in \Phi(X)$ but $v_1 \notin N_Y(u_1)$. We can then obtain $\Gamma = \{\{v_1, v_4, v_5\}, \{v_2, v_4, v_5\}, \{v_3, v_4, v_5\}\}$ by enumerating over $\Phi(X)$. Since $N_{\Phi(X)}(u_1) = \{v_2, v_3, v_4, v_5\}$, Γ can be further refined as $\widehat{\Gamma} = \{\{v_1, v_4, v_5\}\}$. According to Lemma 3, we can obtain the refined $\Phi(X) = \{v_1, v_4, v_5\}$ by removing the vertex set $T = \{v_2, v_3\}$.

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 *MB*s sooner.

Lemma 4. *Given a subset* $X \subseteq L$, *if there exists a vertex* $u \in$ cand_p(X) *that* $\Phi(X) \subseteq N_R(u)$, *then for any k-biplex* $H(X \bigcup S, Y)$ *formed by* $S \subseteq$ cand_p(X) *and* $Y \subseteq \Phi(X)$, *we have that* $H(X \bigcup S \bigcup \{u\}, Y)$ *is also a k-biplex.*

Proof. It could be proved similarly as Lemma 2. □

Lemma 4 enables it to add such vertices $u \in$ *cand_p*(X) that $\Phi(X) \subseteq N_R(u)$ to X, since all *MB*s 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 *cand_p*(X). In fact, it would be enough to only require the vertex u to cover all vertices of Y_i in $\widehat{\Gamma}$ instead of Γ. This is because all k-biplexes $H(X, Y_i)$ formed by subsets $Y_i \in \Gamma \setminus \widehat{\Gamma}$ are contained in previous *MB*s and will lead to a duplication no matter whether $N_R(u)$ include such subsets or not.

Example 3. Suppose $k = 1$ and $\theta = 1$ for Fig. 1a. We consider current node $X_1 = \{u_1\}$ with $\Phi(X_1) = \{v_1, v_2, v_3, v_4\}$ and *cand_p*$(X_1) = \{u_2, u_3, u_4\}$. Based on Lemma 4, we can remove u_3 from *cand_p*(X_1) and add it to X_1 such that $X_1 = \{u_1, u_3\}$. Note that the next node is $X_3 = \{u_1, u_3, u_2\}$ instead of $X_2 = \{u_1, u_2\}$. Moreover, X_2, X_5, X_6 and X_7 would be pruned since u_3 has been removed from *cand_p*(X_1).

To further prune unnecessary branches, we propose to refine *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 *cand_p*(X) that cannot form any *MB* with X. We then develop a strong condition for *cand_p*(X) based on all subsets $Y_i \in \widehat{\Gamma}$.

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

Proof. We prove it by contradiction. Suppose there exists a *MB* $\mathcal{B}(X \bigcup S, Y)$ formed by $S \subseteq cand\_p(X)$ 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 $Y \subseteq Y_i$, $\exists Y_i \in \Gamma$. Two different cases for Y need to be further exploited as follows.

i) $Y_i \in \Gamma \setminus \widehat{\Gamma}$**:** In this case, there exists a vertex $w \in cand\_q(X)$ that $Y \subseteq N_R(w)$. Therefore, a larger *MB* $\mathcal{B}'(X \bigcup S \bigcup \{w\}, Y)$ can be formed, which leads to a contradiction to the assumption.

ii) $Y_i \in \widehat{\Gamma}$**:** In this case, we can obtain a larger k-biplex $H'(X \bigcup \{u\}, Y)$ by removing $S \setminus \{u\}$ from the *MB* $\mathcal{B}$, which contradicts to the condition. We thus finish the proof. □

Lemma 5 indicates that we can remove a vertex u from the candidate set $cand\_p(X)$, if u satisfies some conditions. Intuitively, it turns to check all $Y_i \in \widehat{\Gamma}$ 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. θ, a vertex u can be added to X to form a large k-biplex $H(X \bigcup \{u\}, Y')$ with $Y' \subseteq Y$ (condition ¬A) if and only if $\delta(u, Y) \geq \theta - k$ and $|\Phi(X \bigcup \{u\})| \geq \theta$.*

Proof. We prove it by two directions, and start with "$\Longrightarrow$". If $H(X \bigcup \{u\}, Y')$ with $Y' \subseteq Y$ is a k-biplex, we can obviously obtain $|\Phi(X \bigcup \{u\})| \geq |Y'| \geq \theta$ and $\delta(u, Y) \geq \delta(u, Y') \geq |Y'| - k \geq \theta - k$.

For the other direction "$\Longleftarrow$", we consider two different cases.

i) $\delta(u, Y) \geq \theta$**:** In this case, we prove it by showing that $(X \bigcup \{u\}, N_Y(u))$ 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 $(X \bigcup \{u\}, N_Y(u))$, since $\delta(u, Y) - \delta(u, Y) = 0$ and $\forall v \in N_Y(u)$, $|X \bigcup \{u\}| - \delta(v, X \bigcup \{u\}) = |X| - \delta(v, X) \leq k$.

ii) $\theta - k \leq \delta(u, Y) < \theta$**:** In this case, we try to construct a k-biplex whose left side is $X \bigcup \{u\}$. We can always find a subset Ψ of $\Phi(X \bigcup \{u\}) \setminus N_Y(u)$ with size $|\Psi| = \theta - \delta(u, Y) \leq k$, since $\Phi(X \bigcup \{u\}) \geq \theta$ and $\theta - k \leq \delta(u, Y) < \theta$. We will show that $(X \bigcup \{u\}, N_Y(u) \bigcup \Psi)$ is a k-biplex. Since $H(X, Y)$ is a k-biplex, we can deduce that $(X, N_Y(u) \bigcup \Psi)$ is also a k-biplex w.r.t. θ, since $|N_Y(u) \bigcup \Psi| = \theta$. To show $(X \bigcup \{u\}, N_Y(u) \bigcup \Psi)$ is a k-biplex, we need to show that u can be added to $(X, N_Y(u) \bigcup \Psi)$. This can be easily verified due to the fact $|N_Y(u) \bigcup \Psi| - \delta(u, N_Y(u) \bigcup \Psi) = |\Psi| \leq k$. □

Combining the above two lemmas, we can efficiently refine $cand\_p(X)$. To be specific, we can remove u from $cand\_p(X)$, if $\forall Y_i \in \widehat{\Gamma}$, $\delta(u, Y_i) < \theta - k$ or $|\Phi(X \bigcup \{u\})| < \theta$. We note that *Enum* also adopts some refining strategies, and all of them depend on $\Phi(X)$, i.e., $\delta(u, \Phi(X)) \leq \theta - k$, whereas our techniques are based on more refined ones, providing stronger pruning power.

Example 4. Suppose $k = 1$ and $\theta = 3$ in Fig. 2. We still consider the current node $X = \{u_2, u_3, u_4\}$ and the corresponding set $cand\_q(X) = \{u_1\}$. Based on the early stopping condition, we can obtain $cand\_p(X) = \{u_5, u_6\}$. As mentioned before, we have $\Gamma = \{\{v_1, v_4, v_5\}, \{v_2, v_4, v_5\}, \{v_3, v_4, v_5\}\}$ and $\widehat{\Gamma} = \{Y_1 = \{v_1, v_4, v_5\}\}$. Based on Lemmas 4 and 6, we can prune u_5 and u_6 due to $|\delta(u_5, Y_1)| = |\{v_4, v_5\}| < 3$ and $|\delta(u_6, Y_1)| = |\{v_1, v_4\}| < 3$ respectively. However, we cannot prune them only based on $\Phi(X)$, since $|\Phi(X \cup \{u_5\})| = |\{v_i\}_{i=2}^{5}| > 3$ and $|\Phi(X \cup \{u_6\})| = |\{v_i\}_{i=1}^{4}| > 3$.

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 θ, and lists all large *MB*s, i.e., maximal k-biplexes with the size at each side at least θ. *D2K* accepts two parameters k' and θ' and lists all k'-plexes with the size at least θ'. For *D2K*, we set $k' = k+1$ and $\theta' = 2\theta$ and apply a post-processing step to filter out those *MB*s that are not large (i.e., with the size at one side smaller than θ). 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 2.66 GHz processor and 32 GB of memory. The source codes are available at https://personal.ntu.edu.sg/c.long/code/iMB.zip.

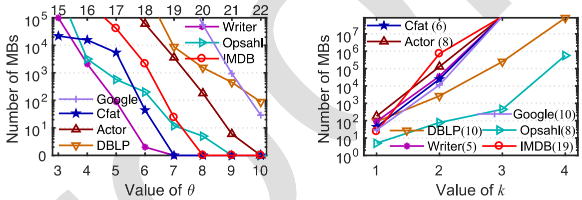
# of MBs. We show the results of the effect of varying θ and k on the number of large *MB*s in Fig. 3. The results that are out of the y-range are not shown. Generally, the number of *MB*s exponentially decreases or increases as θ or k increases, respectively. Note that we use the indexes located at the top of Fig. 3a as the x-axis for IMDB. Based on these results, we can determine the meaningful settings of θ and k for different datasets. We set θ 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. 5a and 5b, 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 θ, although they are finally discarded.

Performance on Biclique Setting ($k = 0$). Since 0-biplexes are bicliques, we compare with maximal biclique enumeration algorithms, *LCM* [15] 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

TABLE 1
Datasets Used in Our Experiments

Name	Category	Vertices	Edges
Cfat	Miscellaneous	400	802
Opsahl	Authorship	38,741	58,595
Writer	Affiliation	135,569	144,340
Actors	Affiliation	520,223	1,470,404
IMDB	Communication	1,324,748	3,782,463
DBLP	Authorship	5,425,963	8,649,016
Google	Hyperlink	20,200,070	14,693,125

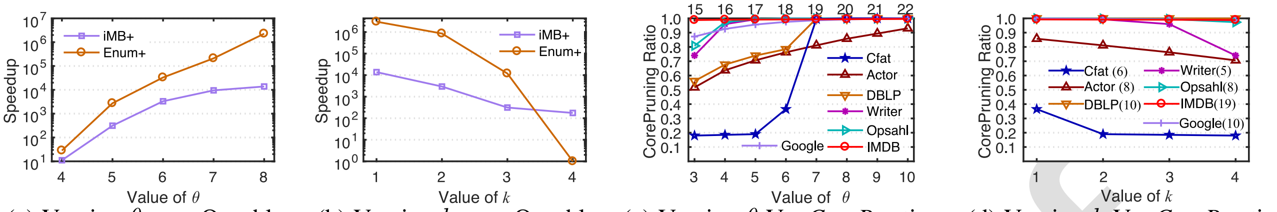


(a) Varying θ (b) Varying k

Fig. 3. Effect of θ and k on the number of *MB*s (the numbers behind the dataset names correspond to the values of θ, e.g., "Cfat(6)").

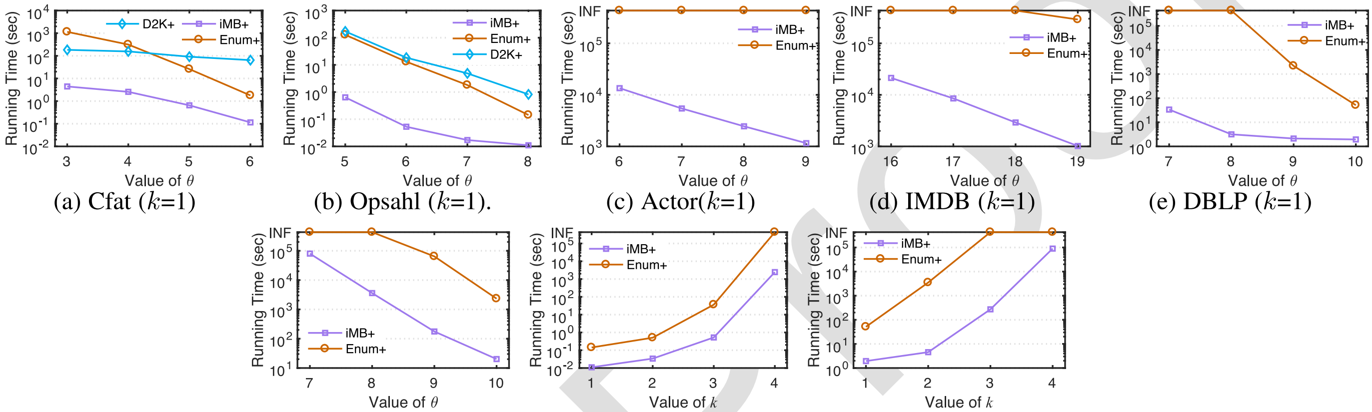
TABLE 2
Running Time on Biclique Setting With k=0 and θ=1 (sec)

Algo.	Cfat	Opsahl	Writer	Actors	IMDB	DBLP
iMB	**0.001**	**0.078**	**0.787**	35.06	**406.06**	**956.12**
LCM	0.001	0.292	1.244	**26.57**	956.16	23882
iMBEA	0.004	8.212	82.97	435.82	2765.7	6543.8



(a) Varying θ over Opsahl (b) Varying k over Opsahl (c) Varying θ Vs. *CoreP.* ratio. (d) Varying k Vs. *CoreP.* ratio

Fig. 4. *Corepruning* performance over value of θ and k (the numbers behind the dataset names correspond to the values of θ, e.g., "Cfat(6)").



(a) Cfat (k=1) (b) Opsahl (k=1). (c) Actor(k=1) (d) IMDB (k=1) (e) DBLP (k=1)

(f) Google (k=1) (g) Opsahl (θ=8) (h) DBLP (θ=10)

Fig. 5. Efficiency evaluation over value of θ and k.

results verify the effectiveness of the pruning techniques that are newly introduced in this paper.

4.1 Evaluation on Real Datasets

We now compare our algorithms *iMB* and *iMB+* with existing methods on real datasets, and study the efficiency by considering the effect of various values of θ and k. We denote the running time of the algorithm as infinite (INF) if it cannot finish in 120 hours.

Effect of CorePruning. We first analyze the performance of *CorePruning* by varying θ and k in Fig. 4. We measure the speedups of *iMB+* (*Enum+*) over *iMB* (*Enum*). Generally, *iMB+* and *Enum+* are significantly faster than *iMB* and *Enum*, respectively, thus showing the necessity of *CorePruning*. The speedups of two algorithms are obviously not directly comparable, given that they are based on different base algorithms. Note that for the setting $k = 4$ in Fig. 4b, both *Enum* and *Enum+* cannot finish in limited time. Moreover, the *CorePruning* ratios (defined as the number of pruned vertices/total number of vertices) are more than 0.5 for most datasets and settings, which further demonstrates the effectiveness of *CorePruning*. For Cfat, since the degrees of vertices are almost the same, the k_{max}-core includes nearly 62.5 percent of vertices in the graph. Note we use k_{max} to denote the maximum core number such that $(k_{max}+1)$-core is empty. This leads to lower performance of *CorePruning*. Conversely, IMDB, with a small dense k_{max}-core, always achieves high *CorePruning* ratios.

Effect of Varying θ. We study the impact of θ on the running time, number of *MBs* and proposed pruning techniques, and report results in Figs. 5c, 5d, 5e, and 5f. We first observe that our algorithm *iMB+* significantly outperforms existing methods by up to four orders of magnitude. Moreover, the running times of *iMB+* and *Enum+* significantly decrease as θ increases. It is easy to understand based on the following two observations. First, the number of *MBs* (shown in Fig. 3a) exponentially decreases as θ increases, thereby reducing the complexity. Second, some pruning techniques are derived based on θ. For example, the ratio of vertices pruned by *CorePruning* (shown in Fig. 4c) increases as θ grows, which leads to better performance for larger θ.

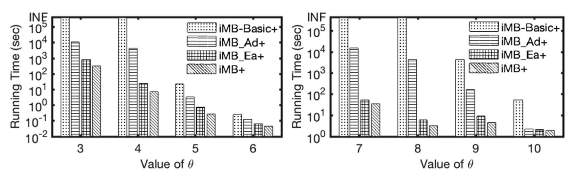
Effect of Varying k. We then investigate the effect of varying k in Figs. 5g and 5h. In general, it is evident that our algorithm outperforms existing methods. To be specific, the running time increases with the increase of k, which could be explained by the results from Figs. 3b and 4d, i.e., the number of *MBs* exponentially increases as k increases, which incurs more computations. The *CorePruning* ratio decreases as k grows.

Effect of Different Strategies. We evaluate the impact of various strategies of the proposed algorithm in Fig. 6. Both *iMB_Ad+* (only with advanced node expansion) and *iMB_Ea+* (only with early stopping strategy) perform better than *iMB-Basic+* (Algorithm 1), which further demonstrates two different strategies and their pruning techniques can improve efficiency. Moreover, *iMB+*, which combines both strategies, performs the best.

4.2 Evaluation on Synthetic Datasets

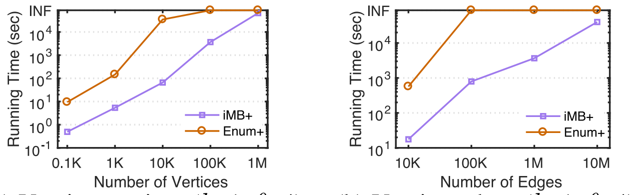
We evaluate the scalability of *iMB+* on synthetic graphs of varying numbers of vertices and densities. We report the running time of returning the first 1000 large *MBs* and denote INF as 24 hours.

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. 7a 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



(a) Writer ($k=1$) (b) DBLP ($k=1$)

Fig. 6. Efficiency evaluation over different strategies.



(a) Varying vertices (k=1, θ=4) (b) Varying edges (k=1, θ=4)

Fig. 7. Running time on synthetic datasets.

we return a fixed number of *MB*s, the running time increases with the increase of θ. 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. 7b. 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 (α, β)-core [16], bitruss [17], biclique [14] and k-biplex [4]. However, those studies on (α, β)-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. 3, 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 exiting methods by up to four orders of magnitude.

Acknowledgments

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

References

[1] J. Wang, A. P. De Vries, and M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proc. 29th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 501–508.

[2] M. Ley, "The DBLP computer science bibliography: Evolution, research issues, perspectives," in *Proc. Int. Symp. String Process. Inf. Retrieval* 2002, pp. 1–10.

[3] M. Kaytoue, S. O. Kuznetsov, A. Napoli, and S. Duplessis, "Mining gene expression data with pattern structures in formal concept analysis," *Inf. Sci.*, vol. 181, no. 10, pp. 1989–2001, 2011.

[4] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu, "Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market and protein networks," *Statist. Anal. Data Mining*, vol. 2, no. 4, pp. 255–273, 2009.

[5] L. Wang, "Near optimal solutions for maximum quasi-bicliques," *J. Combinatorial. Optim.*, vol. 25, no. 3, pp. 481–497, 2013.

[6] T. A. Oliveira *et al.*, "A VNS approach for book marketing campaigns generated with quasi-bicliques probabilities," *Electron. Notes Discrete Math.*, vol. 58, pp. 15–22, 2017.

[7] H. S. Dutta and T. Chakraborty, "Blackmarket-driven collusion among retweeters-analysis, detection, and characterization," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1935–1944, 2020.

[8] S. Dhawan, S. C. R. Gangireddy, S. Kumar, and T. Chakraborty, "Spotting collective behaviour of online frauds in customer reviews," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 245–251.

[9] A. K. Poernomo and V. Gopalkrishnan, "Towards efficient mining of proportional fault-tolerant frequent itemsets," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2009, pp. 697–706.

[10] X. Liu, J. Li, and L. Wang, "Modeling protein interacting groups by quasi-bicliques: Complexity, algorithm, and application," *IEEE/ACM Trans. Comput. Bio. Bioinform.*, vol. 7, no. 2, pp. 354–364, Apr.–Jun. 2010.

[11] F. Ruskey, "Combinatorial generation," Univ. Victoria, Victoria, BC, Canada, Woking Ver. 1j-CSC 425/520, 2003

[12] D. Berlowitz, S. Cohen, and B. Kimelfeld, "Efficient enumeration of maximal k-plexes," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 431–444.

[13] A. Conte, T. D. Matteis, D. D. Sensi, R. Grossi, A. Marino, and L. Versari, "D2K: Scalable community detection in massive networks via small-diameter k-plexes," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1272–1281.

[14] Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, and M. A. Langston, "On finding bicliques in bipartite graphs: A novel algorithm and its application to the integration of diverse biological data types," *BMC Bioinform.*, vol. 15, 2014, Art. no. 110.

[15] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations*, 2004. Q1

[16] B. Liu, L. Yuan, X. Lin, L. Qin, W. Zhang, and J. Zhou, "Efficient (α, β)-core computation: An index-based approach," in *Proc. World Wide Web Conf.*, 2019, pp. 1130–1141.

[17] K. Wang, X. Lin, L. Qin, W. Zhang, and Y. Zhang, "Efficient bitruss decomposition for large-scale bipartite graphs," in *Proc. 36th IEEE Int. Conf. Data Eng.*, 2020, pp. 661–672.