

# On finding the maximum edge biclique in a bipartite graph: a subspace clustering approach

Eran Shaham \*

Honghai Yu \*

Xiao-Li Li \*

## Abstract

Bipartite graphs have been proven useful in modeling a wide range of relationship networks. Finding the maximum edge biclique within a bipartite graph is a well-known problem in graph theory and data mining, with numerous real-world applications across different domains. We propose a probabilistic algorithm for finding the maximum edge biclique using a Monte Carlo subspace clustering approach. Extensive experimentation with both artificial and real-world datasets shows that the algorithm is significantly better than the state-of-the-art technique. We prove that there are solid theoretical reasons for the algorithm’s efficacy that manifest in a polynomial complexity of time and space.

**Keywords**—*Maximum edge bipartite subgraph; Biclique; Subspace clustering; Data mining; Graph mining*

## 1 Introduction.

Biclique detection is a well-known problem in graph theory and data mining, with numerous real-world applications across different domains [3, 6, 9, 13, 15, 17, 18, 22, 23, 29]. Take for example text networks depicting the relationship between words and documents. Bicliques reveal documents that share common words, which can later be utilized for automated topic classification and tagging.

Given a bipartite graph and its corresponding partition into two disjoint sets of vertices, a *biclique* is a complete bipartite *subgraph* such that every vertex of the first partition is connected to every vertex of the second partition (see example in Figure 1, where a vertex set  $\{i_3, i_4, i_6\}$  and a vertex set  $\{j_3, j_5\}$  form a biclique). Mathematically, the notion of *biclique* is defined as follows.

**DEFINITION 1.** *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 such that  $\forall (i, j) \in E, i \in U, j \in V$ . A biclique within  $G$  is a couple (set pair)  $(I, J)$  such that  $I \subseteq U, J \subseteq V$  and  $\forall i \in I, j \in J, (i, j) \in E$ .*

The computational complexity of finding the maximum biclique depends on the exact objective function used. In contrast to the well-known MAXIMUM CLIQUE problem [12, 24], the maximum biclique problem has

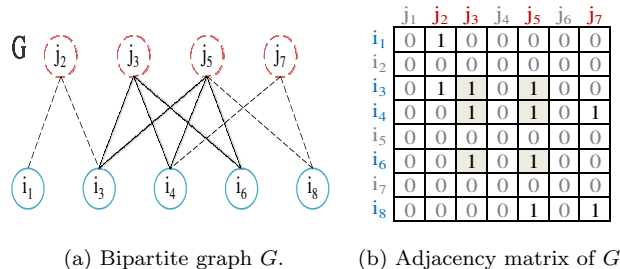


Figure 1: Bipartite graph  $G$  and its corresponding adjacency matrix, comprising the maximum edge biclique  $(\{i_3, i_4, i_6\}, \{j_3, j_5\})$  of size 6 edges and 5 vertices.

three distinct variants, with the following objective function  $\mu(I, J)$ :

- (1)  $\mu(I, J) = |I| + |J|$  — known as the MAXIMUM VERTEX BICLIQUE problem. The problem can be solved in polynomial time using a minimum cut algorithm [14].
- (2)  $\mu(I, J) = |I|$ , where  $|I| = |J|$  — known as the BALANCED COMPLETE BIPARTITE SUBGRAPH problem (also known as the BALANCED BICLIQUE problem). The problem was proved to be NP-complete [14].
- (3)  $\mu(I, J) = |I| \times |J|$  — known as the MAXIMUM EDGE BICLIQUE problem. The problem was proved to be NP-complete [11].

In this paper we focus on the problem of finding the *maximum edge biclique*. We propose an efficient Biclique Subspace Clustering (BSC) algorithm to tackle this challenging problem. Our extensive experimental results across artificial and real-world datasets demonstrate that our proposed BSC algorithm is significantly better than the state-of-the-art technique.

The remaining parts of the paper are organized as follows. In Section 2 we review related work. Section 3 introduces our proposed BSC algorithm, gives a proof of its optimality, and supplies run-time analysis. Next, Section 4 presents the extensive experiments conducted

\*Institute for Infocomm Research (I<sup>2</sup>R), A\*STAR, Singapore, {eran-shaham, yuhh, xll1i}@i2r.a-star.edu.sg.

and their detailed experimental results using both artificial and real-world data. Finally, we conclude the paper with a discussion and directions for future research in Section 5.

## 2 Related Work.

A wealth of research has been undertaken to study the MAXIMUM EDGE BICLIQUE problem. This research has received much attention due to its wide range of applications in areas such as bioinformatics [22], epidemiology [17], formal concept analysis [6], manufacturing problems [3], molecular biology [18], machine learning [15], management science [23], and conjunctive clustering [15].

Most of the algorithms tackling the problem can be divided into the following three main categories: (i) relaxation of the problem by bounding a characteristic of a graph; (ii) exploitation of some class of graph; and (iii) reduction to other domains. Among the algorithms in the first category we can find: bounding the biclique’s size [22], bounding the graph’s arboricity [4], and bounding the graph’s vertices degree [1]. Among the algorithms in the second category we can find: limitation to convex bipartite graphs [1, 7, 18], and limitation to chordal bipartite graphs [7]. Among the algorithms in the last category we can find: reduction to maximal clique [12, 24], and reduction to frequent itemsets [10, 25, 27].

The reduction of the problem to finding a maximal *clique* has the benefit of a well-researched field with an abundance of heuristics and approximation algorithms (recall that the problem is NP-complete). However, in order to perform such reduction, an inflation of the bipartite graph is needed. Such inflation is achieved by adding all possible edges between vertices of the same partition [12]. The edge inflation transformation makes the problem computationally impractical due to the large number of edges it adds. Consider  $G = (U \cup V, E)$  to be a bipartite graph with  $|U| = m$ ,  $|V| = n$ , and  $|E| = e$ . Inflating the bipartite graph results in a graph  $G' = (V', E')$ , where  $|V'| = |U| + |V| = m + n$  and  $|E'| = |E| + \binom{|U|}{2} + \binom{|V|}{2} = e + \binom{m}{2} + \binom{n}{2}$ . The “*fill*” of a graph (also known as “edge density”, or “connectance”) is defined as the proportion of the number of edges to the total number of possible edges [5]. Therefore, while  $fill(G) = \frac{e}{m \times n}$ , we get  $fill(G') = \frac{e + \binom{m}{2} + \binom{n}{2}}{\binom{m+n}{2}}$ . To illustrate how dramatic is the change, take for example the bipartite graph  $G$  with  $|U| = |V| = k$  (for some  $k \in \mathbb{N}$ ), and  $|E| = 0$  (i.e.,  $E = \emptyset$ , no edges). While the fill of  $G$  is zero, the asymptotic fill of  $G'$  is  $\sim 50\%$  ( $fill(G) = \frac{0}{k \times k} = 0$ ;  $fill(G') = \frac{0 + \binom{k}{2} + \binom{k}{2}}{\binom{k+k}{2}} = \frac{k^2 - k}{2k^2 - k} \underset{k \rightarrow \infty}{=} 0.5$ ).

The reduction of the problem to frequent itemsets benefits, as above, by relying on a rich field of research.

However, it contains other difficulties. Literature has shown [28] that a transactional database corresponds to a bipartite graph  $G = (U \cup V, E)$ , where  $U$  is the set of items (itemsets),  $V$  is the set of transactions (tids), and  $E$  is the set of pairs (item, transaction), i.e., an edge in the bipartite graph represents a transaction comprising the item. Take for example a set of products offered by a supermarket. A transaction would be a subset of products (itemset) purchased by a customer. Therefore, finding frequent itemsets corresponds to finding bicliques, and finding a maximal frequent closed itemset corresponds to finding a maximal biclique. However, transforming a frequent itemset to a biclique, although requiring a trivial post-processing step, can be highly time consuming [10]. Therefore, these reductions to other domains may present practical and scalability problems [29], and may not fully utilize the special characteristics of the bipartite graph.

Recently, Zhang et al. [29] introduced the iMBEA algorithm for the enumeration of maximal bicliques in a bipartite graph. The algorithm uses an efficient branch-and-bound technique to prune away non-maximal subtrees of the search tree. Despite the theoretical complexity of an exponential run-time, the algorithm outperforms the previous state-of-the-art algorithms: (i) MICA [1] – the best known general graph algorithm; and (ii) LCM-MBC [10] – a prime frequent closed itemsets algorithm (an improvement of LCM [25] algorithm). Zhang et al. [29] report in their paper that the iMBEA algorithm is consistently faster than the LCM-MBC algorithm on both artificial and real-world datasets. Furthermore, it was shown that the performance of the iMBEA algorithm is at least three orders of magnitude faster than the MICA algorithm.

## 3 Finding Maximum Biclique.

We are now ready to present the BSC algorithm. In Subsection 3.1 we describe a Monte Carlo algorithm for extracting a list of maximal bicliques. Next, we prove in Subsection 3.2 that the list contains, with fixed probability, an optimal biclique. Finally, we present in Subsection 3.3 the run-time analysis of the algorithm.

For ease of readability, we adopt the graph’s adjacency matrix representation, defined as follows (see the example in Figure 1b, which is the adjacency matrix representation of the bipartite graph  $G$  in Figure 1a).

**DEFINITION 2.** *Let  $G = (U \cup V, E)$  be a bipartite graph such that  $|U| = m$ , and  $|V| = n$ . The adjacency matrix  $X$  of graph  $G$  is a  $[m \times n]$  matrix such that  $X_{i,j} = 1$  if  $(i, j) \in E$  and  $X_{i,j} = 0$  otherwise.*

The input of the BSC algorithm is therefore an adjacency matrix  $X$  of a given bipartite graph  $G$ , consisting of only

---

**Algorithm 1** : BSC algorithm for finding maximal bicliques.

---

**Input:**  $X$ , a  $[m \times n]$  matrix of boolean numbers.  
**Output:** List of maximal bicliques.  
**Initialization:** Setting of  $N$  and  $|P|$  is thoroughly discussed in the following sections.

```

1: loop  $N$  times
2:   // Seeding phase
3:   choose a set of rows  $P$  uniformly at random;
4:   set  $I \leftarrow \emptyset, J \leftarrow \emptyset$ ;
5:   // Column addition phase
6:   foreach column  $j \leftarrow 1, n$  do
7:     if  $X_{P,j} = \mathbf{1}$  then
8:       add  $j$  to  $J$ ;
9:   // Row addition phase
10:  foreach row  $i \leftarrow 1, m$  do
11:    if  $X_{i,J} = \mathbf{1}$  then
12:      add  $i$  to  $I$ ;
13:  return list of  $(I, J)$ ;
```

---

boolean numbers, namely 0 and 1. The output of the BSC algorithm is a list of maximal bicliques, i.e., a list of submatrices of ones, representing maximal bicliques within  $G$  (the graph may contain multiple, possibly overlapping, maximal bicliques). The BSC algorithm itself uses a subspace clustering approach [11, 21]. This common technique uses iterative random projection (i.e., a Monte Carlo strategy) to obtain the biclique’s seed, which is later expanded into a maximal biclique.

**3.1 The BSC Algorithm.** Algorithm 1 presents the BSC algorithm (the Java code of the algorithm is *freely available* at the paper’s supporting webpage [20]). As in the case of many Monte Carlo algorithms, the structure of the BSC algorithm is very simple, and can be divided into the following four stages:

- (i) Seeding (lines 2-4): a random selection of a set of rows to serve as a seed of the maximal biclique.
- (ii) Addition of columns (lines 5-8): accumulating only columns that comply with the seed.
- (iii) Addition of rows (lines 9-12): this is similar to the previous stage (ii), but accumulating only those rows that comply with the columns accumulated in the previous stage (ii).
- (iv) Polynomial repetition (line 1): repetition of the above three steps provides a probabilistic guarantee to find a maximum biclique.

REMARK 1. The *Monte Carlo* nature of the BSC algorithm is revealed in phase (i) where random seeds are generated. The *subspace clustering* nature of the BSC

algorithm is revealed later, in phases (ii) and (iii), where the seed of phase (i) is expanded to form a maximal subset of columns over a maximal subset of rows, i.e., a maximal biclique.

REMARK 2. To ease readability, we present the BSC algorithm as first obtaining the columns (stage ii) and subsequently obtaining the rows (stage iii). However, the algorithm is symmetric, i.e., it can also be performed by first obtaining the rows and then obtaining the columns. In fact, running the algorithm in an interleaving fashion (i.e., alternating between first obtaining the rows, and first obtaining the columns) might help in finding imbalanced bicliques (see Theorem 3.2 for analysis of the probability of finding a biclique in relation to the dimensions of its rows and columns).

REMARK 3. Due to the Monte Carlo nature of the proposed BSC algorithm, its iterations (line 1) are independent of each other. The algorithm can therefore be implemented in an efficient distributed computing environment to take advantage of parallel computing or special hardware in a straightforward manner.

REMARK 4. To ease readability, lines 7 and 11 use the short notations of:  $X_{P,j} = \mathbf{1}$  and  $X_{i,J} = \mathbf{1}$ , respectively, which have the meaning of:  $\forall i \in P, X_{i,j} = 1$  and  $\forall j \in J, X_{i,j} = 1$ , respectively.

REMARK 5. The BSC algorithm has an inherent ability to mine multiple, possibly overlapping, bicliques by utilizing the independent random projection on each repetitive run, to reveal columns and rows relevant only to a specific biclique.

REMARK 6. The BSC algorithm is not designed for the enumeration of all maximal bicliques, which may be exponential in size [4, 29]. The algorithm has a polynomial number of iterations, and thus, the size of the return list is also polynomial. Next, we prove that the returned list contains, with fixed probability, an optimal biclique.

**3.2 Optimality of the Algorithm.** Clearly, the proposed BSC algorithm can be viewed as a heuristic method, and the extensive experimental evidence of Section 4 shows it to be efficacious in both artificial and real-world datasets. Next, we prove that there are solid theoretical reasons for this efficacy.

Usually, a bipartite graph (adjacency matrix) would contain many bicliques (submatrices of ones). From a practical point of view, an application would probably prefer a biclique with as many rows as possible (to increase the statistical relevance), and with as many

columns as possible (to increase the data correlation) [21]. However, most probably these objectives will conflict with each other. Take for example the extreme case of a biclique  $(I, J)$  with  $|I| = m$  (or,  $|J| = n$ ). The biclique would probably have a zero or very small size of  $J$  ( $I$ , respectively). This is due to the “curse of dimensionality” [8], where data tend to be sparse in high dimensional space. As such, the biclique would be far less useful as it would probably hold unimportant information regarding the data structure. The solution is therefore to define a *rank* which models the trade-off between the number of rows  $|I|$  and the number of columns  $|J|$  in the biclique. We adopt the following objective function as a rank measurement [2, 13, 16, 19, 21, 26]: the inclusion of an additional column in  $J$  is worth the exclusion of a  $(1-\gamma)$ -fraction of the rows in  $I$ .

At this point, we have all we need in order to formally define a *ranked biclique*. However, we extend the model to include two additional parameters,  $\alpha$  and  $\beta$ , that allow the user to specify the minimum dimensions of the mined biclique:  $\alpha$  - the minimum number of rows expressed as a fraction of  $m$ , and  $\beta$  - the minimum number of columns expressed as a fraction of  $n$ . If the user wishes not to constrain the biclique’s dimensions, trivial defaults can be used, i.e.,  $|I|, |J| = 1$ .

**DEFINITION 3.** *Let  $0 < \alpha, \beta, \gamma < 1$  be fixed parameters and  $|P|$  the size of the randomly selected seed rows. A ranked biclique of matrix  $X$  is a couple  $(I, J)$ , with  $I$  a subset of the rows and  $J$  a subset of the columns, that satisfies the following:*

- **Rank:** *The rank of a biclique  $(I, J)$  is defined as  $\mu(|I|, |J|)$ , where  $\mu(x, y) = x(1/\gamma)^y$ .*
- **Size:** *The number of rows is  $1 \leq \alpha m + |P| \leq |I|$  and the number of columns is  $1 \leq \beta n \leq |J|$ .*

Our problem thus turns into finding an optimal ranked biclique (or, *optimal* for brevity): a biclique of maximum rank that is larger than a given minimum size.

The main result of this subsection is Theorem 3.2, which states that if the BSC algorithm runs a polynomial number of iterations then we are guaranteed, with a fixed probability, to find an *optimal* biclique. The proof relies on the important insight of Procopiuc et al. [21] that in an optimal biclique there must be a relatively small subset of rows, of size  $\mathcal{O}(\log n)$ , which *determines* the participating columns of the biclique. The term of such a subset is a *discriminating set*, and is defined as follows.

**DEFINITION 4.** *Let  $(I, J)$  be a biclique. The set  $P \subseteq I$  is a discriminating set for  $J$  if it satisfies:*

$$(1) \forall j \in J, i \in P: X_{i,j} = 1 \text{ (or, } \forall j \in J, X_{P,j} = \mathbf{1}).$$

$$(2) \forall j \notin J, \exists i \in P: X_{i,j} = 0 \text{ (or, } \forall j \notin J, X_{P,j} \neq \mathbf{1}).$$

The structure of the proof is inspired by Procopiuc et al. [21] and consists of two major stages. First, we show that an *optimal* biclique can be located using a log-size discriminating set with a probability of at least 0.5. Based on this capability, we then show that when running the BSC algorithm in a polynomial number of iterations, it finds an optimal biclique with a probability of at least 0.5.

Next, we show that for an *optimal* biclique  $(I^*, J^*)$ , there are many small sets of size  $\mathcal{O}(\log n)$ , each of which is a discriminating set with a probability of at least 0.5. This result is important since upon selecting  $P \subseteq I^*$ , we can deduce  $J^*$  and subsequently  $I^*$ . Furthermore, the capability of finding a discriminating set with a probability of at least 0.5 is later used by Theorem 3.2 to find a biclique in a polynomial number of iterations.

**THEOREM 3.1.** *Let  $(I^*, J^*)$  be an optimal biclique. Any randomly chosen subset  $P \subseteq I^*$  of size  $|P| > \log(2n)/\log(1/\gamma)$ , is a discriminating set for  $J^*$ , with a probability of at least 0.5.*

*Proof.* We show that for any  $P$  that satisfies the above, condition (1) of Definition 4 always holds, and the probability that condition (2) does not hold is less than 0.5.

Condition (1) of Definition 4 is always satisfied, as  $P \subseteq I^*$ , and thus  $\forall j \in J^*, X_{P,j} = \mathbf{1}$  (line 7 in Algorithm 1).

Moving to condition (2) of Definition 4, we note that  $P$  fails to be a discriminating set for  $J^*$  only if there exists a column  $j \notin J^*$  such that  $X_{P,j} = \mathbf{1}$ . Next, we show that the probability of this happening for a *particular* column  $j$  is at most  $\gamma^{|P|}$ , and for *some* column  $j$  is at most  $n\gamma^{|P|} < 0.5$ .

**LEMMA 3.1.** *Let  $I \subseteq I^*$  and  $j \notin J^*$ . If  $X_{I,j} = \mathbf{1}$ , then  $|I| \leq \gamma|I^*|$ .*

*Proof.* The biclique  $(I, J)$ , with  $|I| > \gamma|I^*|$  and  $J = J^* \cup \{j\}$ , is a biclique satisfying  $\mu(I, J) > \mu(I^*, J^*)$ , contradicting the optimality of  $(I^*, J^*)$ . ■

Therefore, the probability of choosing all rows of  $P$  ( $P \subseteq I^*$ ) out of the set  $I$  of “bad rows” ( $I \subseteq I^*$ ) is bounded by:

$$\prod_{k=0}^{|P|-1} \frac{|I| - k}{|I^*| - k} < \left( \frac{|I|}{|I^*|} \right)^{|P|} \leq \left( \frac{\gamma|I^*|}{|I^*|} \right)^{|P|} = \gamma^{|P|}.$$

The above probability refers to a *particular* column  $j$ . Therefore, the probability that *some* column  $j \notin J^*$  will violate condition (2) of Definition 4 is bounded (after

substituting  $|P| > \log(2n)/\log(1/\gamma)$  by:  $(n-|J^*|)\gamma^{|P|} \leq (n-\beta n)\gamma^{|P|} = n(1-\beta)\gamma^{|P|} < n\gamma^{|P|} < 0.5$ . Note that as  $\beta$  decreases, the probability of a set to discriminate also decreases (while still being higher than 0.5), as more columns that do not belong to the biclique need to be filtered out, or vice versa.  $\square$

The outcome of Theorem 3.1 is important since upon selecting  $P \subseteq I^*$  we can deduce  $J^*$ . Moving to the second part of the proof, we show that when the BSC algorithm runs a polynomial number of iterations, it finds, with a probability of at least 0.5, an *optimal* biclique. We base this on Theorem 3.1, which shows the abundance of randomly selected discriminating sets of size  $\mathcal{O}(\log n)$  with a discriminating probability of at least 0.5.

**THEOREM 3.2.** *Let  $P$  be a discriminating set for the optimal biclique  $(I^*, J^*)$  and  $|I^*| \geq \alpha m + |P|$ . Provided  $N \geq 2 \ln 2/\alpha^{|P|}$ , the BSC algorithm will find the optimal biclique  $(I^*, J^*)$ , with a probability of at least 0.5.*

*Proof.* The probability of satisfying  $P \subseteq I^*$  is at least  $\alpha^{|P|}$  (given  $|I^*| \geq \alpha m + |P|$ ):

$$\prod_{k=0}^{|P|-1} \frac{|I^*| - k}{m - k} > \left( \frac{|I^*| - |P|}{m - |P|} \right)^{|P|} > \left( \frac{\alpha m}{m} \right)^{|P|} = \alpha^{|P|}.$$

Following Theorem 3.1, any given  $P \subseteq I^*$  is a discriminating set for  $J^*$  with a probability of at least 0.5. Therefore, the probability that all  $N$  iterations fail to find a discriminating rows set  $P$  (using the inequality  $(1 - 1/x)^x < 1/e$  for  $x \geq 1$ ) does not exceed  $(1 - 0.5\alpha^{|P|})^N < 0.5$ . It follows that the algorithm’s chances of finding a biclique upon  $P \subseteq I^*$  is at least 0.5. When it does, from the discriminating property of  $P$ , we obtain that  $J = J^*$ .

Next, we show that  $I = I^*$ . A row  $i$  is added to  $I$  only if (line 11 in Algorithm 1):  $X_{i,J} = \mathbf{1}$ . Since  $J = J^*$  and  $P \subseteq I^*$ , each  $i \in I^*$  would be added to  $I$ , namely  $I \supseteq I^*$ . From the optimality of  $(I^*, J^*)$  we obtain that  $I = I^*$  (similarly to Lemma 3.1 proof).  $\square$

The outcome of Theorems 3.1 and 3.2 is that the BSC algorithm’s chance of mining an optimal biclique increases as  $\beta$  and  $\alpha$  increases (see Theorems 3.1 and 3.2, respectively) which in turn requires a smaller number of iterations. The other side of the coin is that for small sized bicliques, the BSC algorithm needs a larger (while still polynomial) number of iterations. Also, Theorem 3.2 requires  $|I^*| \geq \alpha m + |P|$ , which constrains the optimal biclique to have a minimum  $\mathcal{O}(\log n)$  of rows. To tackle this limitation, we tend, in terms of future research, to integrate frequent itemsets techniques in order to efficiently obtain those small seeds.

**3.3 Run-time.** The inner for-loops take  $\mathcal{O}(mn)$  time. The total number of iterations is upper bounded by Theorem 3.2 as  $N = \mathcal{O}(1/\alpha^{|P|})$ , with  $|P|$  bounded by Theorem 3.1 to  $|P| = \mathcal{O}(\log(n)/\log(1/\gamma))$ . All in all, the run-time is *polynomial*:  $mn^{\mathcal{O}(1)}$ , for the constants  $\alpha$  and  $\gamma$  are independent of the matrix dimensions. Experiments reported in the next section show these bounds to in fact be largely pessimistic.

## 4 Experiments.

The previous section laid the theoretical foundations for the algorithm’s capability to efficiently mine *ranked bicliques*. This section demonstrates that the algorithm is also capable of efficiently mining *maximum edge bicliques*. The BSC algorithm is extensively evaluated and compared to the state-of-the-art algorithm. The experiments are divided into two sets. The first set of experiments uses artificial data in order to establish an easy-to-use setting of parameters and demonstrate the scalability of the BSC algorithm. The use of artificial data, which naturally enables tighter control, is important as it facilitates the examination of specific, isolated properties of the algorithm. The second set of experiments uses both artificial and real-world data. It aims to compare the performance of the BSC algorithm to the recent state-of-the-art iMBEA algorithm [29].

### Experimental Methodology.

We adopt the experimental design suggested by Zhang et al. [29] which makes use of the following two random bipartite graph models (the code for the random bipartite graph generators is freely available at the paper’s supporting webpage [20]).

- (a) Erdős-Rényi random bipartite graph model [5]: each possible edge in the bipartite graph has a probability  $\rho$ , independent of other edges, to be included in the graph. The probability  $\rho$  is also referred to as “density”.
- (b) A modified Erdős-Rényi random bipartite graph model that enables variability in the degree of vertices. Given a bipartite graph  $G = (U \cup V, E)$ , the degree of each vertex  $i \in U$  is governed by the following two parameters: (i)  $\mu$ : mean vertex degree; and (ii)  $CV$ : coefficient of variation. Note that:  $CV = \sigma/\mu$ , where  $\sigma$  is the standard deviation,  $\mu$  is the mean, and  $CV$  is the coefficient of variation. Also note that setting  $CV = 0$  is equivalent to using the previous Erdős-Rényi model as  $\mu = \rho$ .

The above degree places edges to every  $i \in U$ . The end point of each edge on  $V$  is selected with uniform probability. For example, if a vertex  $u \in U$  has been assigned a degree of two, then two vertices of  $V$ ,

$v_1, v_2 \in V$ , will be selected, uniformly at random, to serve as edge end points, namely:  $(u, v_1), (u, v_2) \in E$ .

The experiments were conducted using the platform 32x Intel® Xeon® @ 2.20GHz CPU with 128GB RAM, running Linux operating system. However, all algorithms were tested on a single core, i.e., no parallelism has been used.

**4.1 BSC: Properties of the Algorithm.** The advantage of using artificial data is that we have maximum control in verifying the validity of the bicliques found (in comparison to real-world data). Specifically, the contribution of the experimentation with artificial data is threefold. First, to establish a “best practice”, easy-to-use, setting of parameters. Secondly, to enable the verification of theoretical bounds and show that these bounds are coarse while better performance is achieved in practice. Finally, through the experiments, the actual run-time and scalability of the algorithm is demonstrated.

*Experiment I: Discriminating Probability.*

The discriminating set size,  $|P|$ , directly affects the run-time of the BSC algorithm. Theorem 3.1 provides us with the following bound:  $|P| > \log(2n)/\log(1/\gamma)$ , where  $\gamma$  is used to determine the rank of the biclique (see Definition 3). The bound depends on  $\gamma$ , a parameter of which the user has no knowledge. In order to get a sense of what the value of  $|P|$  is in practice, we conducted the following experiment. We first created random bipartite graphs (adjacency matrices) of  $|U| = 1000$  and  $|V| = 1000$ , with  $\rho \in \{0.2, 0.4, 0.6, 0.8\}$ , and  $CV \in \{0.0, 0.3, 0.6, 0.9, 1.2\}$ . We set the dimensions of the biclique size to  $\alpha, \beta \in \{0.2, 0.4, 0.6, 0.8\}$ . Then, we generated a random biclique within the specified dimensions and put it at a random location in the adjacency matrix, overriding the existing values. Then, a size -  $k$  subset of the biclique rows was chosen at random  $N = 100,000$  times, checked whether it discriminated (according to Definition 4), and what its discriminating probability was (i.e., in how many of the  $N$  times did the set actually discriminate). This process was repeated for  $k=1, \dots$  until reaching a value for  $k$  which the subset successfully discriminated in all of the  $N$  trials. Due to the random nature of the BSC algorithm, we repeated the experiment 10 times for each of the above configurations. Therefore, the experimentation results given hereafter report the average performance of the BSC algorithm. Due to space limitations, detailed quartile charts are given at the paper’s supporting webpage [20].

Figure 2 depicts the probability to discriminate as a function of the discriminating set size  $|P|$  across the parameters  $\beta$  and  $\rho$ . We present here only the results for

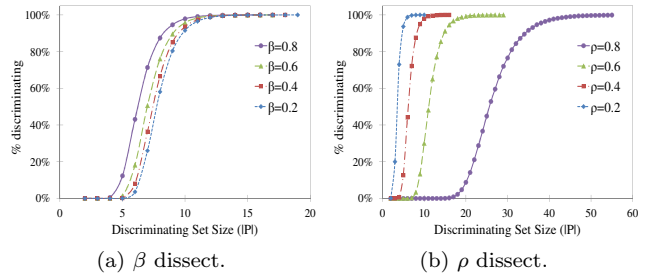


Figure 2: The probability to discriminate as a function of the discriminating set size  $|P|$ .

the configurations  $\{\alpha=0.6, \rho=0.4, CV=0.6\}$  and  $\{\alpha=0.8, \beta=0.8, CV=0.0\}$  for figures 2a and 2b, respectively, as the results for the other configurations exhibit similar patterns.

The following important observations can be made from the experiment: (i) smaller sizes of  $|P|$  can also have a high discriminating probability (e.g., Figure 2a: 92% for  $|P|=10$ ). Since  $|P|$  appears as an exponent in the estimated running time (see Subsection 3.3), choosing smaller  $|P|$  will reduce the run-time, without having a major negative effect on the results; (ii) a larger discriminating set is needed when the biclique has fewer columns or when the graph has a higher level of density (figures 2a and 2b, respectively). This is due to the fact that the discriminating set has to filter out more columns which do not belong to the biclique (see Theorem 3.1); and (iii) the parameters  $\alpha$  and  $CV$  have no effect on the discriminating probability.

*Experiment II: Discriminating Set Size.*

The previous experiment shows that using a smaller discriminating set will not only exponentially decrease the run-time, but will also ensure a reasonable discriminating probability. It also shows that the discriminating set size depends on the density of the graph  $\rho = \frac{|E|}{|U| \times |V|}$  ( $= \frac{\|X\|_0}{m \times n}$  using adjacency matrix notation), and on the ratio of the biclique columns  $\beta = \frac{|J|}{|V|}$  ( $= \frac{|J|}{n}$ ). In this experiment, we wish to provide an easy-to-use setting for the discriminating set size. The experiment was conducted in the same manner as the previous one, except for recording different sizes of random bipartite graphs (from small ones of  $[10 \times 10]$  to large ones of  $[100,000 \times 100,000]$ ), and using a discriminating probability of 50% (see Theorem 3.1).

Figure 3 depicts the relation between the discriminating set size  $|P|$ , the density of the graph  $\rho$ , and the various ratios of the biclique columns  $\beta$ . The exponential fittings of  $|P| = c_1 e^{c_2 \rho}$  for the ratios  $\beta = \{0.2, 0.4, 0.5, 0.6, 0.8\}$  are:  $|P| = \{0.50e^{4.68\rho}, 0.36e^{4.95\rho}, 0.20e^{5.57\rho}, 0.13e^{5.96\rho}, 0.07e^{6.35\rho}\}$  ( $R^2 = \{0.98, 0.99, 0.98, 0.98, 0.97\}$ ), respectively.



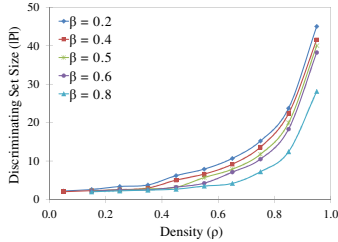


Figure 3: Discriminating set size  $|P|$  vs. the density  $\rho$ .

Moreover, the constants  $c_1$  and  $c_2$  are linearly dependent in  $\beta$ :  $c_1 = 0.64 - 0.77\beta$  ( $R^2 = 0.93$ ), and  $c_2 = 4.0 + 3.0\beta$  ( $R^2 = 0.95$ ). An important finding from the experiment is the obtainment of an easy-to-use,  $\gamma$  free, formula for setting  $|P|$ :  $|P| = (0.6 - 0.8\beta)e^{(4+3\beta)\rho}$ . While the density of the graph can be easily calculated, the biclique to be mined is unknown, and thus also the ratio  $\beta$  is unknown. Nevertheless,  $\beta$  can be set according to a user’s prior knowledge or randomly guessed, i.e.,  $\beta \in [0.0, 1.0]$ .

#### Experiment III: Number of Iterations.

Theorem 3.2 provides us with the following bound on the number of iterations:  $N \geq 2 \ln 2 / \alpha^{|P|}$ . While the previous experiment supplies an easy-to-use formula for setting  $|P|$ , the bound is still hard to set as  $\alpha$  ( $= \frac{|L|}{m}$ ) is unknown. In this experiment, we wish to provide an easy-to-use setting for the number of iterations. The experiment was conducted in the same manner as the previous one, except for setting  $N = mn$  and recording the number of iterations and coverage of the explored planted biclique.

An important finding from the experiment is that the actual average number of iterations needed is 0.98% of the set  $N = mn$  ( $\sigma=4.44\%$ ), with an average coverage of 99.71% ( $\sigma=3.04\%$ ). Thus, an easy-to-use setting of the number of iterations to 1% of the graph size would provide the desirable high coverage.

#### Experiment IV: Scalability.

Subsection 3.3 provides us with the following *polynomial* run-time bound:  $mn^{\mathcal{O}(1)}$ . The aim of this experiment is to examine the actual run-time for various inputs sizes. The experiment was conducted in the same manner as the previous one, except for recording the actual run-time till exploration of the planted biclique.

Figure 4 depicts the run-time of the BSC algorithm to  $|U| \times |V|$ . The important finding from the experiment is that the BSC algorithm is *linear* in the graph size. The results corroborate the feasibility and scalability of the algorithm.

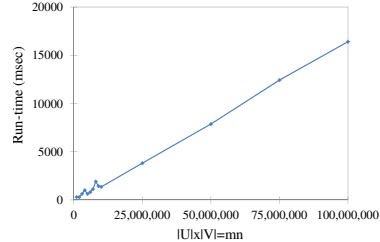


Figure 4: Scalability of the BSC algorithm to  $|U| \times |V|$ .

## 4.2 BSC vs iMBEA: Performance Comparison.

The experiments of the previous section yield promising results which demonstrate the capability and scalability of the BSC algorithm. The experiments in this section aim to compare the performance of the BSC algorithm to the recent state-of-the-art algorithm on both artificial and real-world data. Doing so will demonstrate the capability of the BSC algorithm to efficiently find the maximum edge biclique for a wide range of domains and graph characteristics. To achieve that, we compare the BSC algorithm to the recent state-of-the-art iMBEA algorithm [29],<sup>1</sup> which significantly outperforms the previous state-of-the-art MICA[1] and LCM-MBC[10] algorithms, on both artificial and real-world data.

The comparison experiments are divided into two sets. The first set of experiments uses artificial data, while the second set of experiments uses real-world data. The first set of experiments were conducted in the same manner as the artificial experiments in the previous subsection. To facilitate a fair comparison, we compared the outcome of the algorithms at the end of a one-hour run-time. The second set of experiments uses 10 real-world datasets encompassing a wide spectrum of domains with diverse bipartite graph properties.

#### Experiment V: Comparison on Density and CV.

The goal of the experiment is to compare the performance of the BSC and iMBEA algorithms by means of change in density and CV. The experiment was conducted in the same manner as the previous one, except for setting  $|U| = 1000$ ,  $|V| = 1000$ , and holding  $CV = 0.0$  for the density examination and  $\rho = 0.5$  for the CV examination.

Figures 5 and 6 depict the relationship between the run-time and coverage of the iMBEA and BSC algorithms as a function of the bipartite graph density and CV, respectively. The important finding from the experiment is the superiority of the BSC algorithm over the iMBEA algorithm for the entire range of densities and

<sup>1</sup>Zhang et al. [29] presented two algorithms: MBEA and iMBEA. We present here only a comparison with iMBEA as the authors have shown the superiority of iMBEA over MBEA on both artificial and real-world data. The authors have graciously provided us with their implementation of the algorithm.

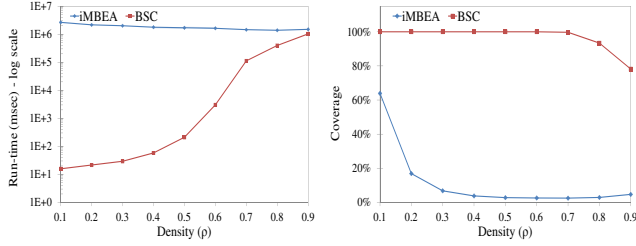


Figure 5: BSC and iMBEA run-time and coverage performance comparison as a function of density.

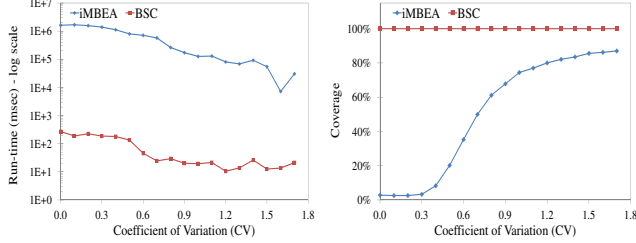


Figure 6: BSC and iMBEA run-time and coverage performance comparison as a function of  $CV$ .

$CV$ , with an average run-time improvement of 4.6 and 3.9 orders of magnitude for figures 5 and 6, respectively, and an average coverage improvement of 1.4 and 1.0 orders of magnitude for figures 5 and 6, respectively.

#### Experiment VII: Performance Comparison with Real-world Datasets.

The following set of experiments use 10 real-world datasets comprising a wide spectrum of domains and diverse bipartite graph properties. The datasets were taken from the KONECT repository [9] and their properties are summarized in Table 1 (sorted by ascending order of  $|E|$ ). As the ground-truth (i.e., the maximum edge biclique) is unknown, we conducted the experiment in the following way. The iMBEA algorithm was run till completion while recording the maximum edge biclique

Table 1: Real-world Datasets.

Dataset Name	$ E $	$ U $	$ V $
MovieLens 100k	100,000	943	1,682
YouTube	293,360	94,238	30,087
Location	293,697	172,091	53,407
GitHub	440,237	56,519	120,867
Wikiquote (en)	549,210	21,607	94,756
Wikinews (en)	901,416	10,764	163,008
MovieLens 1M	1,000,209	6,040	3,706
Wikibooks (en)	1,164,576	32,583	134,942
VisualizeUs	2,298,816	17,122	82,035
MovieLens 10M	10,000,054	69,878	10,677

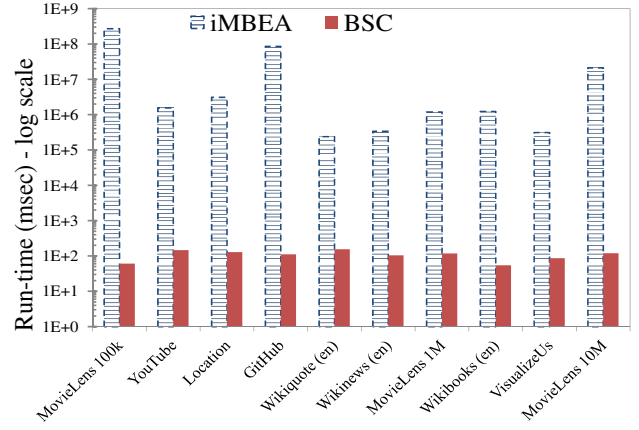


Figure 7: Run-time of the iMBEA and BSC algorithms on various real-world datasets.

exploration time and dimensions, which now serve as the ground-truth. Next, the BSC algorithm was run while recording the exploration time of the ground-truth. As in previous experiments, the run of the BSC algorithm was repeated 10 times.

Figure 7 depicts the run-time of the iMBEA and BSC algorithms on the various real-world datasets listed in Table 1. The important finding from the experiment is the superiority of the BSC algorithm over the iMBEA algorithm for the entire range of datasets, with an average run-time improvement of 4.5 orders of magnitude.

## 5 Discussion, Conclusions and Future Work.

The MAXIMUM EDGE BICLIQUE problem is well known in graph theory and data mining, with numerous real-world applications across different domains. Its importance is reflected in the vast body of prior literature.

The paper presents the BSC algorithm, a probabilistic method for finding the maximum edge biclique using a Monte Carlo subspace clustering approach. While the BSC algorithm can be viewed as an effective heuristic, we show that there is a strong theoretical base for its efficacy. The theoretical analysis promises a fixed probability of finding an optimal biclique (in the sense of balancing the number of rows and the number of columns), while giving a polynomial bound to the complexity of time and space. Extensive experimentation of the BSC algorithm with artificial data provides a “best practice”, easy-to-use setting of parameters, which present a significant improvement over the theoretical bounds. Furthermore, comprehensive experimentation with both artificial and real-world datasets shows that the algorithm is significantly better than the state-of-the-art technique. The results corroborate the usability, scalability and feasibility of the BSC algorithm.



The task of finding small sized bicliques requires a relatively large (while still polynomial) number of iterations. This is due to the relatively low probability of finding good seeds to serve as discriminating sets. Techniques of frequent itemsets could assist, by finding these good seeds. We believe there is far more to be explored in combining the methods of frequent itemsets and subspace clustering in terms of future research.

## References

- [1] G. ALEXE, S. ALEXE, Y. CRAMA, S. FOLDES, P. L. HAMMER, AND B. SIMEONE, *Consensus algorithms for the generation of all maximal bicliques*, *Discrete Applied Mathematics*, 145 (2004), pp. 11–21.
- [2] F. ALQADAH AND R. BHATNAGAR, *An effective algorithm for mining 3-clusters in vertically partitioned data*, in *International Conference on Information and Knowledge Management*, ACM, 2008, pp. 1103–1112.
- [3] M. DAWANDE, P. KESKINOCAK, J. M. SWAMINATHAN, AND S. TAYUR, *On bipartite and multipartite clique problems*, *Journal of Algorithms*, 41 (2001), pp. 388–403.
- [4] D. EPPSTEIN, *Arboricity and bipartite subgraph listing algorithms*, *Information Processing Letters*, 51 (1994), pp. 207–211.
- [5] P. ERDŐS AND A. RÉNYI, *On random graphs I.*, *Publicationes Mathematicae*, 6 (1959), pp. 290–297.
- [6] B. GANTER AND R. WILLE, *Formal concept analysis*, vol. 284, Springer, 1999.
- [7] T. KLOKS AND D. KRATSCHE, *Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph*, *Information Processing Letters*, 55 (1995), pp. 11–16.
- [8] H. P. KRIEGEL, P. KRÖGER, AND A. ZIMEK, *Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering*, *Transactions on Knowledge Discovery from Data*, 3 (2009), pp. 1–58.
- [9] J. KUNEGIS, *KONECT – The Koblenz Network Collection*, in *International Conference on World Wide Web Companion*, 2013, pp. 1343–1350.
- [10] J. LI, G. LIU, H. LI, AND L. WONG, *Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: a one-to-one correspondence and mining algorithms*, *Transactions on Knowledge and Data Engineering*, 19 (2007), pp. 1625–1637.
- [11] S. LONARDI, W. SZPANKOWSKI, AND Q. YANG, *Finding biclusters by random projections*, *Theoretical Computer Science*, 368 (2006), pp. 217–230.
- [12] K. MAKINO AND T. UNO, *New algorithms for enumerating all maximal cliques*, in *Algorithm Theory*, vol. 9, 2004, pp. 260–272.
- [13] A. A. MELKMAN AND E. SHAHAM, *Sleeved CoClustering*, in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 635–640.
- [14] R. G. MICHAEL AND D. S. JOHNSON, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, 1979.
- [15] N. MISHRA, D. RON, AND R. SWAMINATHAN, *On finding large conjunctive clusters*, *Learning Theory and Kernel Machines*, (2003), pp. 448–462.
- [16] T. MURALI AND S. KASIF, *Extracting conserved gene expression motifs from gene expression data*, in *Pacific Symposium on Biocomputing*, vol. 8, 2003, pp. 77–88.
- [17] R. A. MUSHLIN, A. KERSHENBAUM, S. T. GALLAGHER, AND T. R. REBBECK, *A graph-theoretical approach for pattern discovery in epidemiological research*, *IBM systems journal*, 46 (2007), pp. 135–149.
- [18] D. NUSSBAUM, S. PU, J. R. SACK, T. UNO, AND H. ZARRABI-ZADEH, *Finding maximum edge bicliques in convex bipartite graphs*, *Computing and Combinatorics*, (2010), pp. 140–149.
- [19] C. F. OLSON AND H. J. LYONS, *Simple and efficient projective clustering*, in *International Conference on Knowledge Discovery and Information Retrieval*, 2010, pp. 45–55.
- [20] *Paper’s supporting webpage*. <https://www.dropbox.com/s/h8riaac5qdgsc9a/SDM16.ReadMe.txt>.
- [21] C. M. PROCOPIUC, M. JONES, P. K. AGARWAL, AND T. MURALI, *A Monte Carlo algorithm for fast projective clustering*, in *International Conference on Management of Data*, 2002, pp. 418–427.
- [22] M. J. SANDERSON, A. C. DRISKELL, R. H. REE, O. EULENSTEIN, AND S. LANGLEY, *Obtaining maximal concatenated phylogenetic data sets from large sequence databases*, *Molecular Biology and Evolution*, 20 (2003), pp. 1036–1042.
- [23] J. M. SWAMINATHAN AND S. R. TAYUR, *Managing broader product lines through delayed differentiation using vanilla boxes*, *Management Science*, 44 (1998), pp. 161–172.
- [24] E. TOMITA, A. TANAKA, AND H. TAKAHASHI, *The worst-case time complexity for generating all maximal cliques and computational experiments*, *Theoretical Computer Science*, 363 (2006), pp. 28–42.
- [25] T. UNO, M. KIYOMI, AND H. ARIMURA, *LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets*, in *ICDM Workshop on Frequent Itemset Mining Implementations*, vol. 126, 2004.
- [26] M. L. YIU AND N. MAMOULIS, *Iterative projected clustering by subspace mining*, *Transactions on Knowledge and Data Engineering*, 17 (2005), pp. 176–189.
- [27] M. J. ZAKI AND C. J. HSIAO, *CHARM: An efficient algorithm for closed itemset mining*, in *International Conference on Data Mining*, vol. 2, 2002, pp. 457–473.
- [28] M. J. ZAKI AND M. OGIHARA, *Theoretical foundations of association rules*, in *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1998, pp. 71–78.
- [29] Y. ZHANG, C. A. PHILLIPS, G. L. ROGERS, E. J. BAKER, E. J. CHESLER, AND M. A. LANGSTON, *On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types*, *BMC Bioinformatics*, 15 (2014), pp. 110–127.