Efficient Maximum (α, β)-Quasi Biclique Computation on Bipartite Graphs

Submission ID: 530

Abstract. A bipartite cohesive subgraph refers to a bipartite subgraph in which the vertices are highly interconnected. In this paper, we consider the (α, β)-*quasi biclique* model, which allows for connection proportions on two sides of vertices to be no less than α and β, respectively. Building upon this model, we propose the maximum (α, β)-quasi biclique search problem, which finds applications in various domains, including abnormal behavior detection, community group search, and biological gene expression detection. Due to the inefficiency of the baseline, we design effective preprocessing methods, upper bounding and reduction techniques, and advanced branching strategies, which collectively constitute our optimized algorithm called `MVQB`. Finally, through extensive experiments on real-world bipartite graphs, we validate the efficiency improvement of our proposed algorithm `MVQB` over the baseline, which demonstrates a speed improvement of up to six orders of magnitude.

Keywords: bipartite cohesive graphs · maximum (α, β)-quasi biclique

1 Introduction

Bipartite graphs serve as a robust structure for modeling interactions between two distinct groups of entities [5,17]. This unique property enables these graphs to effectively represent and analyze complex relationships in many real-world scenarios, such as collaboration networks [13], customer-product networks [26], and user-page networks [4]. Furthermore, *dense* or *cohesive* subgraphs are particularly valuable in bipartite graphs as they often contain significant information relevant to addressing real-world challenges [10, 20, 29]. These subgraphs, characterized by a high degree of connectivity between their vertices, can reveal patterns and anomalies that are crucial for practical applications. For example, in fraud detection [29], the cohesive bipartite subgraphs help identify unusual groupings of interactions that may indicate fraudulent activity.

To identify dense subgraphs within bipartite graphs, researchers have proposed a range of cohesive subgraph models. Some of the representatives include bicliques [20], (α, β)-bicores [10], and k-biplexs [24, 29]. These cohesive models serve diverse purposes and impose varying conditions on the connectivity patterns within the subgraphs. For instance, bicliques require complete interconnectivity between the two sides of vertices, while (α, β)-bicores and k-biplexes relax bicliques by requiring that each vertex connects to at least a *constant* number and disconnects from at most a *constant* number of vertices on the opposite side, respectively. Since these cohesiveness constants are unrelated to the size of cohesive subgraph, the identified cohesive subgraphs with unreasonable cohesiveness constants often exhibit low density, rendering them inadequate for effectively representing a cohesive structure in practice. In this paper, we focus on the (α, β)-quasi biclique model, which requires that the connection of each vertex in the subgraph reaches a certain *proportion* in relation to the size of vertices on the opposite side. Based on this cohesive bipartite model, we investigate the Maximum (α, β)-Quasi Biclique Search (MQBS) problem.

Challenges. The MQBS problem has been proven to be NP-hard [17]. Previous studies [17,28] then focus on either heuristic or approximation methods; however, these approaches cannot be modified to guarantee an exact solution for MQBS. To efficiently find the exact solution, a common approach is to adopt the branch-and-bound framework, similar to that used for other cohesive models mentioned earlier. However, unlike the biclique and k-biplex models, a notable difference of quasi-bicliques is that they are non-hereditary, i.e., an induced subgraph of an (α, β)-quasi biclique is not necessarily an (α, β)-quasi biclique. This implies that we cannot leverage the nice hereditary property to design useful pruning and reduction techniques in the branch-and-bound framework.

Our Solutions. We propose two exact algorithms based on the branch-and-bound framework for solving the MQBS problem. First, we provide an intuitive baseline method called ENUM. This method initially reduces the input graph into a bicore, then adopts a random vertex selection strategy to exhaustively explore all vertex combinations through a simple binary branching method. Although the methodology of ENUM is straightforward, it suffers from poor graph preprocessing, a lack of effective upper bounding and reduction rules, and a naive branching strategy. To address these limitations, we propose our advanced algorithm, termed MVQB. MVQB introduces a more powerful preprocessing method, which effectively utilizes the common neighbor information between two vertices on the same side and graph coloring techniques. Subsequently, we design a series of upper bounding methods and reduction rules, which are useful for pruning unnecessary branches and reducing the candidate sets during enumeration. Finally, to improve the performance of MVQB in branching, we devise an advanced pivot-based branching strategy, which selects a pivot vertex and conducts specific branching operations based on this pivot. Compared to the simple branching in ENUM, the pivot-based branching effectively leverages the inherent characteristics of the (α, β)-quasi biclique model, thereby improving performance.

Contributions. The contributions of this paper can be summarized as follows.

- We propose exact solutions for the MQBS problem, i.e., a baseline approach called ENUM and an advanced yet efficient method named MVQB.
- We develop a series of novel techniques in MVQB, including refined graph preprocessing techniques, upper bounding and reduction rules, and a novel branching method. These techniques effectively mitigate the inherent limitations of the (α, β)-quasi biclique model, which lacks the hereditary property.
- We conduct extensive experiments between ENUM and MVQB across ten real-world datasets. The results demonstrate that 1) MVQB outperforms ENUM by

up to six orders of magnitude in practical efficiency; and 2) the proposed advanced techniques designed for `MVQB` improve the algorithmic performance.

2 Preliminaries

Let $G = (L \cup R, E)$ be an undirected, unweighted bipartite graph, where L and R are two disjoint sets of vertices, and $E \subseteq L \times R$ represents the set of edges. We denote the total number of vertices in G by $|L| + |R| = n$ and the number of edges by $|E| = m$, respectively. Given a pair of vertex sets (X, Y) with $X \subseteq L$ and $Y \subseteq R$, we use $G[X \cup Y] = (X \cup Y, E_{X,Y})$ to define a subgraph of G induced by the vertex sets X and Y, where $E_{X,Y} = \{(x, y) \in E \mid x \in X, y \in Y\}$. For a vertex $u \in X$ (resp. $v \in Y$), the neighbor set and degree of u (resp. v) are denoted by $\Gamma_G(u, Y)$ and $\delta_G(u, Y)$ (resp. $\Gamma_G(v, X)$ and $\delta_G(v, X)$), respectively, where $\delta_G(u, Y) = |\Gamma_G(u, Y)|$ (resp. $\delta_G(v, X) = |\Gamma_G(v, X)|$). Particularly, when the context is clear, we use $\Gamma_G(u)$ and $\delta_G(u)$ to denote the neighbor set and degree of a vertex u, respectively. Besides, we define the common neighbors of two vertices u_1 and u_2 from the same side of vertices (i.e., both from L or R) as $\Delta(u_1, u_2)$, i.e., $\Delta(u_1, u_2) = \{w \mid (u_1, w) \in E \text{ and } (u_2, w) \in E\}$.

A classic bipartite subgraph model is called *biclique*, which has complete interconnectivity between the two sides of vertices. Obviously, the biclique model is the most cohesive subgraph model in bipartite graphs. To extend the practical applicability of bicliques, we consider a relaxed model of bicliques.

Definition 1 ((α, β)-Quasi Biclique). *Given a bipartite graph $G = (L \cup R, E)$ and two real numbers α, β in $[0, 1]$, an induced subgraph $G([X \cup Y])$ is an (α, β)-*quasi biclique (`QBC`) *if and only if every vertex $u \in X$ has a degree no less than $\alpha \cdot |Y|$ and every vertex $v \in Y$ has a degree no less than $\beta \cdot |X|$.*

Compared to the biclique model, our proposed `QBC` model introduces two additional parameters α and β to relax the requirement of full connectivity. Note that when $\alpha = \beta = 1$, the `QBC` model degenerates to the biclique model. Nonetheless, for general cases, `QBC` differs greatly from the biclique model in that the former lacks the *hereditary property* inherent to the latter. More specifically, any induced subgraph of a biclique remains a biclique; however, this is not always the case for `QBC`. We next introduce the problem studied in this paper.

Problem 1. Given a bipartite graph $G = (L \cup R, E)$, two values $\alpha, \beta \in (0.5, 1]$, and thresholds θ_L, θ_R, the Maximum (α, β)-Quasi Biclique Search (`MQBS`) problem aims to find an (α, β)-quasi biclique $G([X \cup Y])$ with the largest cardinality, where $|X| \geq \theta_L$ and $|Y| \geq \theta_R$.

We note that in the above `MQBS` problem, we introduce two practical constraints, which are based on the followings. First, without imposing constraints on the parameters α and β, `MQBS` may yield a disconnected subgraph, which is less useful in practice. On the other hand, if the values of α and β lie within the range $(0.5, 1]$, we can can always ensure that the obtained (α, β)-quasi biclique is a connected subgraph, as proved in the below Lemma 1. Second, without imposing constraints on the size of the returned subgraph, `MQBS` will spend considerable time searching for solutions that are too small to be practically meaningful.

Lemma 1. *Given two real numbers α, β in $(0.5, 1]$, any two vertices in the same side of vertices of an (α, β)-quasi biclique have at least one common neighbor.*

The omitted proofs in this paper can be found in Appendix of the technical report [1]. We remark that the `MQBS` problem is NP-hard as proved in [17]. In this paper, we focus on providing a practically efficient exact algorithm for `MQBS`.

2.1 A Baseline Solution

A straightforward method to find the maximum (α, β)-quasi biclique is to enumerate the vertex sets and return the one with the largest cardinality. Such a baseline (and our advanced algorithm introduced in Section 3) follow the *branch-and-bound* framework, which recursively solves the problem via a process of *branching*. This process involves dividing the problem of finding the largest (α, β)-quasi biclique into multiple sub-problems. Each sub-problem, or branch, is defined by a unique pair of disjoint vertex sets, $(S_L \cup S_R, C_L \cup C_R)$. Here, the partial set $S_L \cup S_R$ contains vertices that form an (α, β)-quasi biclique, which must be included in any solution within this specific branch. Conversely, the candidate set $C_L \cup C_R$ consists of vertices that may potentially be added to $S_L \cup S_R$ to create a larger solution. The solution of a branch is clearly the largest (α, β)-quasi biclique within that branch. A given (α, β)-quasi biclique is considered part of a branch $(S_L \cup S_R, C_L \cup C_R)$ if it 1) includes all vertices in $S_L \cup S_R$ and 2) is a subgraph of $G[(S_L \cup S_R) \cup (C_L \cup C_R)]$. Thus, solving the branch $(\emptyset, L \cup R)$ is equivalent to finding the maximum (α, β)-quasi biclique in $G[L \cup R]$.

As mentioned in Section 2, an (α, β)-quasi biclique does not enjoy the hereditary property for efficient pruning during the enumeration. In other words, our baseline solution only examines the current vertex set based on the definition of (α, β)-quasi biclique and the size constraint associated with the problem. To speed up the whole enumeration process, we provide a necessary definition for graph preprocessing operations based on the following definition.

Definition 2 ((α, β)-Bicore [14]). *Given a bipartite graph $G = (L \cup R, E)$ and two integers α and β, an (α, β)-bicore $G([X \cup Y])$ is the maximal induced subgraph of G in which all the vertices in $u \in X$ have degree at least α and all the vertices in $v \in Y$ have degree at least β.*

Based on Definition 2, we can easily obtain the following graph preprocessing.

Lemma 2. *An (α, β)-quasi biclique $G[X \cup Y]$ is in an $(\lceil \alpha |Y| \rceil, \lceil \beta |X| \rceil)$-bicore.*

We summarize our baseline method `ENUM` in Algorithm 1, where the description is deferred to Appendix of the technical report [1] due to the space limit.

Algorithm 1: The baseline method: `ENUM`

```
   Input: A bipartite graph $G = (L \cup R, E)$, two real numbers $\alpha$ and $\beta$ in
          $(0.5, 1]$, and two thresholds $\theta_L$ and $\theta_R$
   Output: The maximum $(\alpha, \beta)$-quasi biclique
1  $B^* \leftarrow \emptyset$;
2  Reduce $G$ to its $(\lceil \alpha \cdot \theta_R \rceil, \lceil \beta \cdot \theta_L \rceil)$-bicore by Lemma 2;
3  Enum_Rec($\emptyset$, $L \cup R$);
4  return $G[B^*]$;
   Procedure: Enum_Rec($S_L \cup S_R, C_L \cup C_R$)
5  if $|S_L \cup S_R \cup C_L \cup C_R| \leq |B^*|$ or $|S_L \cup C_L| < \theta_L$ or $|S_R \cup C_R| < \theta_R$ then
6      return;
7  if $G[(S_L \cup C_L) \cup (S_R \cup C_R)]$ is an $(\alpha, \beta)$-quasi biclique then
8      $B^* = (S_L \cup C_L) \cup (S_R \cup C_R)$; return;
9  $v \leftarrow$ Randomly select a vertex from $C_L \cup C_R$;
10 Enum_Rec($S_L \cup S_R \cup \{v\}$, $C_L \cup C_R \backslash \{v\}$);
11 Enum_Rec($S_L \cup S_R$, $C_L \cup C_R \backslash \{v\}$);
```

Limitations of the Baseline. Although the idea of `ENUM` is quite straightforward, it still has several limitations. First, despite some preprocessing operations, the candidate set of `ENUM` remains large, which affects the subsequent exact search in the branch-and-bound framework. Second, `ENUM` does not apply any effective upper bound techniques or reduction rules specific to (α, β)-quasi bicliques, preventing it from pruning unnecessary branches and candidate vertices. Third, the branching strategy of `ENUM`, based on random vertex selection, is simple, leading to numerous search operations during the branching process.

3 Our Advanced Approach

To solve the limitations of `ENUM`, we introduce our advanced method, `MVQB`, for the `MQBS` problem in this section. In particular, we first present a more efficient graph preprocessing technique before conducting the branch-and-bound search in Section 3.1. Then, we introduce effective upper bounding techniques and reduction rules tailored for (α, β)-quasi bicliques in Section 3.2. Finally, Section 3.3 gives a more sophisticated branching strategy based on pivot vertices, along with our advanced algorithm `MVQB` equipped with all the proposed techniques.

3.1 A More Effective Graph Preprocessing Method

Effective graph preprocessing techniques can further reduce the search space that needs to be explored in the enumeration. However, `ENUM` only considers the degree of vertices during the graph preprocessing, while overlooking the significance of the relationships between vertex pairs. Thus, we propose a new preprocessing technique based on the number of common neighbors between vertex pairs.

Proposition 1. *If there exist two vertices $u_1, u_2 \in L$ (resp. $u_1, u_2 \in R$) such that $\varDelta(u_1, u_2) < 2\lceil \alpha \cdot \theta_R \rceil - \theta_R - 1$ (resp. $2\lceil \beta \cdot \theta_L \rceil - \theta_L - 1$), then vertices u_1 and u_2 cannot coexist in an (α, β)-quasi biclique with threshold θ_R (resp. θ_L).*

By Proposition 1, we can determine whether two vertices on the same side of a bipartite graph can coexist in a solution by calculating the number of common neighbors between them. We next introduce our new preprocessing technique.

Graph Coloring-based Preprocessing Technique. Given a bipartite graph $G = (L \cup R, E)$ and two thresholds θ_L and θ_R, any two vertices u_1 and u_2 on the same side of G can coexist in the same (α, β)-quasi biclique if and only if they do not violate the conditions specified in Proposition 1. Based on this, we formalize the coexistence relationships between vertices on the two sides, L and R, of G by abstracting them into two simple undirected graphs, G_L and G_R, respectively. Specifically, vertices on each side of the bipartite graph G can be mapped to the vertices within G_L or G_R, where the mapping method is as follows. If two vertices u_1 and u_2 on the same side of a bipartite graph can coexist within the same (α, β)-quasi biclique according to Proposition 1, an edge is generated between their corresponding vertices in the corresponding undirected graph, i.e., G_L or G_R. After constructing G_L and G_R, we apply a greedy graph coloring algorithm to the two undirected graphs, assigning each vertex a unique color such that no two adjacent vertices share the same color. To balance the computational time and the quality of the graph coloring, we adopt the fast greedy coloring heuristic presented in [23]. For a given vertex v, let $\mathcal{C}(v)$ represent the set of unique color numbers assigned to its adjacent vertices. Proposition 2 details our proposed graph coloring-based preprocessing technique.

Proposition 2. *Given a bipartite graph $G = (L \cup R, E)$ and two threshold values θ_L and θ_R, we can safely remove a vertex $v \in L$ (resp. $v \in R$) if $\mathcal{C}(v) < \theta_L - 1$ (resp. $\mathcal{C}(v) < \theta_R - 1$) in the corresponding undirected graph G_L (resp. G_R).*

We also present an example for the above technique in Appendix of [1].

3.2 Upper Bounding and Reduction Rules

Effective upper bounding and reduction rules can reduce the search space in the enumeration, thereby improving the practical performance of the branch-and-bound based algorithm. Generally speaking, for models with hereditary properties (such as bicliques), if the selected set in a particular branch no longer constitutes a feasible solution, we can easily prune that branch since any sub-branch containing the selected set cannot yield a feasible solution. However, due to the lack of hereditary property in the (α, β)-quasi biclique model, designing appropriate upper bounding and reduction rules is extremely challenging. Nevertheless, in this section, we address this challenge by focusing on certain properties of (α, β)-quasi bicliques. For clarity, we will illustrate the upper bounds and reduction rules using vertices from the set L. Unless specified otherwise, these upper bounds and reduction rules also apply to vertices in the vertex set R.

Upper Bounding Rules. The upper bound technique provides an estimation on the size of the optimum solution that can be obtained from a particular branch, and then uses this bound to determine whether the branch can be pruned or not. Next, we introduce an upper bound based on vertex degree, derived from the definition of (α, β)-quasi bicliques.

Lemma 3 (Degree-based Upper Bound). *Given a branch $B = (S_L \cup S_R, C_L \cup C_R)$, the degree-based upper bound UB_d is shown as $UB_d = UB_{dl} + UB_{dr}$, where $UB_{dl} = \lfloor \min_{v \in S_L} \delta_G(v, S_R \cup C_R)/\alpha \rfloor$ and $UB_{dr} = \lfloor \min_{v \in S_R} \delta_G(v, S_L \cup C_L)/\beta \rfloor$.*

With Lemma 3, we can compute the degree-based upper bound on the maximum (α, β)-quasi biclique attainable by a branch. If this upper bound is less than the size of the recorded optimum solution, we can safely prune the branch. Besides this upper bound, we also identify an important characteristic of (α, β)-quasi biclique that exhibits monotonicity, which enables the design of another efficient upper bound technique. This characteristic relates to the number of missing neighbors in the selected set of a branch. Specifically, within a given branch $B = (S_L \cup S_R, C_L \cup C_R)$, the number of neighbor vertices absent in S_R (resp. S_L) for a vertex $v \in S_L$ (resp. $v \in S_R$) demonstrates a non-decreasing characteristic during the branching process. Our second upper bounding technique is to calculate the upper bound of missing neighbors allowable for the vertices in the selected set of a branch, and then determine whether this bound has exceeded the current number of missing neighbors in the selected set. We summarize this upper bound technique in Lemma 4.

Lemma 4 (Missing Degree-based Upper Bound). *Given a branch $B = (S_L \cup S_R, C_L \cup C_R)$, the missing degree-based upper bounds UB_{ml} and UB_{mr} on the left and right sides of B are shown as $UB_{ml} = \lfloor (1-\alpha) \cdot UB_{dl} \rfloor$ and $UB_{mr} = \lfloor (1-\beta) \cdot UB_{dr} \rfloor$.*

Lemma 4 only defines the maximum number of missing neighbors allowed for a branch. To unleash its pruning power, we let $\nabla(S_L)$ be the maximum missing degree of vertices on the left side of the selected set, i.e., $\nabla(S_L) = |S_R| - \min\limits_{u \in S_L} \delta_G(u, S_R)$. Note that a similar definition $\nabla(S_R)$ can be defined symmetrically on S_R. Now we can prune a branch if $\nabla(S_L) > UB_{ml}$ or $\nabla(S_R) > UB_{mr}$.

Reduction Rules. If there are vertices in the candidate set of a branch that cannot be added to the selected set to generate a larger solution, these vertices can be removed from the candidate set. For ease of presentation, we define the lower bound on the right side of a branch as $LB(B_R) = \left\lceil \frac{\nabla(S_L)}{1-\alpha} \right\rceil$. The lower bound on the left side of a branch can be defined symmetrically. Subsequently, we propose two novel reduction rules based on the information of degree and common neighbor, respectively.

Lemma 5 (Degree-based Reduction Rule). *Given a branch $B = (S_L \cup S_R, C_L \cup C_R)$ and a vertex $u \in C_L$ (resp. $u \in C_R$), u can be removed from the candidate set if $\delta_G(u, S_R \cup C_R) < \alpha \cdot LB(B_R)$ (resp. $\delta_G(u, S_L \cup C_L) < \beta \cdot LB(B_L)$).*

Lemma 5 only considers the degree of vertices. Below, we propose a reduction rule that takes into account the number of common neighbors between vertices.

Lemma 6 (Common Neighbor-based Reduction Rule). *For a branch $B = (S_L \cup S_R, C_L \cup C_R)$ and a vertex $u \in C_L$ (resp. $u \in C_R$), u can be removed from the candidate set if there exists a vertex $v \in S_L$ (resp. $v \in S_R$) such that $\Delta(u, v) < 2\lceil \alpha \cdot LB(B_R) \rceil - LB(B_R) - 1$ (resp. $\Delta(u, v) < 2\lceil \alpha \cdot LB(B_L) \rceil - LB(B_L) - 1$).*

3.3 Our Advanced Branching Method and the Final Algorithm

To address the limitations of `ENUM` in the branching strategy, we present our advanced branching method in this part. We also present our advanced algorithm, named `MVQB`, that integrates all the previously mentioned techniques.

Pivot-based Branching Rules. In the baseline `ENUM`, the branching strategy involves randomly selecting a vertex from the candidate set as the branching vertex. This arbitrary branching behavior often results in many unnecessary branches, significantly impacting the running time of the branch-and-bound algorithm (see our experiments). We introduce a sophisticated pivot-based branching rule as follows. We first define a methodology for finding a suitable *pivot vertex*, and then outline the specific branching methods based on the pivot vertex.

Definition 3 (Pivot Vertex). For a branch $B = (S_L \cup S_R, C_L \cup C_R)$, a vertex w is said to be the *pivot vertex* if w satisfies the following conditions:
(1) We first look for vertices that satisfy $\delta_G(w, S_R \cup C_R) < \alpha|S_R \cup C_R|$ when $w \in S_L$ or $\delta_G(w, S_L \cup C_L) < \beta|S_L \cup C_L|$ when $w \in S_R$;
(2) If there are no vertices that meet Condition (1), then we look for vertices that satisfy $\delta_G(w, S_R \cup C_R) < \alpha|S_R \cup C_R|$ when $w \in C_L$ or $\delta_G(w, S_L \cup C_L) < \beta|S_L \cup C_L|$ when $w \in C_R$;
(3) If multiple vertices satisfy Conditions (1) or (2), then the pivot vertex w is the one with the minimum degree among them.

It is straightforward to demonstrate that for any branch not yet forming an (α, β)-quasi biclique, a pivot vertex w that meets the criteria of Definition 3 can always be found. Moreover, depending on whether w resides in $S_L \cup S_R$ or $C_L \cup C_R$, two distinct branching cases are applicable.
Branching Case 1: $w \in S_L \cup S_R$. The purpose of this case is to incorporate a certain number of non-neighboring vertices of w into $S_L \cup S_R$, thus ensuring that any sub-branch containing the selected set cannot yield any valid (α, β)-quasi biclique. Without loss of generality, we assume that the pivot vertex $w \in S_L$, but the following description symmetrically applies to $w \in S_R$.

We first define the number of missing neighbors of w in the candidate set as $p = |C_R| - \delta_G(w, C_R)$. Then, we use q to denote the maximum number of non-neighbors of w that can be added to S_R from C_R:

$$q = \lfloor (1 - \alpha) \cdot \lfloor \delta_G(w, S_R \cup C_R)/\alpha \rfloor \rfloor - (|S_R| - \delta_G(w, S_R)). \quad (1)$$

The first half of Equation 1 represents the maximum number of non-neighbors that vertex w can additionally include in a valid (α, β)-quasi biclique, while the second half indicates the number of neighbors that w has already lost in the selected set. We remark that $p > q$ since the current graph is not a qualified (α, β)-quasi biclique, otherwise we can terminate the branching operation directly. Upon defining p and q, it becomes apparent that any sub-branch with the selected set can contain at most q vertices in $C_R \backslash \Gamma_G(w, C_R) = \{v_1, v_2, \dots, v_p\}$. Consequently, the branching strategy for Case 1 results in $q+1$ sub-branches. **(1)** The first sub-branch removes v_1 from C_R; **(2)** The i_{th} sub-branch removes v_i from C_R and includes $\{v_1, v_2, \dots, v_{i-1}\}$ to S_R, where $2 \leq i \leq q$; **(3)** The $(q+1)_{th}$ sub-branch removes $\{v_{q+1}, \dots, v_p\}$ from C_R and includes $\{v_1, v_2, \dots, v_q\}$ to S_R.

Branching Case 2: $w \in C_L \cup C_R$. The main objective of Case 2 is to move w to the selected set through a simple binary branching operation, thereby making one of the sub-branches satisfies the conditions of Case 1 again. The branching strategy of Case 2 generates the following two sub-branches. **(1)** The first sub-branch includes the pivot vertex w to $S_L \cup S_R$, i.e., $B_1 = (S_L \cup S_R \cup \{w\}, C_L \cup C_R \backslash \{w\})$; **(2)** The second sub-branch removes w from $C_L \cup C_R$, i.e., $B_2 = (S_L \cup S_R, C_L \cup C_R \backslash \{w\})$.

Our Advanced Algorithm: `MVQB`. After establishing the advanced pivot-based branching rule, we are ready to present our algorithm, namely `MVQB`, which integrates all the proposed techniques, in Algorithm 2. Basically, `MVQB` majorly consists of two parts: graph preprocessing (Lines 2-4) and recursive branch-and-bound search (Line 5). Unlike `ENUM`, which only reduces the input graph into a bicore based on the given thresholds, `MVQB` incorporates our advanced preprocessing technique in Line 3. To maximize the effects of graph preprocessing, `MVQB` also iteratively applies the bicore reduction and the graph coloring-based preprocessing technique until no more vertices can be removed (Lines 2-4). After the graph preprocessing stage, `MVQB` invokes the procedure `MVQB_Rec` to exhaustively search for the largest (α, β)-quasi biclique. Specifically, `MVQB_Rec` can be divided into three sub-stages, which are pruning (Lines 7-11), graph reduction (Line 12), and branching (Lines 13-20), respectively. Pruning conditions include: 1) the size of the current graph is smaller than the size of the recorded optimal solution, or the number of vertices on one side is less than the given threshold (Line 7); 2) the entire branch already forms a valid (α, β)-quasi biclique (Lines 9-10); 3) the degree-based upper bound in the current branch is no greater than the current size lower bound $|B^*|$, or the missing degree on any side in the selected set exceeds the missing degree-based upper bound (Line 11). Once a branch passes these pruning tests, `MVQB_Rec` applies the reduction rules proposed in Lemma 5 and Lemma 6 to further narrow down the candidate vertex set (Line 12). Following this, `MVQB_Rec` enters its final sub-stage, i.e., the branching stage (Lines 13-20). Unlike the random branching vertex selection strategy of the baseline `ENUM`, `MVQB_Rec` uses our proposed pivot-based branching method. That is, `MVQB_Rec` selects a pivot vertex w according to Definition 3 (Line 13), and divides into two different branching scenarios based on whether w is in the selected set S (Lines 14-17) or the candidate set C (Lines 18-20). If $w \in S$, for

Algorithm 2: Our advanced algorithm: `MVQB`

```
Input: A bipartite graph $G = (L \cup R, E)$, two real numbers $\alpha$ and $\beta$ in
       $(0.5, 1]$, and two thresholds $\theta_L$ and $\theta_R$
Output: The maximum $(\alpha, \beta)$-quasi biclique
1  $B^* \leftarrow \emptyset$;
2  while true do
3      Remove vertices that (1) do not belong to $(\lceil \alpha \cdot \theta_R \rceil, \lceil \beta \cdot \theta_L \rceil)$-bicore by
         Lemma 2; (2) meet the conditions of Proposition 2;
4      if no vertices are removed then break;
5  MVQB_Rec($\emptyset$, $L \cup R$);
6  return $G[B^*]$;
   Procedure: MVQB_Rec($S_L \cup S_R, C_L \cup C_R$)
7  if $S_L \cup S_R \cup C_L \cup C_R \leq |B^*|$ or $|S_L \cup C_L| < \theta_L$ or $|S_R \cup C_R| < \theta_R$ then
8      return;
9  if $G[(S_L \cup C_L) \cup (S_R \cup C_R)]$ is an $(\alpha, \beta)$-quasi biclique then
10     $B^* = (S_L \cup C_L) \cup (S_R \cup C_R)$; return;
11 if $UB_d \leq |B^*|$ or $\nabla(S_L) > UB_{ml}$ or $\nabla(S_R) > UB_{mr}$ then return;
12 Refine $C_L \cup C_R$ by the reduction rules from Lemma 5 and Lemma 6;
13 $w \leftarrow$ Find a pivot vertex according to Definition 3;
14 if $w \in S_L \cup S_R$ then
15     Assume $w \in S_L$; $p = |C_R| - \delta_G(w, C_R)$;
16     $q = \lfloor (1-\alpha) \cdot \lfloor \min_{v \in S_L} \delta(v, S_R \cup C_R)/\alpha \rfloor \rfloor - (|S_R| - \delta_G(w, S_R))$;
17     Create $q + 1$ sub-branches by Branching Case 1;
18 else
       // Branching Case 2
19     MVQB_Rec($S_L \cup S_R \cup \{w\}$, $C_L \cup C_R \backslash \{w\}$);
20     MVQB_Rec($S_L \cup S_R$, $C_L \cup C_R \backslash \{w\}$);
```

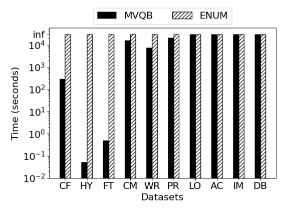
ease of description, we assume $w \in S_L$. `MVQB_Rec` then calculates the number of missing neighbors p regarding w in the candidate set (Line 15) and the maximum number of non-neighbors q that can still be added to the selected set (Line 16). Subsequently, we generate $q + 1$ sub-branches following Case 1 of the proposed pivot-based branching strategy (Line 17). On the other hand, if $w \in C$, a simple binary branching is required as in our Branching Case 2 (Lines 19-20).

4 Experiments

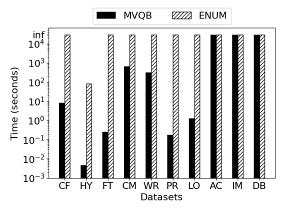
We conduct extensive experimental studies by comparing the baseline algorithm `Enum` (Section 2.1) and our advanced algorithm `MVQB` (Section 3). All algorithms are written in C++, compiled using g++ with -O3, and executed on a Linux server equipped with an Intel(R) Core(TM) i5-10500 CPU. The running time of each experiment is limited to 30,000 seconds. Our codes can be found in [1].

Table 1: Statistics of the datasets

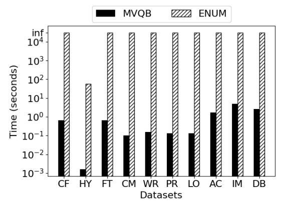
Dataset	Abbreviation	$\|L\|$	$\|R\|$	m	density	d_L	d_R	θ^*
Cfat	CF	200	200	1,534	3.84×10^{-2}	16	16	3
Hywikibooks	HY	242	1,714	2,379	5.70×10^{-1}	683	17	3
FilmTrust	FT	1,508	2,071	35,494	1.14×10^{-2}	244	1,044	46
CondMat	CM	16,726	22,015	58,595	2.00×10^{-2}	106	18	4
Writer	WR	89,356	46,213	144,340	3.50×10^{-3}	42	246	4
Producer	PR	48,833	138,844	207,268	3.10×10^{-3}	512	30	6
Location	LO	172,091	53,407	293,687	3.20×10^{-3}	28	12,189	5
Actor	AC	127,823	383,640	1,470,404	3.00×10^{-3}	294	646	8
IMDB	IM	428,439	896,308	3,782,463	1.00×10^{-3}	1,334	1,590	26
Dblp	DB	1,425,813	4,000,150	8,649,016	2.00×10^{-4}	951	119	7



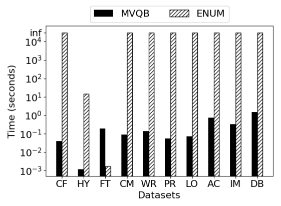
(a) $\alpha = \beta = 0.6$



(b) $\alpha = \beta = 0.7$



(c) $\alpha = \beta = 0.8$



(d) $\alpha = \beta = 0.9$

Fig. 1: Running time in seconds when varying α and β

Datasets. Our experiments select 10 real-world datasets from the KONECT Project[1]. Table 1 describes the statistics of the datasets, where the *density* is defined as $m/(|L| \cdot |R|)$, and d_L and d_R quantify the maximum degrees of vertices in vertex sets L and R, respectively. Moreover, the default threshold θ^* is set for different graphs. Especially, we set the $\alpha = \beta = 0.8$ as the default ratio.

4.1 Against the Baseline

Exp-1: Efficiency Evaluation of Varying Values of α and β. In this experiment, we compare `MVQB` with `ENUM` by varying values of α and β, where the results are shown in Figure 1. Overall, we find that `MVQB` demonstrates superior practical performance compared to `ENUM` across all different values of α and β, with this advantage being more obvious at larger values of α and β (e.g., 0.8

[1] http://konect.cc/

Table 2: Running time in seconds when varying thresholds

Algorithm	`ENUM`				`MVQB`			
Threshold / Dataset	θ^*-1	θ^*	θ^*+1	θ^*+2	θ^*-1	θ^*	θ^*+1	θ^*+2
CF	OOT	OOT	OOT	OOT	1.19	0.65	0.08	0.07
HY	OOT	57.30	0.01	0.01	0.06	0.01	0.01	0.01
FT	OOT	OOT	OOT	1027.00	0.75	0.66	0.09	0.04
CM	OOT	OOT	OOT	OOT	14.80	0.10	0.05	0.01
WR	OOT	OOT	OOT	OOT	8.75	0.15	0.07	0.04
PR	OOT	OOT	OOT	0.03	1.98	0.13	0.06	0.05
LO	OOT	OOT	OOT	0.04	0.30	0.13	0.07	0.07
AC	OOT	OOT	OOT	OOT	29.40	1.73	0.93	0.91
IM	OOT	OOT	OOT	OOT	4.01	3.56	0.46	0.31
DB	OOT	OOT	OOT	OOT	236.00	2.58	1.97	1.16

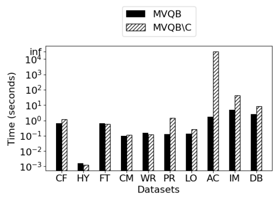
and 0.9). For example, when $\alpha = \beta = 0.9$, `MVQB` finds the maximum (α, β)-quasi biclique in the CF dataset in only 0.7 seconds, while `ENUM` fails to complete the task within the given 30,000 seconds. In other words, the advantage of `MVQB` over `ENUM` is on the order of five magnitudes. This advantage is primarily due to `MVQB`'s superior graph preprocessing techniques, upper bounds, reduction rules, and branching strategies. We also observe that both algorithms perform undesirable on certain datasets for α and β values of 0.6 and 0.7, as smaller values of α and β indicate an increased difficulty in solving `MQBS`.

Exp-2: Efficiency Evaluation of Varying Threshold Values. We next compare the two algorithms across different values of thresholds in Table 2. Similarly, we observe that our proposed advanced method, `MVQB`, significantly outperforms the baseline algorithm, `ENUM`, across all different values of thresholds, achieving a lead of nearly six magnitudes in some cases. For instance, in the CM dataset with the threshold $\theta = \theta^* + 2$, `ENUM` fails to find the optimum solution within the time limit, while our `MVQB` completes the task in only 0.01 seconds. Furthermore, as the threshold values increase, both algorithms show clear improvements in running time, which can be due to the improved graph preprocessing and pruning effects provided by higher threshold values.

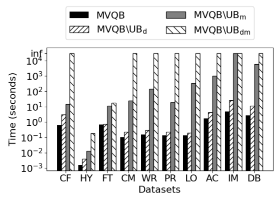
4.2 Ablation Studies

To further assess the effectiveness of our proposed advanced techniques adopted in `MVQB`, we conduct the following ablation studies.

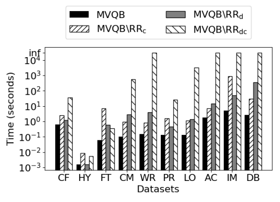
Exp-3: Efficiency Evaluation of Preprocessing Technique. In this experiment, we validate the contribution of the graph coloring-based preprocessing technique to the efficiency improvement of `MVQB` (and defer the detailed preprocessing information, i.e., the size of the remaining graph after preprocessing, to Appendix of [1]). Our algorithms used in this experiment are `MVQB` and `MVQB\C`, where the latter is a variant of Algorithm 2 without the graph coloring-based preprocessing technique. The results are presented in Figure 2(a). We find that the total running time of `MVQB` is less than that of `MVQB\C` across most datasets. In particular, on the AC dataset, `MVQB` shows an efficiency advantage over `MVQB\C`



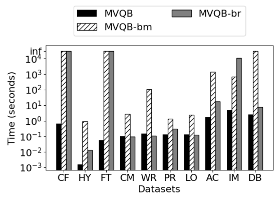
(a) Preprocessing



(b) Upper bounding rules



(c) Reduction rules



(d) Branching rules

Fig. 2: Ablation studies of the proposed techniques in our algorithm MVQB

that exceeds four magnitudes. This clearly demonstrates the effectiveness of our advanced preprocessing technique.

Exp-4: Efficiency Evaluation of Various Upper Bounds. We also implement multiple versions to thoroughly validate the effectiveness of our proposed upper bounding rules, namely MVQB, MVQB\$UB_d$, MVQB\$UB_m$, and MVQB\$UB_{dm}$. Specifically, (1) MVQB\$UB_d$ denotes Algorithm 2 without using the degree-based upper bound (Lemma 3), (2) MVQB\$UB_m$ refers to Algorithm 2 excluding the missing degree-based upper bound (Lemma 4), and (3) MVQB\$UB_{dm}$ indicates that Algorithm 2 is implemented without using either of the upper bounding rules. The results are shown in Figure 2(b). Overall, our proposed two upper bounding techniques significantly enhance the runtime efficiency of MVQB. Without using either upper bound, namely MVQB\$UB_{dm}$, eight out of ten datasets fail to find the optimum solution within the time limit. Furthermore, experimental results clearly suggest that the missing degree-based upper bound performs substantially better than the degree-based upper bound, as MVQB\$UB_d$ outperforms MVQB\$UB_m$ in the vast majority of datasets. For example, in the IM dataset, the running time of MVQB\$UB_d$ is only 1 second, while MVQB\$UB_m$ exceeds the time limit. Nevertheless, the degree-based upper bound still further improves the effectiveness of the missing degree-based upper bound across nearly all datasets.

Exp-5: Efficiency Evaluation of Various Reduction Rules. We implement several variants of our algorithm to assess the effectiveness of the proposed reduction rules, which include MVQB, MVQB\$RR_d$, MVQB\$RR_c$, and MVQB\$RR_{dc}$. Specifically, (1) MVQB\$RR_d$ represents Algorithm 2 without the degree-based reduction rule (Lemma 5), (2) MVQB\$RR_c$ represents Algorithm 2 omitting the common neighbor-based reduction rule (Lemma 6), and (3) MVQB\$RR_{dc}$ is the implementation of Algorithm 2 without both reduction rules. As shown in Figure 2(c), the two reduction rules we propose make a significant contribution to the runtime efficiency of MVQB, since the performance of MVQB is superior to that of MVQB\$RR_d$

and `MVQB`$\backslash RR_c$ across all datasets. More specifically, our common neighbor-based reduction technique `MVQB`$\backslash RR_c$ yields a relatively greater performance improvement for the `MVQB` algorithm compared to the degree-based reduction rule. For instance, in the IM dataset, the running time of `MVQB`$\backslash RR_d$ is 86 seconds, while the running time of `MVQB`$\backslash RR_c$ reaches 1,000 seconds. This clearly demonstrates that reduction rules designed based on common neighbor can greatly reduce the search space of the enumeration in certain scenarios.

Exp-6: Efficiency Evaluation of Varying Branching Rules. To validate the effectiveness of the proposed pivot-based branching rule, we use the following three algorithms with different branching strategies, which are `MVQB`, `MVQB-bm`, and `MVQB-br`, respectively. Specifically, (1) `MVQB-bm` represents the method where Algorithm 2 branches by selecting the vertex with the *minimum degree* in the candidate set, and (2) `MVQB-br` is the approach in which Algorithm 2 uses the same *random* branching vertex selection approach as the baseline `ENUM`. The performance of these variants is illustrated in Figure 2(d). From the figure, it is clear that our proposed pivot-based branching method outperforms the other two branching strategies in the vast majority of cases. For instance, in the AC dataset, `MVQB` finds the maximum (α, β)-quasi biclique in only 1.7 seconds, while `MVQB-bm` and `MVQB-br` take 17.5 seconds and 1412 seconds, respectively. More importantly, in the CH and FT datasets, `MVQB-bm` and `MVQB-br` fail to find the optimum solution within the given 30,000 seconds, whereas our pivot-based branching method identifies the optimum solution in 0.6 second. These results clearly demonstrate the superiority of our advanced branching method.

5 Related Work

Quasi Biclique. James et al. [2] introduced the definition of γ-quasi bicliques, which requires the number of edges is at least a certain proportion γ of the total number of edges possible between its two vertex sets. Building on this concept, Dmitry et al. [12] developed a Mixed Integer Programming approach to search for the largest one. Mishara et al. [21] considered the vertex connectivity, introducing the concept of ϵ-quasi bicliques for studying cases where the degree of connectivity on *one* side does not exceed a certain missing proportion ϵ on the other side. On this basis, Liu et al. [15] designed a scalable algorithm using Giraph. However, this model, due to imposing constraints only on vertices on one side, could generate unbalanced subgraphs. To address this issue, Liu et al. [17] proposed the concept of δ-quasi bicliques, requiring the connectivity ratio between each vertex in one vertex set and those in the other set to not fall below δ. They proved that finding the largest δ-quasi biclique is an NP-hard problem and introduced a heuristic algorithm to tackle this issue. Later, Wang [28] provided an approximation algorithm while violating δ by a bounded ratio. Chang et al. [5] extended this model to weighted bipartite graphs and utilized integer programming to solve the problem.

Other Cohesive Models in Bipartite Graphs. Eppstein et al. [11] proved that enumerating maximal bicliques cannot be done in polynomial time, which has inspired researchers to design more efficient algorithms [3,6]. Furthermore, there are studies that tackle the maximum biclique search problem, characterized by the vertex variant [9] and the edge variant [20,22,25,27], depending on the definition of "maximum" in terms of the total number of vertices and edges, respectively. For the (α, β)-cores, Ding et al. [10] introduced an index structure to study the online version, and Liu et al. [14] reduced the space complexity to $O(m)$. Liu et al. [16] studied the problem in a distributed fashion, while Luo et al. [19] considered the dynamic version of the problem. Zou [32] introduced the k-bitruss model, which requires each edge to be part of at least k "butterflies". Another related model is called the k-biplex, where each vertex in a k-biplex is allowed to disconnect at most k vertices in the other partition of vertex set. Based on the k-biplex, the problems of maximum k-biplex search [18,29] and maximal k-biplex enumeration [7,8,30,31] are extensively studied.

6 Conclusion

In this paper, we elaborated on the (α, β)-quasi biclique model and explore the problem of finding the maximum (α, β)-quasi biclique. We proposed an exact algorithm `MVQB` for the problem, which incorporates efficient preprocessing methods, effective upper bounding and reduction techniques, and advanced branching strategies. Finally, the experimental comparisons on real-world datasets confirm the superior performance of `MVQB` over the baseline method by up to 6 orders of magnitude faster, achieving significant improvement. In the future, we plan to design the parallel version of our proposed `MVQB` algorithm.

References

1. https://anonymous.4open.science/r/mvqb-978B/ (Technical Report and Code)
2. Abello, J., Resende, M.G., Sudarsky, S.: Massive quasi-clique detection. In: LATIN. pp. 598–612 (2002)
3. Abidi, A., Zhou, R., Chen, L., Liu, C.: Pivot-based maximal biclique enumeration. In: IJCAI. pp. 3558–3564 (2020)
4. Beutel, A., Xu, W., Guruswami, V., Palow, C., Faloutsos, C.: Copycatch: Stopping group attacks by spotting lockstep behavior in social networks. In: WWW. pp. 119–130 (2013)
5. Chang, W.C., Vakati, S., Krause, R., Eulenstein, O.: Exploring biological interaction networks with tailored weighted quasi-bicliques. BMC Bioinformatics **13**, 1–9 (2012)
6. Chen, L., Liu, C., Zhou, R., Xu, J., Li, J.: Efficient maximal biclique enumeration for large sparse bipartite graphs. PVLDB **15**(8), 1559–1571 (2022)
7. Dai, Q., Li, R.H., Cui, D., Liao, M., Qiu, Y.X., Wang, G.: Efficient maximal biplex enumerations with improved worst-case time guarantee. Proc. ACM Manag. Data **2**(3) (2024)
8. Dai, Q., Li, R.H., Ye, X., Liao, M., Zhang, W., Wang, G.: Hereditary cohesive subgraphs enumeration on bipartite graphs: The power of pivot-based approaches. Proc. ACM Manag. Data **1**(2) (2023)

9. Dawande, M., Keskinocak, P., Swaminathan, J.M., Tayur, S.: On bipartite and multipartite clique problems. Journal of Algorithms **41**(2), 388–403 (2001)
10. Ding, D., Li, H., Huang, Z., Mamoulis, N.: Efficient fault-tolerant group recommendation using alpha-beta-core. In: CIKM. pp. 2047–2050 (2017)
11. Eppstein, D.: Arboricity and bipartite subgraph listing algorithms. Information Processing Letters **51**(4), 207–211 (1994)
12. Ignatov, D.I., Ivanova, P., Zamaletdinova, A.: Mixed integer programming for searching maximum quasi-bicliques. In: ICNA. pp. 19–35 (2018)
13. Ley, M.: The DBLP computer science bibliography: Evolution, research issues, perspectives. In: SPIR. pp. 1–10. Springer (2002)
14. Liu, B., Yuan, L., Lin, X., Qin, L., Zhang, W., Zhou, J.: Efficient (α, β)-core computation: An index-based approach. In: WWW. pp. 1130–1141 (2019)
15. Liu, H.F., Su, C.T., Chu, A.C.: Fast quasi-biclique mining with Giraph. In: Proceedings of the IEEE International Congress on Big Data. pp. 347–354 (2013)
16. Liu, Q., Liao, X., Huang, X., Xu, J., Gao, Y.: Distributed (α, β)-core decomposition over bipartite graphs. In: ICDE. pp. 909–921 (2023)
17. Liu, X., Li, J., Wang, L.: Modeling protein interacting groups by quasi-bicliques: Complexity, algorithm, and application. TCBB **7**(2), 354–364 (2008)
18. Luo, W., Li, K., Zhou, X., Gao, Y., Li, K.: Maximum biplex search over bipartite graphs. In: ICDE. pp. 898–910 (2022)
19. Luo, W., Yang, Q., Fang, Y., Zhou, X.: Efficient core maintenance in large bipartite graphs. Proc. ACM Manag. Data **1**(3) (2023)
20. Lyu, B., Qin, L., Lin, X., Zhang, Y., Qian, Z., Zhou, J.: Maximum and top-k diversified biclique search at scale. The VLDB Journal **31**(6), 1365–1389 (2022)
21. Mishra, N., Ron, D., Swaminathan, R.: A new conceptual clustering framework. Machine Learning **56**, 115–151 (2004)
22. Peeters, R.: The maximum edge biclique problem is NP-complete. Discrete Applied Mathematics **131**(3), 651–654 (2003)
23. San Segundo, P., Rodríguez-Losada, D., Jiménez, A.: An exact bit-parallel algorithm for the maximum clique problem. Computers & Operations Research **38**(2), 571–581 (2011)
24. Sim, K., Li, J., Gopalkrishnan, V., Liu, G.: Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market and protein networks. Statistical Analysis and Data Mining **2**(4), 255–273 (2009)
25. Wang, J., Yang, J., Zhang, C., Lin, X.: Efficient maximum edge-weighted biclique search on large bipartite graphs. TKDE **35**(08), 7921–7934 (2022)
26. Wang, J., De Vries, A.P., Reinders, M.J.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: SIGIR. pp. 501–508 (2006)
27. Wang, K., Zhang, W., Lin, X., Qin, L., Zhou, A.: Efficient personalized maximum biclique search. In: ICDE. pp. 498–511 (2022)
28. Wang, L.: Near optimal solutions for maximum quasi-bicliques. Journal of Combinatorial Optimization **25**, 481–497 (2013)
29. Yu, K., Long, C.: Maximum k-biplex search on bipartite graphs: A symmetric-BK branching approach. Proc. ACM Manag. Data **1**(1) (2023)
30. Yu, K., Long, C., Liu, S., Yan, D.: Efficient algorithms for maximal k-biplex enumeration. In: SIGMOD. p. 860–873 (2022)
31. Yu, K., Long, C., P, D., Chakraborty, T.: On efficient large maximal biplex discovery. TKDE **35**(1), 824–829 (2023)
32. Zou, Z.: Bitruss decomposition of bipartite graphs. In: DASFAA. pp. 218–233 (2016)

A Omitted Proofs

A.1 Proof of Lemma 1

Proof. We prove this lemma by contradiction. Assume $H = G([X \cup Y])$ is an (α, β)-quasi biclique, where vertices $u_1, u_2 \in X$ have no common neighbors. By definition, $\delta_G(u_1) > 0.5|Y|$, $\delta_G(u_2) > 0.5|Y|$. If $\Gamma_G(u_1) \cap \Gamma_G(u_2) = \emptyset$, implying that there are $\delta_G(u_1) + \delta_G(u_2) > |Y|$ different vertices in the Y set, which contradicts the assumption. Thus, the proof is complete. □

A.2 Proof of Lemma 2

Proof. According to Definitions 1 and 2, Lemma 2 trivially holds. □

A.3 Proof of Proposition 1

Before the proof, we first show the following.

Lemma 7. *Given a real number $\alpha > 0.5$ and two positive integers m and n where $m > n$, then $2\lceil \alpha m \rceil - m \geq 2\lceil \alpha n \rceil - n - 1$ holds.*

Proof. Since $m > n$, we assume that $m = n + k$ where k is a positive integer. To prove $2\lceil \alpha m \rceil - m \geq 2\lceil \alpha n \rceil - n - 1$, we can prove that $2\lceil \alpha n + \alpha k \rceil - 2\lceil \alpha n \rceil \geq k - 1$. It is easy to verify that either $\lceil \alpha n + \alpha k \rceil = \lceil \alpha n \rceil + \lceil \alpha k \rceil$ or $\lceil \alpha n + \alpha k \rceil = \lceil \alpha n \rceil + \lceil \alpha k \rceil - 1$. Below, we will prove that $2\lceil \alpha n + \alpha k \rceil - 2\lceil \alpha n \rceil \geq k - 1$ always holds no matter $\lceil \alpha n + \alpha k \rceil = \lceil \alpha n \rceil + \lceil \alpha k \rceil$ or $\lceil \alpha n + \alpha k \rceil = \lceil \alpha n \rceil + \lceil \alpha k \rceil - 1$.
Case 1: $\lceil \alpha n + \alpha k \rceil = \lceil \alpha n \rceil + \lceil \alpha k \rceil$. We can derive

$$2\lceil \alpha n + \alpha k \rceil - 2\lceil \alpha n \rceil = 2\lceil \alpha k \rceil \geq 2\alpha k > k > k - 1. \tag{2}$$

Case 2: $\lceil \alpha n + \alpha k \rceil = \lceil \alpha n \rceil + \lceil \alpha k \rceil - 1$. First, we know that

$$2\lceil \alpha n + \alpha k \rceil - 2\lceil \alpha n \rceil = 2\lceil \alpha k \rceil - 2 \tag{3}$$

Then, since $\alpha > 0.5$ and $k \geq 1$, we can derive

$$\lceil 2\alpha k \rceil - k = \lceil 2\alpha k - k \rceil = \lceil (2\alpha - 1)k \rceil \geq 1 \tag{4}$$

$$\frac{k+1}{2} \leq \frac{\lceil 2\alpha k \rceil}{2} \leq \left\lceil \frac{2\alpha k}{2} \right\rceil = \lceil \alpha k \rceil \tag{5}$$

$$2\lceil \alpha k \rceil - 2 \geq k - 1 \tag{6}$$

Thus, the proof is complete. □

Proof (Proof of Proposition 1). We prove this by contradiction. Assume that vertices u_1 and u_2 both exist in an (α, β)-quasi biclique $G([X \cup Y])$ with threshold θ_R. Then, we know that $\delta_G(u_1, Y) \geq \lceil \alpha \cdot |Y| \rceil$, so equivalently $|Y| - \delta_G(u_1, Y) \leq |Y| - \lceil \alpha \cdot |Y| \rceil$. Similarly, vertex u_2 satisfies the same condition. Then, we can derive that the number of non-common neighbors is $|Y| - \Delta(u_1, u_2) \leq 2|Y| - 2\lceil \alpha \cdot |Y| \rceil$, hence $\Delta(u_1, u_2) \geq 2\lceil \alpha \cdot |Y| \rceil - |Y|$. By Lemma 7, when $|Y| \geq \theta_R$, we have $\Delta(u_1, u_2) \geq 2\lceil \alpha \cdot \theta_R \rceil - \theta_R - 1$, which contradicts the assumption. Therefore, we know that such vertices cannot coexist in an (α, β)-quasi biclique with threshold θ_R. □

A.4 Proof of Proposition 2

Proof. According to the construction methods of G_L and G_R, an edge exists between two vertices if and only if they can coexist in an (α, β)-quasi biclique. Moreover, it is straightforward to prove that $\mathcal{C}(v) + 1$ represents the size of the largest (α, β)-quasi biclique that vertex v can participate in. Thus, if $\mathcal{C}(v) < \theta_L - 1$, it is easy to see that this vertex cannot contribute to forming an (α, β)-quasi biclique of size at least θ_L. □

A.5 Proof of Lemma 3

Proof. It is obvious that any sub-branch generated from the current branch must include the current selected set; thus, the global degree of the vertices in the selected set determines the maximum size of the (α, β)-quasi biclique that this branch can achieve. Specifically, it depends on the vertex with the minimum degree in the selected set, i.e., $\min_{v \in S_L} \delta_G(v, S_R \cup C_R)$ and $\min_{v \in S_R} \delta(v, S_L \cup C_L)$. By the definition of (α, β)-quasi biclique, we know that UB_{dl} and UB_{dr} represent the sizes of the maximum (α, β)-quasi biclique obtainable on the left and right sides of the current branch, respectively. Thus, we prove this lemma. □

A.6 Proof of Lemma 4

Proof. According to the proof of Lemma 3, we know that UB_{dl} and UB_{dr} are the maximum sizes that can be achieved on the left and right sides of the current branch while satisfying the requirement of an (α, β)-quasi biclique. Furthermore, based on the definition of (α, β)-quasi biclique, it is clear that the maximum allowable number of missing vertices on the left and right sides of the current branch are UB_{ml} and UB_{mr}, respectively. Thus, this lemma is proven. □

A.7 Proof of Lemma 5

Proof. Since the lower bound on the right side (resp. left side) of the branch is $LB(B_R)$ (resp. $LB(B_L)$), this lemma trivially holds. □

A.8 Proof of Lemma 6

Proof. By Proposition 1 and definitions of $LB(\cdot)$, this lemma trivially holds. □

B Omitted Description of Our Baseline Method in Algorithm 1

We summarize our baseline method `ENUM` in Algorithm 1. In particular, we begin by initializing B^*, which keeps track of the vertex set of the maximum (α, β)-quasi biclique, to be empty (Line 1), and then reduce the input graph G to its $(\lceil \alpha \cdot \theta_R \rceil, \lceil \beta \cdot \theta_L \rceil)$-bicore according to Lemma 2 (Line 2). Subsequently, we

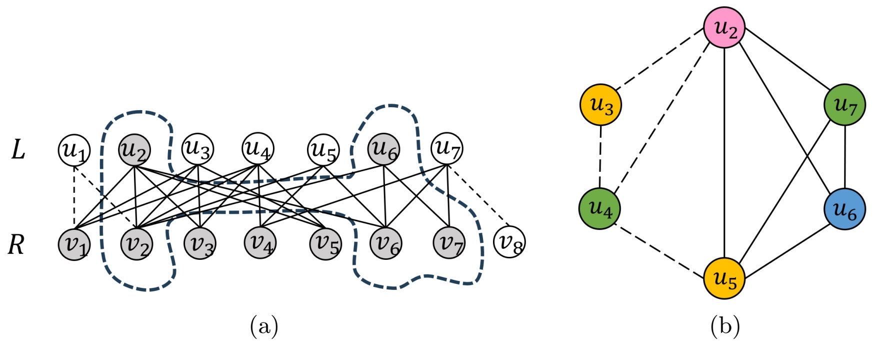


Fig. 3: Reduction process of a bipartite graph $G[L \cup R]$ using the graph preprocessing technique, and Figure 3(b) is the coloring graph G_L based on Figure 3(a).

invoke the procedure `Enum_Rec` to exhaustively search for the maximum (α, β)-quasi biclique in G (Line 3). Specifically, `Enum_Rec` can be divided into two processes: a recursion termination condition (Lines 5-8) and a binary branching process (Lines 9-11). If the size of the current graph does not exceed the recorded optimum value (i.e., $|B^*|$) or the given thresholds (i.e., θ_L and θ_R), we terminate the branch (Lines 5-6). Additionally, if the entire graph $G[S \cup C]$ is already an (α, β)-quasi biclique (Line 7), then we update the globally maintained B^* since a larger solution is found (Line 8). If a branch passes the recursion termination condition, we then proceed to the binary branching process to perform recursive search. The approach adopted by `ENUM` is to randomly select a vertex v from the candidate set $C = C_L \cup C_R$ (Line 9), generating two sub-branches (Lines 10-11). In one sub-branch, the branching vertex v is moved from the candidate set C to the selected set S (Line 10); while in the other sub-branch, v is removed from C (Line 11). Obviously, `ENUM` correctly finds the largest (α, β)-quasi biclique in G.

C Omitted Example

An Example for the Graph Coloring-based Technique. Proposition 2 presents a higher-order graph reduction method compared to the bicore reduction in Lemma 2. To facilitate understanding of this technique, we provide the following example. Figure 3 shows an example of the reduction process for solving the maximum $(0.7, 0.7)$-quasi biclique in a bipartite graph $G = (L \cup R, E)$ with thresholds $\theta_L = 4$ and $\theta_R = 3$, where $L = \{u_1, u_2, \cdots, u_7\}$ and $R = \{v_1, v_2, \cdots, v_8\}$. First, we reduce G to its $(3, 3)$-bicore based on Lemma 2, i.e., removing vertices u_1 and v_8 and their associated edges. Then, we calculate the number of common neighbors of vertices in L, and the coloring graph G_L is generated, as shown in Figure 3(b). According to Proposition 2, vertices with $\mathcal{C}(u) < \theta_L$ cannot form a solution, so u_3 and u_4 are removed, leaving the vertices shown in gray in Figure 3(a). Finally, the bipartite graph formed by the remaining gray vertices is reduced to its $(3, 3)$-bicore again, as indicated by the

Table 3: The size of remaining graph after preprocessing and the preprocessing time (pre-time) for different algorithms

Datasets	$\lvert L'\rvert$	$\lvert R'\rvert$	Pre-time in `MVQB/C`	$\lvert L''\rvert$	$\lvert R''\rvert$	Pre-time in `MVQB`
CF	163	163	0.01	163	163	0.01
HY	318	50	0.01	318	50	0.01
CM	452	592	0.01	412	534	0.02
WR	633	290	0.02	606	264	0.05
PR	518	1,120	0.03	112	200	0.08
LO	880	310	0.04	395	145	0.08
AC	45,787	28,473	0.15	531	328	2.36
IM	3,357	2,080	0.37	49	62	0.54
DB	772	936	1.20	416	494	1.51

dashed area in the figure. Note that the remaining L contains only two vertices, fewer than the required threshold θ_L, thus we can assert that G does not have any $(0.7, 0.7)$-quasi biclique with threshold $\theta_L = 4$. In other words, thanks to our proposed graph coloring-based preprocessing technique, we can obtain the solution only through preprocessing techniques, without entering the later more time-consuming branch-and-bound process.

D Additional Experimental Studies

We also provide the detailed preprocessing information in Table 3, where $|L'|$ and $|R'|$ (resp., $|L''|$ and $|R''|$) denote the number of remaining vertices after preprocessing by `MVQB\C` (resp., `MVQB`). From the table, we can draw the following conclusions. First, despite using more advanced graph preprocessing methods, the time spent on preprocessing by `MVQB` is very close to that of `MVQB\C` in the vast majority of cases. Second, with preprocessing times being similar, `MVQB` achieves significantly better graph preprocessing results than `MVQB\C`. Specifically, in the IM dataset, the sizes of the graph after the preprocessing step in `MVQB\C` are $|L'| = 3,357$ and $|R'| = 2,080$, while `MVQB` achieves an impressive reduction with $|L''| = 49$ and $|R''| = 62$.