Pivot-based Maximal Biclique Enumeration

Aman Abidi , **Rui Zhou** , **Lu Chen** and **Chengfei Liu**
Swinburne University of Technology, Melbourne, Australia
{aabidi, rzhou, luchen, cliu}@swin.edu.au

Abstract

Enumerating maximal bicliques in a bipartite graph is an important problem in data mining, with many applications across different domains such as web community, bioinformatics, etc. Although research has been conducted on this problem, surprisingly, we find that pivot-based search space pruning, which is effective in clique enumeration, has not been exploited in biclique scenario. Therefore, in this paper, we explore the pivot pruning for biclique enumeration. We propose an algorithm for implementing the pivot pruning, powered by an effective index structure Containment Directed Acyclic Graph ($CDAG$). Meanwhile, existing literature indicates contradictory findings on the order of vertex selection in biclique enumeration. As such, we re-examine the problem and suggest an offline ordering of vertices that expedite the pruning. We conduct an extensive performance study using real world datasets from a wide range of domains. The experimental results demonstrate that our algorithm is more scalable and outperforms all the existing algorithms across all datasets and can achieve a significant speedup against the previous algorithms.

1 Introduction

A Bipartite graph is an interesting structure that can be used to represent two disjoint sets of vertices. Let us consider a bipartite graph $G = (U \cup V, E)$, where U and V are two disjoint sets, and $E \subseteq U \times V$. A biclique $B = (X \cup Y)$, $X \subseteq U$, $Y \subseteq V$ is a complete subgraph of a given bipartite graph G s.t. all the vertices of X are connected to all the vertices of Y. The maximal biclique is the one, which is not a proper subgraph of any other biclique in G. Figure 1 shows the bipartite graph and a maximal biclique.

Consider an example in a *Youtube* network containing a relationship between users and groups. A biclique in this scenario consists of a set of users and the set of groups that contain all the corresponding users. Exploring such biclique of users who share a common interest can be proved valuable for analyzing the behaviour and speculate the actions of a user in the network. Generally, identifying *maximal bicliques* proves to be useful, as they are not contained by any other bicliques.

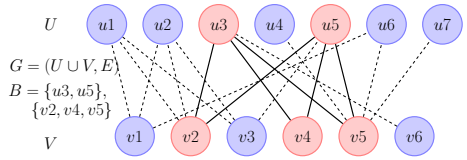


Figure 1: Maximal biclique(B) in a bipartite graph(G).

Therefore, we study the problem of *Maximal biclique enumeration* {*MBE*} in a bipartite graph.

The domain of real world applications of MBE includes community detection, bioinformatics, closed item sets, etc. Some typical applications include detecting overlapping communities in *Noordin Top Terrorist Network* for finding more dangerous elements [Alzahrani and Horadam, 2019], discovering large dense graphs in massive graphs for spam group detection [Gibson *et al.*, 2005], generating probabilistic model for protein-protein interaction network analysis [Schweiger *et al.*, 2011], extracting gene-phenotype information from transactional databases [Xiang *et al.*, 2011], constructing the optimal phylogenetic tree [Driskell *et al.*, 2004] and gene expression [Kaytoue *et al.*, 2011]. From the domain of closed item sets, MBE is used in association rule mining from transactional databases [Vanahalli and Patil, 2019]. Some of the other applications include reducing the role mining complexity in role-based access control system [Wu *et al.*, 2018], learning context-free grammars [Yoshinaka, 2011] and providing a model with constraints for solving chemical process scheduling problems [Mouret *et al.*, 2011]. However, currently, there are not many algorithms which address the problem of enumerating all maximal bicliques in a bipartite graph. The previous algorithms were unable to tackle some challenges associated with the biclique enumeration, which are: (i) *Inefficient pruning technique for the large search space* - the state-of-the-art algorithms iMBEA [Zhang *et al.*, 2014] and LCM-MBC [Li *et al.*, 2007] have deployed pruning techniques to reduce the search space to some extent. However, the resulting search space is still large and exacerbates the cost of enumeration. (ii) *Inefficient ordering of vertices* - the ordering of vertices plays a crucial role in the enumeration of search space [Zhang *et al.*, 2014; Damaschke, 2014]. Although, the ordering proposed improved the overall enumeration time, yet an extra overhead of local ordering was introduced.

In this paper, we aim to address the above two challenges.

We propose **Pivot-based Maximal Biclique Enumeration** (PMBE) algorithm for enumerating all maximal bicliques. PMBE integrates the *pivot pruning* technique with LCM-MBC for reducing the search space. An efficient implementation of the pivot pruning is guaranteed by our proposed *Containment Directed Acyclic Graph (CDAG)* index structure. The semantics of ordering of vertices to enumerate maximal biclique is also studied, and a heuristic ordering is proposed for the heuristic pivot selection, which eventually optimizes the algorithm. Our major contributions are listed as below:

- *Optimized pivot-based pruning algorithm for enumerating all maximal bicliques.* We present an optimized algorithm PMBE, which introduces the classical pivot technique for expediting the search space pruning, for the enumeration of all maximal bicliques.
- *Efficient index structure for containment relationship.* We utilize an efficient offline structure called $CDAG$, for systematizing pivot pruning.
- *Rev-Topological order for heuristic pivot selection.* We study semantics and analyze the ordering of vertices in biclique generation, and suggest an offline ordering *rev-topological* for efficient pivot pruning.
- *Extensive performance studies across real world networks.* We conduct extensive experiments on 10 real datasets to evaluate the efficiency of our proposed algorithms.

2 Preliminaries and Problem Definition

Let $G = (U \cup V, E)$ be a bipartite graph, where U and V are two disjoint sets of vertices, and E is an edge set. A biclique is a complete subgraph of the given graph G, which is formalized as:

Definition 1. *Bipartite Complete Graph or Biclique: A biclique within G is a couple (set pair)* (X, Y) *s.t.* $X \subseteq U, Y \subseteq V$ *and* $\forall u \in X, v \in Y$, $(u, v) \in E$.

A biclique is a *maximal biclique* if it cannot be contained by any other biclique.

Problem definition. Given a bipartite graph $G = (U \cup V, E)$ with no multiple edges and no self-loops. We enumerate all the maximal bicliques in G.

3 Methodology

This section explains how our approach works to tackle the challenges discussed in Section 1. The *pivot pruning* approach has been first introduced by Bronn and Kerbosh [Bron and Kerbosch, 1973] for finding cliques. It has proven to be a simple, yet effective technique in pruning cliques. Surprisingly, the technique has not been utilized, in the field of bicliques. We now present a modified pivot pruning supported by ordering for an optimized enumeration of maximal bicliques. We observed that the *pivot pruning* effectiveness depends on the pivot selection. Therefore, an effective *rev-topological* order is proposed for the same. The proposed modification is achieved in the following steps. *Firstly*, the framework for our algorithm is discussed which uses two techniques namely *pivot pruning* and *rev-topological* ordering for optimization. Secondly, *pivot pruning* is explained using the containment relationship (Definition 2). *Thirdly*, we propose the index $CDAG$, for the effective implementation of pivot pruning. *Fourthly*, we study the semantics of the ordering of vertices and suggest a *rev-topological* order for heuristic pivot selection. *Lastly*, the algorithm PMBE is discussed which enumerates all the maximal bicliques.

Algorithm 1: Enumerate all maximal bicliques

```
   Input  : G = (U ∪ V, E)
   Output: B is the maximal Biclique
1  Function EnumerateMaximalBiclique(U,V)
2      *Create a CDAG
3      *cand = Rev-Topological(V)
4      B = ϕ
5      PMBE(B, U, cand)
6  Function PMBE (B,Y,cand)
7      if cand.isEmpty then
8          return
9      *Pivot Selection
10     foreach v ∈ cand do
11         if rangefinder(p,v) ≠ TRUE then
12             B = Γ(Γ(X ∪ {v}))
13             if B \ (X ∪ {v}) ⊆ cand then
14                 Output (B, Γ(X ∪ {v}))
15                 if (cand \ B) isnotEmpty then
16                     PMBE(B, Γ(X ∪ {v}), cand \ B)
```

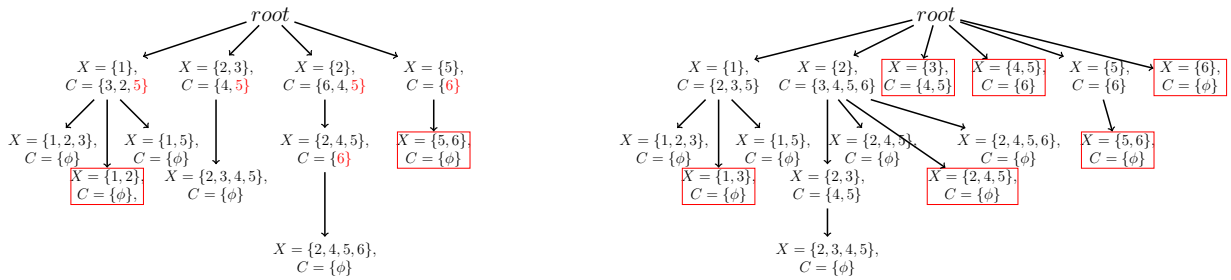
3.1 The Framework

The proposed framework of our approach can be observed in Algorithm 1. The framework uses the techniques of pruning and ordering to optimize the algorithm. We explain the two aspects of the framework, which are search space and enumeration. Some properties of enumeration are adapted from LCM-MBC [Li *et al.*, 2007].

Search Space. For a bipartite graph $G = (U \cup V, E)$, the search space is restricted to one of the vertex sets (U, V), as it can be used to determine the corresponding adjacency list. The search space of MBE is a set enumeration tree for a vertex set, which systematically enumerates the power-sets of the corresponding vertex set using a pre-imposed order. For the set enumeration, a smaller vertex set, say V, is selected. Each of the nodes in the search space represents a possible candidate for a maximal biclique. The root represents an empty set. The *sub search space (sub(X))* for a vertex set X is a subtree that is rooted at the vertex set X. Given a vertex set X, $cand(X)$ denotes the candidates for node X in the search space, containing only the vertex which comes later than the last vertex set in X.

Before introducing enumeration, we first define some notations to simplify the explanation. The common neighborhood for a vertex set X is denoted by $\Gamma(X)$. For a given vertex set $X \in P$ (P is the power-set of V), $\Gamma(X) = \{u | u \in U$ and $\forall v \in X, (u, v) \in E\}$.

The Enumeration. It is accomplished by using two functions as shown in Algorithm 1: (i) *EnumerateMaximalBiclique* and (ii) *PMBE*. EnumerateMaximalBiclique is responsible for performing the *rev-topological* order for an efficient pivot selection (line 3). *PMBE* which has been inspired by



a: PMBE, Pivot is represented by red vertices

b: LCM-MBC

Figure 2: Enumeration tree for different algorithms. C is the cand at each level and red boxes are the duplicate bicliques

LCM-MBC for bipartite graphs, implements the pivot selection (line 9) and performs the *pivot pruning* (line 11) to enumerate all maximal bicliques. The framework includes creating the structure $CDAG$ (line 2) for supporting *pivot pruning* and *rev-topological* ordering. Apart from the pruning, PMBE ensures that a biclique generated is maximal iff $\Gamma(\Gamma(X)) = X$ (line 12) [Li *et al.*, 2007]. The duplicate bicliques are pruned using line 13 [Li *et al.*, 2007]. The resulting optimized algorithm LCM-MBC enumerates the search space shown in Figure 2b. LCM-MBC runs with an exponential complexity of $O(|U||V|\beta)$, where β = number of maximal bicliques. However, it still suffers from the following two problems: firstly, the search space is still very large; secondly, there are numerous duplicate maximal bicliques generated (red boxes).

3.2 Pivot

The pivot technique for maximal clique enumeration is to prune search branches that cannot result in a maximal clique as early as possible. The pivoting consists of the following: instead of iterating over each vertex and checking for maximal cliques, choose a pivot. The results will have to contain either the pivot or one of its non-neighbors, since if none of the non-neighbors of the pivot is included, then we can add the pivot itself to the result. Hence, only the pivot and its non-neighbors need to be tested in the enumeration space. The properties of cliques and bicliques are distinct, hence selecting a pivot with the same notion will not be effective in the case of bicliques. Consequently, the pivot selection approach has to be modified for bicliques.

The motivation behind pivot pruning is to reduce the duplicate biclique enumeration. In our context, a pivot is a vertex selected among the candidates, which can be used to reduce or prune the enumeration space of our problem without losing any results. To define pivot formally, we first define the containment relationship as:

Definition 2. *Containment ($\subset$): For a given bipartite graph G=(U $\cup$ V, E), let $v_1, v_2 \in$ V. Then, v_1 is contained by v_2 iff $\Gamma(\{v_1\})$ is a subset of $\Gamma(\{v_2\})$.*

The containment relationship provides an elegant property which can be utilized as a criterion for pivot selection.

Property 1. *Given $v_1, v_2 \in cand(X)$, if $v_1 \subset v_2$, then sub(v_2) contains sub(v_1).*

Proof. Given $v_1 \subset v_2$, we have $\Gamma(\{v_1\}) \subseteq \Gamma(\{v_2\})$, which in turn implies $\Gamma(\Gamma(\{v_1\})) \subseteq \Gamma(\Gamma(\{v_2\}))$. Here $sub(v_1)$ and $sub(v_2)$ are the power sets of $\Gamma(\Gamma(\{v_1\}))$ and $\Gamma(\Gamma(\{v_2\}))$ respectively. Hence we say $sub(v_2)$ *contains* $sub(v_1)$. □

Therefore, using Property 1, we define the pivot as:

Definition 3. *Pivot: Given a vertex set X, a vertex $p \in cand(X)$ is a pivot if $\exists\, v \in cand(X)$, s.t. $\Gamma(\{v\}) \subset \Gamma(\{p\})$.*

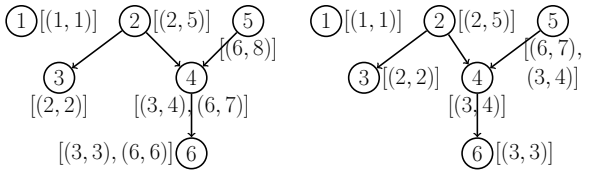
In the search space, at each level of recursion, i.e., vertex set X, we intend to prune away all the contained vertices for a corresponding selected pivot. Further, we propose the pivot pruning as follows:

Proposition 1. *At each recursion level in search space, selecting one pivot $p \in cand(X)$ and pruning all vertices contained by p from cand(X) shall not lose any maximal biclique.*

According to Property 1, the pivot sub search space already contains sub search space of the pruned vertices. Therefore, removing the contained vertices would reduce the overhead of checking the branches that cannot result in maximal bicliques, which would not affect the results. The example of pivot pruning can be seen by comparing Figures 2b and 2a, at level 1 vertices 4 and 6 are contained by pivot 5. The bicliques enumerated (Figure 2b) from vertices 4 and 6 are redundant and an unnecessary extra cost is utilized for handling them. Once the pivot criterion is decided, we now apply the pivot pruning. The naive method includes selecting each neighbor of a vertex and finding it in the neighborhood of the pivot, is expensive and inefficient. Therefore, to address the problem of pivot pruning efficiently, we propose an index structure *Containment Directed Acyclic Graph ($CDAG$).*

3.3 Containment Directed Acyclic Graph

The containment directed acyclic graph ($CDAG$) is an offline structure used to store the containment relationship (Definition 2). The challenge in implementing the pivot pruning is proportional to answering the query of whether the vertex v_2 contains the vertex v_1. $CDAG$ addresses this problem and provides the acknowledgment for the query in almost constant time. It is composed of two components: directed acyclic graph (DAG) and range index (R). A DAG is created for a $cand$ where the directed edges represent the containment relationship. The vertex v_1 is said to be a *child* of v_2 if there exists a direct edge from v_2 to v_1. Although, a DAG



a: Inefficient range index b: Efficient range index

Figure 3: Containment directed acylic graph($CDAG$)

is sufficient for finding the containment relationship, we improve the structure by integrating the range index.

Definition 4. *The range index ($R_v = [x, y]$, where $x, y \in Integer$) for a vertex $v \in cand$, is an interval assigned to the corresponding vertex of a DAG during the traversal. x and y denote the beginning and the termination of the traversal.*

The range index is assigned to each vertex in the DAG and can be utilized to find the containment relationship quickly. The range indexing is performed using a depth first search approach (DFS). We begin the traversal of the DAG with each root vertex and assign the range index to the vertices in a $CDAG$. For an efficient and non-redundant range indexing, the transitive edges from the DAG are removed. The vertices which are not contained by any other vertex are considered to be the root vertex, e.g., vertices $1, 2$ and 5 in Figure 3a. However, if a vertex is contained by more than one vertex, it can have more than one range index, e.g., vertices 4 and 6 in Figure 3a. This unnecessary overhead of range indexes on vertices 4 and 6 can be reduced by adopting a technique from [Agrawal *et al.*, 1989]. We create two types of range index for each vertex in a DAG, first, its own ($rangeindex$) and second, obtained from the common children ($child\_rangeindex$), which has been already visited. The fundamental approach of creating a $CDAG$ is to first create a DAG and then implement the range indexing. For a given graph $cand$ a $CDAG$ can be defined as:

Definition 5. *A $CDAG$, $C = (\bar{V}, \bar{E})$ is an acyclic graph, s.t. $\forall \bar{v} \in \bar{V}, \bar{v}$ is composed of $v \in cand$ and R_v, and $\forall(\bar{v}, \bar{u}) \in \bar{E}$, $\bar{u} \subset \bar{v}$.*

Figure 3b is the required $CDAG$ for the $cand$. There are some common terminologies regarding a $CDAG$. A **path** is defined as a sequence of edges from one vertex to the other in a $CDAG$. A vertex v of a $CDAG$ is said to be **reachable** from another vertex u, if there exists a path starting from u and ending at v. Hence, reachability in a $CDAG$ implies the containment relationship in our problem. Therefore, we conclude the following proposition.

Proposition 2. *v is said to be reachable from x in a CDAG, if range index of v lies within one of the range indexes of x.*

To answer the query of containment relationship between vertices we contrive a function *rangefinder* (line 11) in Algorithm 1. We exploit Proposition 2 to formulate *rangefinder*, i.e., comparing the range index of the vertices in a $CDAG$. The runtime complexity of the *rangefinder* is $O(|\mathrm{R_{pivot}}||\mathrm{R_v}|)$, where $\mathrm{R_{pivot}}$ is the number of range indexes of pivot and $\mathrm{R_v}$ is the number of range indexes of vertex v. Figure 2a displays at each recursive level in an enumeration tree a pivot is selected and results in better pruning of the enumeration space. However, the pruning power of a pivot depends on the selection of the pivot. If a pivot selected does not contain any vertex, then *rangefinder* will be overhead. Therefore, we select a pivot at each recursive level without any extra cost, s.t. it has the potential to contain more vertices. We now discuss the pivot selection technique.

3.4 Heuristic Pivot Selection

The pivot selection requires an approach to be fast yet effective at the same time. We propose a heuristic pivot selection, where we select a vertex from $cand(X)$ in constant time. To ensure the effectiveness of the pivot, we study the ordering of $cand(X)$. Ordering of a vertex set manifests a critical role in the pivot selection. The algorithms like MICA and LCM-MBC have been using the lexicographical sort. The current fastest algorithm for our problem iMBEA, suggests sorting the vertices according to the non-decreasing order of the number of common neighbors which incurs extra cost. The contradicting conclusion for the same problem on efficiency of discovering biclique is proposed in [Damaschke, 2014], suggesting the non-increasing order. From these contradicting techniques, we conclude that the better sorting technique being opted is closely related to the algorithm being implemented for the enumeration. We propose an offline ordering which saves the cost at each recursion level and also ensures the effectiveness of pivot selection at the same time. The order follows the property that if $v_2 \subset v_1$ then v_2 should occur before v_1. With this specific ordering, we can easily select the pivot from the $cand$, which ensures that all pruned vertices during the enumeration do not affect the results, as at the end, pivot enumerates their collective search space (Proposition 1). The $CDAG$ created stores the reachability information and can be utilized to propose the required order which is *rev-topological* order. The *rev-topological* order is one where no vertex in a $CDAG$ can occur before its reachable vertices. The *rev-topological* order is achieved by exploiting the already known algorithm for topological ordering.

Proposition 3. *The $cand(X)$ is ordered using rev-topological order offline using the $CDAG$ to increase pivot pruning power.*

Following the selection of the ordering of vertices, we utilize the heuristic approach to select a pivot among $cand(X)$. Since it is known that each vertex in $cand(X)$ can reach only the vertices before its occurrence, hence we select the last vertex in $cand(X)$. Therefore, given a list of vertices $cand(X)$ sorted in a *rev-topological* order, we heuristically select the last element of $cand(X)$ as the pivot. *Rev-topological* ordering visits all the root vertices in a $CDAG$ and performs a DFS to explore the subtrees of each root. The implementation of *rev-topological* can be achieved by performing a reverse topological sort, i.e., a parent in a $CDAG$ appears before the child. Using Figure 3b as a running example, only the vertices 1, 2, and 5 are pushed in the stack initially. The *rev-topological* order obtained for the $CDAG$ in Figure 3b is $\{1, 3, 6, 4, 2, 5\}$, vertex 5 is selected as a pivot. The run time cost of the *rev-topological* ordering is of linear order. In case we have more than one choice for a node selection, a lexicographical selection is incorporated to provide the uniqueness.

Datasets	$\|U\|$	$\|V\|$	$\|E\|$	Bicliques	LCM-MBC	iMBEA	PMBE	PMBE_pivot	PMBE_rev-top	Index Construction
Corporate Leadership	20	44	99	66	0.002	0.003	0.003	0.004	**0.002**	0.005
Unicode	254	868	1255	460	0.028	0.03	0.028	0.031	**0.025**	0.025
UCforum	899	1421	33720	16261	1.068	1.299	0.77	1.1	**0.76**	0.063
Service	10106	16730	50632	49538	12.682	14.37	13.605	**12.675**	14.657	8.645
Movielens u-t (User-Tag)	4009	20537	95580	166380	594.952	857.364	**438.126**	623.333	474.09	3.989
Movielens u-i (User-Movie)	4009	11610	95580	2365457	31.919	43.651	**29.016**	33.021	29.913	5.789
MovieLens t-i(Tag-Movie)	16528	24129	95580	140266	244.309	220.508	201.62	246.628	**193.478**	14.159
Wikinews (News)	1408	26546	193618	27729	5.483	10.687	**2.668**	5.46	2.861	1.413
Wikibooks (Books)	2884	30997	201727	41713	6.636	13.323	**4.265**	6.753	4.629	2.315

Table 1: Real world datasets and their respective running time (sec.) for enumerating all maximal bicliques

After discussing the techniques for addressing the challenges of pruning and ordering, we devise our algorithm PMBE.

3.5 PMBE

The algorithm incorporates the pivot pruning and *rev-topological* ordering of vertices, utilizing the $CDAG$ for their effective implementation. The heuristic approach is then suggested for fast pivot selection.

Algorithm. Algorithm 1 is the pseudo code for our approach of enumerating all maximal bicliques. Given a bipartite graph $G = (U \cup V, E)$, the vertex set V is considered as the candidate initially. The algorithm is implemented in two steps: firstly, we create $CDAG$ and utilize it to order the candidate set $cand$ offline (lines 2-3) and secondly, to enumerate all the maximal bicliques for $cand$ (line 5) using the function PMBE. The first step has been discussed in detail already, we further discuss the enumeration part. The inputs for enumeration are initially vertex set $B = \phi$, U, and $cand = V$.

Heuristic pivot selection. PMBE starts with the pivot selection, using Proposition 3, we heuristically select the last vertex from the sorted $cand$ (line 9).

Pruning using the pivot. After selecting the pivot, PMBE enumerates maximal bicliques for the $cand$ (line 10-16) by utilizing the optimized pivot pruning technique which uses the function $rangefinder$ (line 11). The $rangefinder$ returns *true* for a vertex *v*, if it is *reachable* from the *pivot*, hence the sub search space of *v* is pruned. After pruning, the maximal biclique is enumerated in line 12. Thus, eliminating a vertex *v* optimizes the algorithm by removing the overhead of biclique generation and duplicate verification.

Subsequently, the maximal biclique is reported in line 14. The recursive call (line 16) with the reduced $cand$ and increased B is only proceeded if there are any candidates present (line 15). The effectiveness of PMBE and LCM-MBC can be compared by using Figure 2, which appreciates two aspects of the PMBE over LCM-MBC. First, the duplicate bicliques (red boxes) have been decreased, second, the efficient pivot selection is achieved using the *rev-topological* order.

Time Complexity. The effective run time of PMBE is from line 7 to line 16 in Algorithm 1. The pivot selection line 9 is done using a heuristic approach in constant time. The complexity of rangefinder is $O(|\mathrm{R_{pivot}}||\mathrm{R_v}|)$, which is almost constant. The function is recursively called only for the maximal bicliques and the total number of maximal bicliques is β (line 16). Each maximal biclique corresponds to a node in the search space with a cost of $O(|V|d_{max})$ each, d_{max} is the maximum degree in V. Furthermore, the total running time complexity of the algorithm is the same as of the enumeration of the search space, i.e., $O(|V|d_{max}\beta)$.

Space Complexity. The space complexity depends on the implementation details. It is independent of the number of maximal bicliques as we do not save them in memory. The major contribution of space complexity is input bipartite graph which requires $O(|E|)$ space and the enumeration in memory. Since we are using DFS the space for enumeration tree requires $O(|V|d_{max})$. Each node in the enumeration tree is a biclique which is $O(|U| + |V|)$. The $CDAG$ created contains each vertex in $cand$, which requires $O(|U|)$ space. Therefore, the total space complexity is $O(|V|d_{max} + |E| + |V| + |U| + |V|) = O(d_{max}|V| + |E|)$.

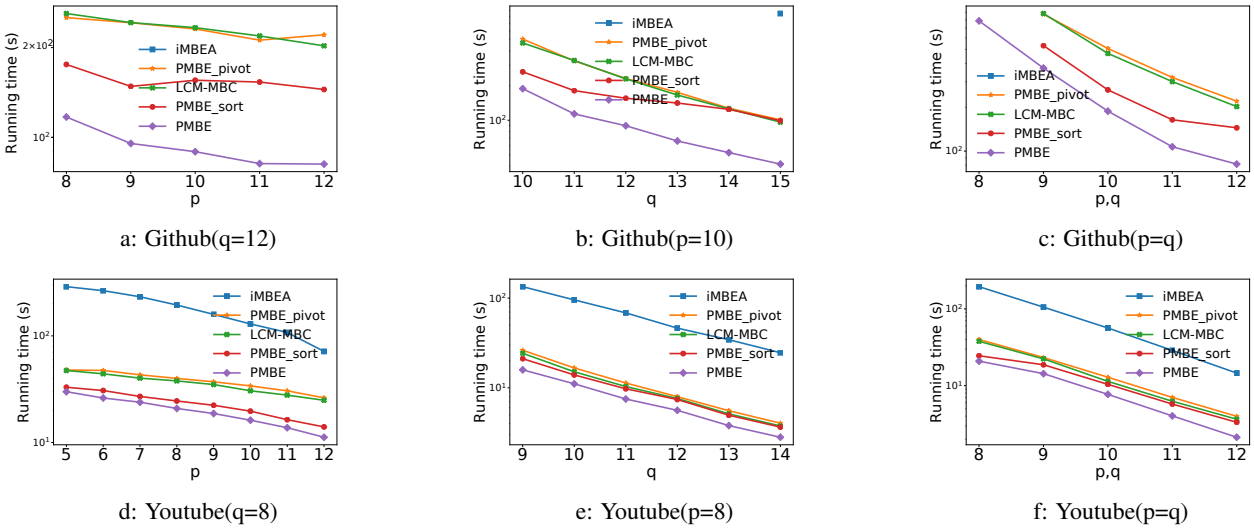
4 Experimental Results

The previous sections laid the conceptual foundations for the PMBE's potential to efficiently enumerate all maximal bicliques. In this section, we evaluate the efficiency of PMBE by comparing with the other state-of-the-art algorithms on real world datasets across numerous domains.

Experimental Setup. All the experiments were conducted on Eclipse IDE, deployed on the platform 64x Intel(R)Core(TM) i5-6400T with CPU frequency 2.20GHz and 8 GB RAM, running Windows 10 Enterprise operating system. To perform a fair and comprehensive comparison between the algorithms we only clock the running time of each algorithm. The running time of each algorithm is averaged over 10, 7 or 5 runs for datasets that can be finished within 10 minutes, half an hour or one hour. The experiments were performed without using any kind of parallelism, i.e., single core was used. All the algorithms were implemented in Java.

Datasets. To establish the dominance of our algorithm, we have selected the datasets s.t. they cover diversified real world application domains. Doing so will demonstrate the capabilities of the PMBE algorithm to efficiently enumerate all the maximal bicliques for a wide range of domains and graph characteristics. The datasets were obtained from KONECT repository [Kunegis, 2013]. Table 1 shows the list of datasets and their corresponding properties.

Algorithms. We now proceed to inspect the performances of the following five algorithms. Firstly, **iMBEA**, the algorithm is inspired by the classical BK algorithm [Bron and Kerbosch, 1973] with an exponential time complexity of $O(d_{max}|V|\beta)$. It employs non-decreasing sorting on $cand$ for efficient branching and pruning techniques to remove paths that cannot lead to maximal bicliques. Secondly, **LCM-MBC**, an optimized version of LCM-MBC [Li *et al.*, 2007]



a: Github(q=12) b: Github(p=10) c: Github(p=q)

d: Youtube(q=8) e: Youtube(p=8) f: Youtube(p=q)

Figure 4: Large real world datasets with thresholds

for bipartite graphs. The optimized version does not require any unnecessary post-processing and duplicate biclique enumeration [Zhang *et al.*, 2014]. Thirdly, **PMBE**, pivot based algorithm which is proposed in this paper. Lastly, to establish the effectiveness of pivot pruning and *rev-topological* ordering individually we use **PMBE_pivot** and **PMBE_rev-top**, where an algorithm uses only the pivot technique or the ordering technique individually. The runtime complexity of PMBE_pivot and PMBE_rev-top are of the same order of PMBE. Apart from the running time of the algorithms, the index construction time which includes the creation of $CDAG$ and performing *rev-topological* ordering is also shown in Table 1. We observe the improvement in the performance of PMBE with the increase in the size of the dataset (Table 1).

Results and Discussion. Table 1 illustrates the performance of all the algorithms on real datasets. From these experiments, we conclude that PMBE has outperformed iMBEA and LCM-MBC in almost all the scenarios. However, for Unicode and Service the PMBE_rev-top and PMBE_pivot outperforms all the other algorithms. We observe, at least one of the versions of PMBE has outmatched iMBEA and LCM-MBC. The results of the experiments also imply the following conclusions: (i) we perceive that ordering plays a more crucial role than the pivot as the efficiency obtained from pivot pruning is less, compared to *rev-topological* order. (ii) Although the individual algorithms are less efficient, the combination of the two techniques enhances the algorithm PMBE further. We extend PMBE to find large maximal bicliques in large datasets, i.e., GitHub ($|U| = 56519$, $|V| = 177386$ and $|E| = 440237$) and Youtube ($|U| = 94238$, $|V| = 124325$ and $|E| = 293360$). Figure 4 displays the results of the algorithms which are all in our favor. The threshold for a biclique is the number of minimum vertices in the two sets of a maximal biclique. p and q are the thresholds for each of the biclique vertex sets [Li *et al.*, 2007]. An essential finding from these experiments is the superiority of the PMBE across the entire domain of the datasets.

5 Related Work

Even though an enormous amount of research has been undertaken for the problem of enumerating maximal bicliques in general graphs, there are only limited algorithms which are directly applied to our problem of enumerating all maximal bicliques. Zhang [Zhang *et al.*, 2014] proposed the algorithm iMBEA, inspired by the traditional BK algorithm for clique enumeration. It has demonstrated better efficiency and scalability as compared to LCM-MBC in their paper. However, it still generates many duplicate bicliques, which reduce its efficiency. Many algorithms like [Makino and Uno, 2004; Zaki and Hsiao, 2002; Uno *et al.*, 2004] proposed converting the biclique problem to some other domain, such as clique or frequent item set. The drawback of these conversion techniques lies in their implementations, which require either pre or post processing steps to obtain corresponding bicliques. Hence, reductions to other domains present practical and scalability problems [Zhang *et al.*, 2014], and may not be fully utilized. Recently, Apurba [Das, 2019], proposed ParMBE, which is a parallelized biclique enumeration algorithm.

6 Conclusions

In this paper, the problem of enumerating all maximal bicliques from a bipartite graph has been explored in detail. We present the PMBE algorithm which incorporates the pivot pruning using the $CDAG$ structure and propose the *rev-topological* ordering for heuristic pivot selection. The PMBE proposed contains advanced pruning and ordering techniques, which reduce the search space without introducing any local extra cost. We conduct extensive experiments on real datasets across various domains to demonstrate the superiority of PMBE over the previous algorithms.

Acknowledgements

This work was jointly supported by the ARC Discovery Projects under Grant No. DP170104747 and DP200103700.

References

[Agrawal *et al.*, 1989] Rakesh Agrawal, Alexander Borgida, and Hosagrahar Visvesvaraya Jagadish. Efficient management of transitive relationships in large data and knowledge bases. *ACM SIGMOD Record*, 18(2):253–262, 1989.

[Alzahrani and Horadam, 2019] Taher Alzahrani and Kathy J. Horadam. Finding maximal bicliques in bipartite networks using node similarity. *Applied Network Science*, 4(1):21:1–21:25, 2019.

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

[Damaschke, 2014] Peter Damaschke. Enumerating maximal bicliques in bipartite graphs with favorable degree sequences. *Information Processing Letters*, 114(6):317–321, 2014.

[Das, 2019] Apurba Das. *Incremental and parallel algorithms for dense subgraph mining*. PhD thesis, Iowa State University, 2019.

[Driskell *et al.*, 2004] Amy C Driskell, Cécile Ané, J Gordon Burleigh, Michelle M McMahon, Brian C O'Meara, and Michael J Sanderson. Prospects for building the tree of life from large sequence databases. *Science*, 306(5699):1172–1174, 2004.

[Gibson *et al.*, 2005] David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 721–732. ACM, 2005.

[Kaytoue *et al.*, 2011] Mehdi Kaytoue, Sergei O Kuznetsov, Amedeo Napoli, and Sébastien Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Information Sciences*, 181(10):1989–2001, 2011.

[Kunegis, 2013] Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350. ACM, 2013.

[Li *et al.*, 2007] Jinyan Li, Guimei Liu, Haiquan Li, and Limsoon Wong. Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1625–1637, 2007.

[Makino and Uno, 2004] Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Scandinavian Workshop on Algorithm Theory*, pages 260–272. Springer, 2004.

[Mouret *et al.*, 2011] Sylvain Mouret, Ignacio E. Grossmann, and Pierre Pestiaux. Time representations and mathematical models for process scheduling problems. *Computers & Chemical Engineering*, 35(6):1038–1063, 2011.

[Schweiger *et al.*, 2011] Regev Schweiger, Michal Linial, and Nathan Linial. Generative probabilistic models for protein–protein interaction networks—the biclique perspective. *Bioinformatics*, 27(13):i142–i148, 2011.

[Uno *et al.*, 2004] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *Fimi*, volume 126, 2004.

[Vanahalli and Patil, 2019] Manjunath K Vanahalli and Nagamma Patil. An efficient parallel row enumerated algorithm for mining frequent colossal closed itemsets from high dimensional datasets. *Information Sciences*, 496:343–362, 2019.

[Wu *et al.*, 2018] Liuyi Wu, Lijun Dong, Yi Wang, Feng Zhang, Victor E Lee, Xiaojun Kang, and Qingzhong Liang. Uniform-scale assessment of role minimization in bipartite networks and its application to access control. *Physica A: Statistical Mechanics and its Applications*, 507:381–397, 2018.

[Xiang *et al.*, 2011] Yang Xiang, Philip RO Payne, and Kun Huang. Transactional database transformation and its application in prioritizing human disease genes. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(1):294–304, 2011.

[Yoshinaka, 2011] Ryo Yoshinaka. Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In Giancarlo Mauri and Alberto Leporati, editors, *Developments in Language Theory - 15th International Conference, DLT 2011, Milan, Italy, July 19-22, 2011. Proceedings*, volume 6795 of *Lecture Notes in Computer Science*, pages 429–440. Springer, 2011.

[Zaki and Hsiao, 2002] Mohammed J Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM, 2002.

[Zhang *et al.*, 2014] Yun Zhang, Charles A Phillips, Gary L Rogers, Erich J Baker, Elissa J Chesler, and Michael A Langston. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. *BMC bioinformatics*, 15(1):110, 2014.