This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix

Michael Raj Dominic Savio[1], Annamalai Sankar[2]
and **Nataraj Ramaiya Vijayarajan[3]**

[1]Department of Applied Mathematics & Computational Sciences,
PSG College of Technology, Peelamedu, Coimbatore 641004, Tamil Nadu, India;
E-mail: mds@ity.psgtech.ac.in

[2]Department of Computer Applications,
PSG College of Technology, Peelamedu, Coimbatore 641004, Tamil Nadu, India;
E-mail: dras@mca.psgtech.ac.in

[3]Department of Information Technology,
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India;
E-mail: RV.NATARAJ@bitsathy.ac.in

ABSTRACT

Several algorithms are available in the literature to find all maximal bicliques from an adjacency matrix. If these algorithms are applied directly on a symmetric adjacency matrix, all maximal bicliques will be generated twice. We propose a novel algorithm, S-Datapeeler, to enumerate all maximal bicliques only once in the context of 3-dimensional symmetric adjacency matrix i.e. zero duplicate patterns are generated. We have compared our results with DataPeeler in 3-D context. The DataPeeler algorithm generates all duplicate patterns, whereas the proposed algorithm completely eliminates all duplicate maximal bicliques. The proposed methodology results in 50% reduction in search space and thereby running time of S-Datapeeler is better for 3-D symmetric datasets.

Keywords: Data Mining, Maximal Bicliques, Symmetric matrix, Algorithms

1. INTRODUCTION

Real-life applications can be modeled by maximal bicliques. Online social networks have connected many people. The communities in these networks may be found by identifying maximal bicliques. The social network can be represented as a 2-dimensional dataset (Li et al., 2007). Interaction between the users is denoted by an edge and the interacting users are denoted by vertices. If one more dimension like week or month or year is added to the existing 2D dataset, that becomes 3-dimensional dataset, then we will be able to extract pattern of communication between these social communities with respect to time, which will be commercially more useful (Selvan et al., 2010). Similar maximal biclique pattern extractions can be performed on telecommunication network users also. Extensive study has been carried out to enumerate maximal bicliques from boolean adjacency matrix in DCI-Closed (Lucchese et al., 2006), DMiner (Besson et al., 2005), LCM (Uno et al., 2004), Bimax (Prelic et al., 2006), and Closet+ (Wang et al., 2003) algorithms. Madeira et al., 2004, discusses about biclustering algorithms to enumerate maximal bicliques from an adjacency matrix, where the dataset is in real value context that represent gene expression matrix. Enumerating maximal bicliques is a highly challenging task and the

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

running time of the algorithm increases exponentially with respect to the number of vertices (Peeters, 2003). Apriori Algorithm (Agrawal et al., 1994) and FP-Growth methodology (Han et al., 2004), find bicliques from a bipartite graph. DCI-Closed (Lucchese et al., 2006) algorithm enumerates all maximal bicliques from an adjacency matrix in a depth first manner. DataPeeler (Cerf et al., 2009) algorithm extracts all maximal bicliques from n-D dataset with user specified size constraints. LCM-MBC algorithm (Li et al., 2007) generates maximal bicliques twice from a symmetric matrix and discards the duplicates later. Twinblade algorithm (Savio et al., 2012) generates all the maximal bicliques only once but works only on 2-D symmetric dataset. Cubeminer (Selvan et al., 2010) algorithm was designed to find all maximal bicliques from an 3-D adjacency matrix. When the same algorithm is used for the 3-D symmetric adjacency matrix, it generates the maximal biclique patterns twice, and one among the two patterns is called as duplicate pattern. In order to remove these duplicate patterns, Cubeminer-MBC algorithm (Ji et al., 2006), that extends Cubeminer, introduced subtree pruning strategies which were based on the symmetric nature of the dataset. Even after introducing these pruning strategies, Cubeminer-MBC removed these duplicate patterns completely for selective datasets. For the rest of the datasets, it removed only to a certain extent. We have extended Datapeeler instead of Cubeminer, because Datapeeler algorithm's performance is better than Cubeminer. The proposed method extends the Datapeeler algorithm by combining a novel element selection technique and subtree pruning strategy. To our knowledge in the literature, no algorithm exists in 3-dimensional context to eliminate all duplicate patterns. Hence, we address the problem of mining maximal bicliques only once from 3-dimensional symmetric adjacency matrix with no self loops.

A dynamic convex hull based clustering algorithm (Theljani et al., 2013) is used for classification, which deals with the data that appear in sequential. An improved clustering algorithm (Yazdani et al., 2013) clusters the data based on particle swarm optimization technique. A novel GSA-based algorithm (Purcaru et al., 2013) generates an optimal path for a robot travelling in partially unknown environment in the presence of multiple obstacles and its performance is compared with a well-known particle swarm based algorithm. These algorithms either concentrate on clustering or classification. We concentrate on triclustering.

In this paper, we propose an efficient algorithm S-Datapeeler in 3-dimensional context, inspired by DataPeeler (Cerf et al., 2009), to enumerate all maximal bicliques without generating duplicates. The major challenge lies in the elimination of all duplicate patterns and it is accomplished with the help of a novel element selection technique. In 3-D context, we have compared our algorithm with DataPeeler (Cerf et al., 2009) algorithm on synthetic datasets.

The rest of the paper is organized as follows. Section 2 details the preliminaries and section 3 discusses the novel methodology adopted, algorithm and a working example. In section 4, we analyze the experimental results and section 5 concludes the paper.

2. PRELIMINARIES

In this section, we present the basic definitions required for further analysis of the paper, with the problem definition. Let A= (H, R, C) be a 3-dimensional symmetric matrix. Let C = {c_1, c_2, c_3, . . . , c_m} be set of

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

column vertices and R = {r_1, r_2, r_3, . . . , r_m} be set of row vertices and where $r_i = c_i$, $1 \le i \le m$. Let H= {h_1, h_2, . . . , h_s}, represents the height set or third dimension, where each h_i is a symmetric adjacency matrix (R, C) with diagonal elements '0', indicating there are no self loops. Each cell is represented by either '0' or '1', where '0' represents no interaction, and '1' represents interaction between the corresponding row vertex and the column vertex. Each adjacency matrix (R, C) represents an undirected graph with no self loops. Let |H| refers to cardinality of the H. We now provide the definitions of bicliques and maximal bicliques in 3-D context.

Definition 1. *Let (H, R, C) be a 3-dimensional symmetric matrix, where H represents the set of height elements, R represents the set of row elements, and C represents the set of column elements. Let (H_S, R_S, C_S) be a subset of (H, R, C) such that $H_S \subseteq H$, $R_S \subseteq R$ and $C_S \subseteq C$. (H_S, R_S, C_S) is defined as a 3-dimensional biclique iff $\forall h \in H_S$, $\forall r \in R_S$, $\forall c \in C_S$, h x r x c is '1' and $R_S \cap C_S$ is empty.*

In Table 1, we have an example of 3-D symmetric adjacency matrix (h_1, h_2, h_3: r_1, r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5). (h_1, h_2, h_3: r_2, r_5 : c_4) is a biclique, whereas (h_1, h_2, h_3: r_2, r_5 : c_3, c_4) is not a biclique, because h_3:r_5:c_3 contains '0'.

Definition 2. *Let (H_S, R_S, C_S) be a 3-dimensional biclique of a 3-dimensional matrix (H, R, C). A 3-dimensional biclique (H_S, R_S, C_S) is defined as a 3-dimensional maximal biclique, iff $\neg\exists$ $h \in H\backslash H_S$, such that (h, R_S, C_S) is a 3-D biclique, $\neg\exists$ $r \in R\backslash R_S$, such that (H_S, r, C_S) is a 3-D biclique, and $\neg\exists$ $c \in C\backslash C_S$, such that (H_S, R_S, c) is a 3-D biclique.*

In other words, a *3-dimensional biclique (H_S, R_S, C_S) is said to be maximal, iff $\neg\exists$ $h \in H\backslash H_S$, such that $\forall r \in R_S$, $\forall c \in C_S$, h x r x c is '1', $\neg\exists$ $r \in R\backslash R_S$, such that $\forall h \in H_S$, $\forall c \in C_S$, h x r x c is '1', and $\neg\exists$ $c \in C\backslash C_S$, such that $\forall h \in H_S$, $\forall r \in R_S$, h x r x c is '1'.* In simple, a 3-D biclique *(H_S, R_S, C_S)* is maximal, iff no proper superset of *(H_S, R_S, C_S)* is 3-D biclique. Continuing with the example, (h_1, h_2, h_3: r_1, r_2, r_5 : c_4) is a maximal biclique. (h_1, h_2 : r_1, r_2 : c_3, c_4) is a biclique, but not maximal because there is an element r_5 such that (h_1, h_2 : r_1, r_2, r_5: c_3, c_4) form a biclique. Since the input is a symmetric matrix, each maximal biclique will be generated twice. One among the two is a duplicate pattern. In this paper, we define a duplicate pattern as follows: a 3-D maximal biclique *(H_S, R_S, C_S)* is a duplicate pattern if minimum (R_S) > minimum (C_S). For example, (h_1, h_2 : r_3, r_4 : c_1, c_2, c_5) is a duplicate pattern of the maximal biclique (h_1, h_2 : r_1, r_2, r_5: c_3, c_4), since minimum(r_3, r_4) = r_3, minimum(c_1, c_2, c_5) = c_1, and $r_3 > c_1$.

In order to include user specified constraints, we include the following definition w.r.t. maximal biclique, with an assumption that p ≤ q always hold: *A 3-D maximal biclique (H_S, R_S, C_S) is said to be (w,p,q)-large if $|H_S| \ge w$, and $|R_S| \ge p$ or $|C_S| \ge p$, and the other is atleast q.*

Problem Definition. *Our problem is to enumerate all (w,p,q)-large maximal bicliques from 3-dimensional symmetric matrix without generating duplicates, where w, p and q are user defined constraints such that $|H_S| \ge w$, and $|R_S| \ge p$ or $|C_S| \ge p$, and the other is atleast q for any such 3-D maximal biclique (H_S, R_S, C_S).*

Here, we discuss briefly the working principle of Datapeeler, since S-Datapeeler extends it. It starts with a root node containing two n-sets (X, Y), where X is initially empty and Y contains all the n-sets. It generates a binary tree and explores it in DFS manner. The element p to be used for enumeration is selected from Y only if it forms a biclique with X. The elements of Y which failed to form a biclique during selection process are obviously removed from Y. After selecting the element p, the left child and right

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

child nodes are generated. The left child node is generated by appending p to X and removing p and *O* from Y, i.e., $(X \cup p, Y \setminus (p \cup O))$, where $O \in Y$ such that $(X \cup p \cup O)$ is not biclique. The right child node is generated as (X, Y\p). Maximality check and anti-monotone property is performed at every recursive call of the algorithm. Maximality check is performed using the stack S, which contains the selected element p associated with right branches of the path from the root node. For any node, if there is an element s in S, such that $(X \cup s \cup Y)$ is biclique, then the node is discarded. Anti-monotone property states that: if a biclique is not frequent, none of its superset is frequent. It is used to prune the search space, i.e., helps to enumerate *(w,p,q)-large* maximal bicliques only. If both maximality check and anti-monotone check are true, then it is output as a maximal biclique, provided Y is empty. For example let us consider the 3-D dataset given in Table 1. The root node is denoted by $(\varnothing:\varnothing:\varnothing)$ $(h_1, h_2, h_3 : r_1, r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5)$. If the selected element is h_1, then the left child node is $(h_1 :\varnothing:\varnothing)$ $(h_2, h_3 : r_1, r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5)$ and the right child node is $(\varnothing:\varnothing:\varnothing)(h_2, h_3 : r_1, r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5)$. In this way, it proceeds selecting elements one by one and arrives at finding all maximal bicliques. When the dataset is symmetric, if Datapeeler is applied, the maximal bicliques are generated twice. In order to generate the maximal bicliques only once, we discuss the new methodology in the next section in 3-D context.

Table 1: An Example for 3-D Symmetric Matrix

h_1

	c_1	c_2	c_3	c_4	c_5
r_1	0	0	1	1	0
r_2	0	0	1	1	0
r_3	1	1	0	0	1
r_4	1	1	0	0	1
r_5	0	0	1	1	0

h_2

	c_1	c_2	c_3	c_4	c_5
r_1	0	1	1	1	0
r_2	1	0	1	1	1
r_3	1	1	0	1	1
r_4	1	1	1	0	1
r_5	0	1	1	1	0

h_3

	c_1	c_2	c_3	c_4	c_5
r_1	0	1	0	1	1
r_2	1	0	1	1	1
r_3	0	1	0	1	0
r_4	1	1	1	0	1
r_5	1	1	0	1	0

3. 3-D MAXIMAL BICLIQUE ENUMERATION ALGORITHM

This section discusses a novel selection strategy to achieve zero duplicate enumeration adopted in S-Datapeeler. We strictly follow the elements in its lexicographic order while selecting the element particularly from R and C (Dias et al., 2005). While selecting the elements, we select row element and its corresponding column element alternatively on the right path only from the root. Height element is not

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

selected at all, on this right path only from the root. The maximal bicliques generated in the left subtree of column element selection are duplicates w.r.t. the maximal bicliques generated in the left subtree of row element selection. Hence, we do not generate or process the left subtree, whenever a column element is selected on the right path only from the root. Due to this subtree pruning technique, the complete set of duplicate maximal bicliques is eliminated. In the rest of the nodes, i.e., nodes other than right path only from the root, any element may be selected.

Consider the data given in Table 1 and Figure 1. (∅:∅:∅) (h_1, h_2, h_3: r_1, r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5) is the root node. We select the element r_1 to generate left child node (∅: r_1 :∅) (h_1, h_2, h_3: r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5) and the right child node (∅:∅:∅)(h_1, h_2, h_3: r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5) as shown in Figure 1. From the right node of the root, element c_1 is selected. The left subtree of the root will generate all maximal bicliques that contain r_1.

Figure 1 depicts the binary tree enumeration on the right path only by selecting the elements r_1, c_1, r_2, c_2, r_3, and c_3 in the order specified respectively. Whenever r_1, r_2 and r_3 are selected, we say odd left child nodes. When c_1, c_2, and c_3 are selected, we say even left child nodes. Here r_1=c_1, r_2=c_2 and r_3=c_3 respectively. The maximal bicliques enumerated from the even left child node will be the duplicate pattern of maximal bicliques enumerated from immediate previous odd left child node. The maximal bicliques enumerated from the even left child node when c_1 is selected will be the duplicate pattern of maximal bicliques enumerated from odd left child node when r_1 is selected. Obviously it holds for r_2 & c_2, and r_3 & c_3. Therefore, the crossed nodes are pruned, which leads to zero duplicate pattern generation. We denote maximal bicliques enumerated from odd left child node (encircled 1, 3 and 5 in Figure 1) by ODD_MBC(r_i) and maximal bicliques enumerated from even left child node (encircled 2, 4 and 6 in Figure 1) by EVEN_MBC(c_i) respectively, where 1≤ i ≤m.

Lemma 1. *Let (H, R, C) be a 3-dimensional symmetric matrix. Let* (∅:∅:∅)*(H : R : C) be the root node. If $r_i \in R$ is selected from the root to generate left and right child nodes, then all maximal bicliques of left subtree will contain r_i.*

Proof. Let (∅:∅:∅) (H : R : C) be a root node, where r_1, r_2, r_3, . . . , $r_m \in$ R be set of row vertices and c_1, c_2, c_3, . . . , $c_m \in$ C be set of column vertices. Since r_i is selected, left child node is generated as (∅: r_i : ∅) (H : R\$r_i$:C) and the right child node is generated as (∅:∅:∅)(H : R\$r_i$:C). If right child node is considered, right child node does not contain r_i, since it is removed. Therefore, the maximal bicliques generated from the right subtree will not contain r_i. Considering the left child node, whenever an element is added, it is going to prevail in the entire subtree. Therefore, all the maximal bicliques generated from the left subtree of the root will contain r_i. □

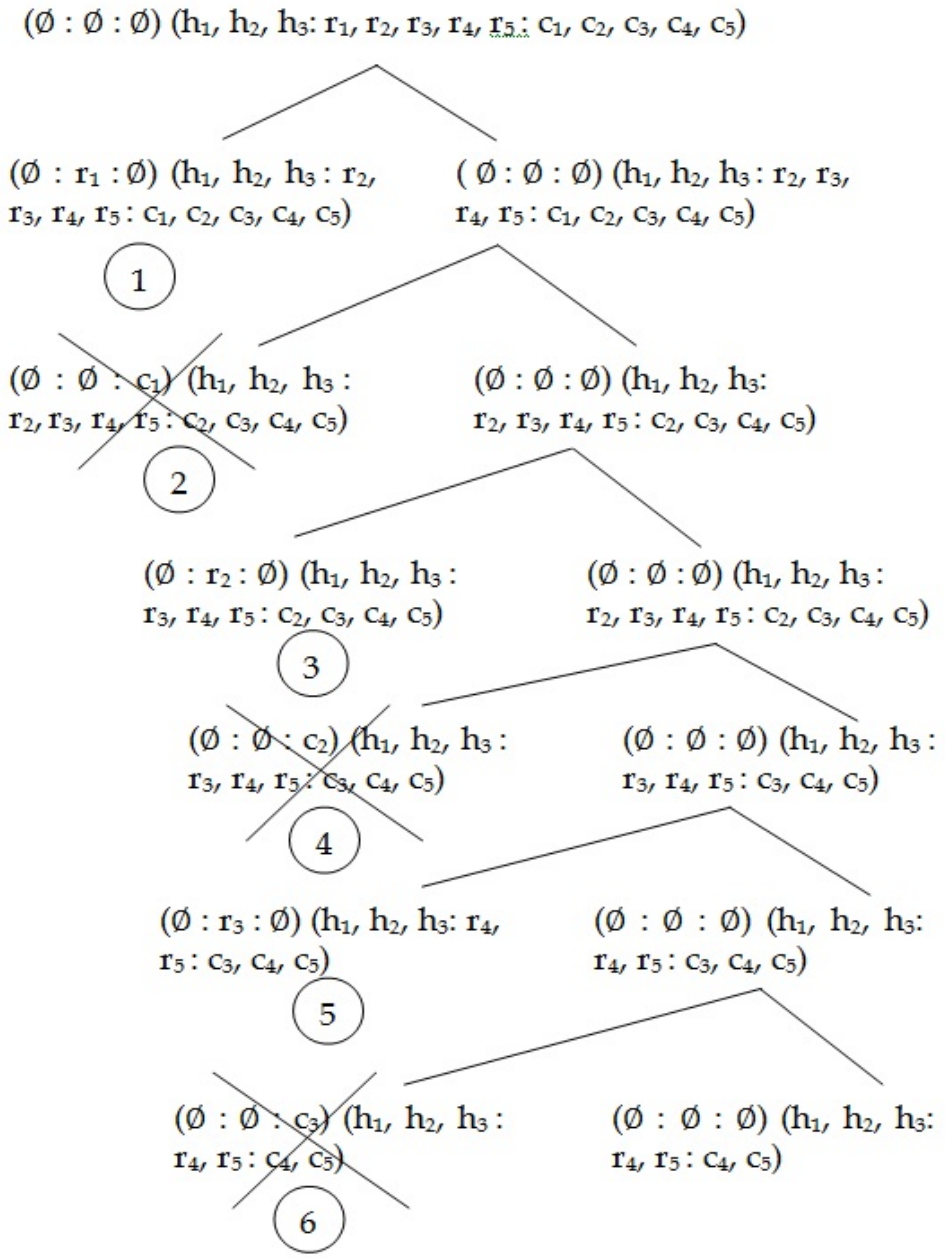
Consider the dataset given in Table 1. If r_1 is selected from the root node, maximal bicliques enumerated from the left subtree of the root will contain r_1. According to lemma 1, (h_1, h_2, h_3: r_1, r_2, r_5 : c_4), (h_1, h_2 : r_1, r_2, r_5: c_3, c_4), (h_2, h_3: r_1, r_3, r_5 : c_2, c_4) and (h_3: r_1, r_2 : c_4, c_5) are such maximal bicliques that contain r_1.

Lemma 2. *Let (H, R, C) be a 3-dimensional symmetric matrix. Let* (∅:∅:∅)*(H : R : C) be the root node. If $r_i \in R$ is selected from the root to generate left child node X_L and right child node X_R, and c_i is selected*

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

from X_R to generate its left child node X_{RL} and right child node X_{RR} respectively, then the maximal bicliques enumerated from subtree of X_{RL} are duplicates w.r.t. X_L.

Proof. Since r_i is selected from the root, as per lemma 1, all the maximal bicliques enumerated in the left subtree of root contain r_i. The right child node of the root node is X_R = (∅:∅:∅)(H : R\$r_i$: C). Now the column element c_i is selected from X_R, where $c_i = r_i$. Left child node of X_R is generated as X_{RL} = (∅:∅: c_i) (H : R\$r_i$: C\$c_i$), and the right child node as X_{RR} = (∅:∅:∅) (H : R\$r_i$: C\$c_i$). If X_{RR} is considered, it does not contain c_i, since it is removed. Therefore, the maximal bicliques enumerated from X_{RR} will not contain c_i. Now all the maximal bicliques enumerated on the left subtree of X_R will contain c_i, since the remaining elements are selected if only they form biclique with respect to c_i. Obviously, it will not contain r_i because it has already been removed. Hence, maximal bicliques enumerated from subtree of X_{RL} are duplicates w.r.t. X_L. □



This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

Figure1. Binary tree diagram depicting the element selection on the right path only from the root for the dataset given in Table 1, where alternatively r_i and c_i are selected where $r_i = c_i$, $1 \leq i \leq 3$

Consider the dataset given in Table 1. If c_1 is selected from the right child of the root node, maximal bicliques enumerated from the left subtree will contain c_1. (h_1, h_2, h_3: r_4 : c_1, c_2, c_5), (h_1, h_2 : r_3, r_4 : c_1, c_2, c_5), (h_2, h_3 : r_2, r_4 : c_1, c_3, c_5), and (h_3: r_4, r_5 : c_1, c_2) are such maximal bicliques that contain c_1. According to lemma 2, we can observe that these are duplicate patterns of maximal bicliques enumerated from the odd left subtree of the root where $r_1 = c_1$. Hence left child node is pruned when c_1 is selected.

Theorem 1. *Let (H, R, C) be a 3-dimensional symmetric matrix. Let* $(\varnothing:\varnothing:\varnothing)(H : R : C)$ *be the root node. Considering only the right path from the root, alternatively select* r_i *and* c_i*, where* $1 \leq i \leq m$ *and always* $r_i = c_i$*, to generate left child and right child nodes. Each* $r_i \in R$ *selected, leads to ODD_MBC(*r_i*) and each* $c_i \in A_1$ *leads to EVEN_MBC(*c_i*). EVEN_MBC(*c_i*) are duplicate patterns of ODD_MBC(*r_i*), where* $r_i = c_i$.

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

Proof. Let (∅:∅:∅)(H : R : C) be the root node. On the right path only from the root, alternatively r_i and c_i are selected in order to generate left and right nodes respectively, where each $r_i = c_i$, $1 \le i \le m$. Each $r_i \in$ R selected, leads to enumeration of maximal bicliques that contain r_i (according to lemma 1) i.e., ODD_MBC(r_i). Similarly, each $c_i \in$ C selected leads to enumeration of maximal bicliques that contain c_i i.e., EVEN_MBC(c_i). Since, alternatively r_i and c_i are selected and $r_i = c_i$, according to lemma 2, EVEN_MBC (c_i) are duplicate patterns of ODD_MBC(r_i), where $1 \le i \le m$. □

Consider the dataset given in Table 1. On the right path only if r_2 is selected, one of the maximal bicliques enumerated from the odd left subtree is (h_1, h_2, h_3: r_2 : c_3, c_4) and contains r_2. According to lemma 3, (h_1, h_2, h_3: r_3, r_4: c_2) is a duplicate pattern enumerated from the even left subtree, when c_2 is selected immediately on the right path only after r_2, where $c_2 = r_2$. Hence in our algorithm, left child node is pruned whenever a column element is selected on the right path only from the root. This leads to complete elimination of duplicate patterns.

3.1 Algorithm S-Datapeeler

INPUT: 3-dimensional symmetric adjacency matrix
OUTPUT: set of all 3-D maximal bicliques with zero duplicates

```
1.  X= null : null : null
2.  Y  = H : R : C
3.  S = null (refers to stack used for pruning bicliques that are subpatterns)
4.  Call S-Datapeeler(X, Y, S )

5.  S-Datapeeler(X, Y, S )
6.  {
7.      If ( X, Y) is maximal w.r.t. S AND ( X, Y)C
8.          If  Y is null
9.              Output X as 3-D maximal biclique
10.         Else
11.             While( ( X ∪ (e = select(Y)) ) is not biclique)
12.                 Y = Y \ e
13.             If !(X is null AND e ∈Y:C)  //theorem 1
14.                 S-Datapeeler(X ∪ e, Y \ (e ∪ O), S)
15.             S-Datapeeler(X, Y \ e, S ∪ e)
16. }
```

The S-Datapeeler algorithm starts with root node containing X as null and Y with entire 3-D symmetric dataset. It works in a depth first manner, where the expansion takes place in binary fashion. Initially, we describe how the *select(Y)* function works (line no. 11). The row and the column elements are chosen alternatively in lexicographic order from the root on its right path only, where each row element is equal to the consecutive column element. The rest of the node elements can be chosen from any dimension.

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

Whenever an element is chosen, it is verified whether it will form a biclique, if not, it is discarded or removed from *Y* (line no. 10). This process continues until a chosen element forms a biclique.

Whenever a recursive call is made, the (X, Y) is verified whether it is maximal with the help of the stack S, which contains all removed elements associated with right branches of the path from the root (line no. 7). If there exists at least one element *s* in stack S, such that ($X \cup \#s \cup Y$) form a biclique, then X is not maximal biclique. $(X, Y)_C$ is used to check whether the maximal biclique will be of *(w,p,q)-large* and if not, the node is pruned so that none of its superset is *(w,p,q)-large*, which takes care of antimonotone property. Otherwise, X is output as maximal biclique, provided Y needs to be empty (line nos. 8 & 9). User defined constraints i.e., *(w,p,q)-large check* is incorporated in the following way:

If (|X.H|+|Y.H| ≥ w)

If (((|X.R|+|Y.R| ≥ q) AND (|X.C|+|Y.C| ≥ p)) OR ((|X.R|+|Y.R| ≥ p) AND (|X.C|+|Y.C| ≥ q)))

//continue… check Y is null and proceed

Line nos. 14 & 15 are used to generate both left and right child nodes respectively. The left node is generated by appending *e* to X and removing *e* and *O* from Y, i.e., ($X \cup e$, $Y \setminus (e \cup O)$), where $O \in Y$ such that ($X \cup e \cup O$) is not biclique. Right child node is generated by keeping X as it is, and by removing the selected element from Y and the stack S is appended with the removed element. When the right path only from the root is considered, it is observed that the right node generation does not update X. X is null at the root node and hence, X is null at all the nodes which fall on right path only from the root. Line no. 13 does verification whether it is a right only path from the root by checking if X is null. Pruning of nodes is done as follows: if the selected element *e* belongs to the column set, and X being null, no left child node is generated, i.e., even left subtree of the nodes on the right path only from the root is not allowed to generate. This pruning has led to removal of all duplicate bicliques completely and is supported by theorem 1. Therefore, our algorithm enumerates all maximal bicliques with zero duplicate patterns.

3.2 Pseudo-code of Algorithm S-Datapeeler

A pseudo-code of implementation is provided below. Most of the parts are self explanatory and wherever necessary required comments are provided. A brief description is also provided immediately after the pseudo-code.

Node structure is defined as :

```
struct dnode   {
    int *rs_x;          int *cs_x;          int *hs_x;          // X part
    int *rs_y;          int *cs_y;          int *hs_y;          // Y part
    int *rstack;        int *cstack;        int *hstack;        // S part
    struct node *next;
};
```

Few declarations and initializations:

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.


```
int *hstack, *rstack, *cstack, *hs_x, *rs_x, *cs_x, *rs_y, *cs_y, *hs_y;
bool ***e;          // e is used to store the 3-dimensional matrix.
int ww, pp, qq;  // user defined constraints – got from the user at run time
int Maximalbicliquecount;
struct dnode *new;
struct dnode *STACK;
```

Initializations

```
*hs_y, *rs_y, *cs_y – contains all height, row, and column elements     (Y part)
*hs_x, *rs_x, *cs_x – contains null, null, null                         (X part)
*hstack, *rstack, *cstack - contains null, null, null                   (S part)
Maximalbicliquecount = 0
STACK – null
```

Step 1

Dynamically each(heights) 2-D symmetric matrix stored as text file is transformed into 3-D symmetric matrix (represent by e)

Step 2

Initially generate the right only path, and on this path store row selected left node alone in STACK. Whenever a column element is selected no left node is generated. (as shown in Figure 1)

```
ODD_flag = true
Do {
    If (ODD_flag == true)//row element is selected
    {
        r = minimum(*rs_y)
        //Generate left child node and push inside stack
        PUSH(*hs_x, *rs_x ∪ r, *cs_x, *hs_y, *rs_y \ r, *cs_y, Stack )  into STACK
        //Generate right child node by modifying its parent
        *rs_y = *rs_y \ r
        *rstack = *rstack ∪ r
        ODD_flag = false
    }
    Else  // column element is selected
    {
        c = minimum(*cs_y)
        //Left child node is not generated – subtree pruning strategy
        //Generate right child node by modifying its parent
        *cs_y = *cs_y \ c
        *cstack = *cstack ∪ c
        ODD_flag = true
    }
} Until(*cs_y is empty)
```

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

Step 3

POP top of STACK into *hs_x, *rs_x, *cs_x, *rs_y, *cs_y, *hs_y, *hstack, *rstack, *cstack

Step 4

```
while ( true )
{
    initially maximality check is done, and if it fails, goto POP_STACK
     //User specified constraint check
    If ( |*hs_x|+|*hs_y| ≥ ww)
        If ( ( ( |*rs_x|+|*rs_y| ≥ qq) AND ( |*cs_x|+|*cs_y| ≥ pp) ) OR
        ( ( |*rs_x|+|*rs_y| ≥ pp) AND ( |*cs_x|+|*cs_y| ≥ qq)))
            select an element from Y

    if no element is selected from Y, goto POP_STACK

    if height element h is selected from hs_y (provided hs_y is not empty)
    {
        //right node generation
        create a new node dynamically to store right – call it as new
        copy X into new.X //copy *hs_x, *rs_x, *cs_x  into new.*hs_x, new.*rs_x, new.*cs_x
        copy *hs_y \ h, *rs_y, *cs_y  into new. *hs_y, new.*rs_y, new.*cs_y
        copy *hstack ∪ h, *rstack, *cstack  into new.*hstack, new.*rstack, new.*cstack
        PUSH new into STACK

        //left node generation is got by modifying the some parts of node
        *hs_x = *hs_x ∪ h
        *hs_y = *hs_y \ h
        w.r.t. c ∈ *cs_y, if !( ∀r ∈ rs_x, h x r x c = true)  then *cs_y = *cs_y\c
    }
    else if row element r is selected from rs_y (provided rs_y is not empty)
    {
        //right node generation
        create a new node dynamically to store right – call it as new
        copy X into new.X
        copy *hs_, *rs_y \ r, *cs_y  into new. *hs_y, new.*rs_y, new.*cs_y
        copy *hstack, *rstack ∪ r, *cstack  into new.*hstack, new.*rstack, new.*cstack
        PUSH new into STACK
```

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.


```
        //left node generation is got by modifying some parts of the node
        *rs_x = *rs_x ∪ r ;    *rs_y = *rs_y \ r
        w.r.t. c ∈ *cs_y, if !( ∀h ∈ hs_x, h x r x c = true) then *cs_y = *cs_y\c
    }
    else if column element c is selected from cs_y (provided cs_y is not empty)
        move *cs_y into *cs_x
    else if (*cs_y empty)
    {
        OUTPUT *hs_x, *rs_x, *cs_x as MAXIMAL BICLIQUE
        Maximalbicliquecount++
        POP_STACK:   if(STACK  ! empty)
                        Copy top of STACK into *hs_x, *rs_x, *cs_x,
                        *rs_y, *cs_y, *hs_y, *hstack, *rstack,*cstack
                     else break;
    }
}
OUTPUT Maximalbicliquecount
```

The pseudo-code of the algorithm above shows the implementation details with computational parameters. The algorithm was implemented non-recursively using stack, i.e., DFS technique is adopted.

Initially the node structure and the computational parameters with their initial values are mentioned. In the node structure, hs - refers to height set, rs – row set, cs – column set, rstack – row stack, cstack – column stack, and hstack – height stack respectively. The inputs are given as follows:
X part : null - initially, Y part : contains all height elements, all row elements and all column elements which are stored as integers, and STACK part – initially NULL
ww – minimum height elements, pp & qq – minimum row and column elements or vice versa

Step 1

Each height represents a 2-D symmetric matrix, and is stored as a text file. Based on the number of heights, dynamically each text file (2-D symmetric matrix) is transformed into 3-D symmetric matrix (denoted by e).

Step 2

This step deals with our contribution of a novel element selection technique and new subtree pruning strategy. Novel element selection technique: On the right path only from the root, alternatively row element and its corresponding column element are selected from Y and added to X on the left node enumeration. New subtree pruning strategy: W.r.t. the novel element selection technique, when row element is selected, the left child node is enumerated and pushed inside STACK. Whenever column

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

element is selected, the left child node is pruned, i.e, not generated. These concepts are depicted by an example in Figure 1. These concepts have led to generation of zero duplicate patterns, thereby reduces the search space by 50%, which increases the performance in terms of running time of S-DataPeeler algorithm over the other.

Step 3

The STACK contains elements that are pushed during Step 2. The top of STACK is popped in to X, Y and stack part S.

Step 4

Initially maximality check is done with respect to the stack part S, and if it fails, top of STACK is popped and continued, and otherwise, user defined constraint check is performed and if it fails, top of STACK is popped and continued. Left node generation and right node generation pseudo-codes are written w.r.t. the element selected from Y part (either height or row or column element). If the column element of Y part is empty, it is reported as maximal biclique. Step 4 is continued until the STACK becomes empty. Obviously, the total numbers of patterns generated and the running time are calculated.

4. EXPERIMENTAL RESULTS AND ANALYSIS

We have implemented and compiled both Datapeeler and S-Datapeeler algorithms using 32-bit Microsoft VC++ compiler, with Windows 7 operating system, in 3 GB RAM and Intel Core i3 processor environment. The datasets were created with home grown random data generator program. The time complexity of the algorithm is directly proportional to the number of patterns generated. Maximal Biclique generation is known to be in the class of NP-Complete (Peeters, 2003).

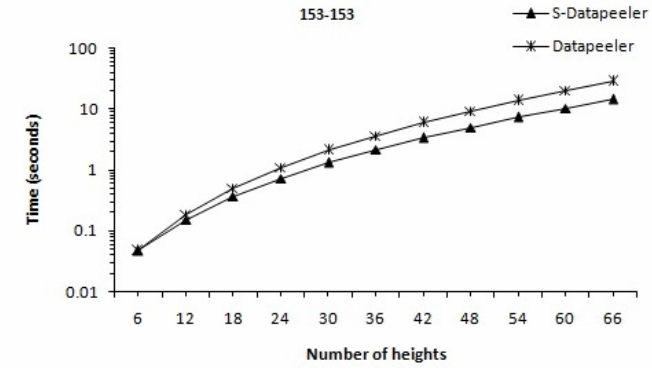


Figure 2. Comparing the running time of S-Datapeeler and Datapeeler with different heights from 6 to 66 and fixing each height with 153 x 153 symmetric dataset

The dataset for 153 people interacting with each other through an online social network have been generated. Density has been kept at 14.99% for each month (height). We make the number of heights

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

vary from 6 to 66, and every maximal biclique pattern is generated with constraints minimum 2 months and 2 persons with any other 2 persons. Comparison has been done between S-Datapeeler and Datapeeler algorithms and the results are shown in Figure 2. When the heights are less (for example 6) there is no difference in running time. Gradually if the heights are increased S-Datapeeler takes only half the running time that of Datapeeler. For example, considering the data for 54 months, Datapeeler takes 14.0 seconds to compute 45070 maximal biclique patterns, whereas, S-Datapeeler takes only 7.2 seconds to compute 22535 patterns.

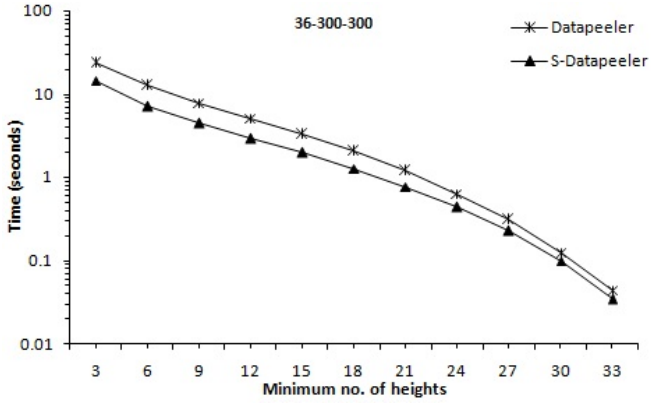


Figure 3. Comparing the running time of S-Datapeeler and Datapeeler with different minimal heights from 3 to 33 on 36 x 300 x 300 symmetric dataset

In the next synthetic dataset, we created 36 x 300 x 300 data with 300 persons interacting with each other on a social online network for 36 months. Each month (height) has a density of 14.99%. We make the minimum number of heights to vary from 3 to 33, and minimum 2 persons should have complete interaction with the other 2 persons. Comparison has been done between Datapeeler and S-Datapeeler, the running time was calculated and the results are shown in Figure 3. It is observed that, performance of S-Datapeeler is comparatively better than Datapeeler. For example, when the minimum height is set to 24, Datapeeler takes 0.633 seconds, while S-Datapeeler takes 0.437. When the minimum height is set to 3, S-Datapeeler takes 14.4 seconds to compute 914 maximal biclique patterns, whereas, the other takes 23.8 seconds to compute 1828 patterns.

We created a 333 x 33 x 33 symmetric matrix dataset representing 333 weeks, and 33 persons connected with one another via online social network. We made the density of connection per week to be 9.8%. We generated 3D maximal bicliques with user specified constraints as minimum 2 weeks and 2 persons should have complete communication with other 2 persons. It took 8.2 seconds to compute 68 patterns for S-Datapeeler algorithm, while Datapeeler took 22.2 seconds to compute 134 patterns.

It is clear that the Datapeeler always generates the pattern twice, due to the symmetric nature of the dataset. If p and q varies, the total number of patterns may be uneven. For example after setting w as 1, p as 1 and q as 2, on the dataset given in Table 1, S-Datapeeler generates 10 patterns while Datapeeler generates 19 patterns.

This article can be cited as M. R. D. Savio, A. Sankar and N. R. Vijayarajan, A Novel Enumeration Strategy of Maximal Bicliques from 3-Dimensional Symmetric Adjacency Matrix, International Journal of Artificial Intelligence, vol. 12, no. 2, pp. 42-56, 2014.

5. CONCLUSION

We have introduced a new algorithm S-Datapeeler, inspired by DataPeeler, to completely prune the duplicate maximal bicliques in 3-D symmetric context. Therefore, 100% elimination of duplicate patterns is achieved with the new element selection strategy proposed in this paper. This pruning has led to fifty percent reduction in search space, and thus performance of S-Datapeeler is better than Datapeeler symmetric datasets. In this paper, S-Datapeeler is discussed in 3-D context, which may also be extended in n-dimensional context.

REFERENCES

Agrawal R., Srikant R., Sept. 1994, *Fast algorithms for mining association rules*, Proceedings of International Conference on Very Large Data Bases, pp. 487-499, Santiago, Chile.

Besson J., Robardet C., Boulicaut J.F., Rome S., 2005 ,*Constraint Based Concept Mining and its Application to Microarray Data Analysis*, Journal of Intelligent Data Analysis, vol. 9, no. 1, pp. 59-82.

Cerf L., Besson J., Robardet C., Boulicaut J.F., 2009, *Closed patterns meet n-ary relations*, ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 1, pp. 1-36.

Dias V.M., de Figueiredo C.M., Szwarcfiter J.L., 2005, *Generating bicliques of a graph in lexicographic order*, Journal of Theoretical Computer Science, vol. 337, pp. 240–248.

Han J., Pei J., Yin Y., Mao R., 2004, *Mining Frequent Pattern without candidate Generation: A Frequent Pattern Approach*, Journal of Data Mining and Knowledge Discovery, vol. 8, no. 1, pp. 53-87.

Ji L., Tan K.L., Tung A.K.H., 2006, *Mining Frequent Closed Cubes in 3D datasets*, Proceedings of 32nd International Conference on Very Large Data-bases, pp. 811-822, Seoul, Korea.

Li J., Liu G., Li H., Wong L., 2007, *Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms*, IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 12, pp. 1625-1637.

Lucchese C., Orlando S., Perego R., 2006, *Fast and Memory Efficient Mining of Frequent Closed Itemsets*, IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 1, pp. 21-36.

Madeira S.C., Oliveira A.L., 2004, *Biclustering Algorithms for Biological Data Analysis: A Survey*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 1 no. 1, pp. 24-45.

Peeters R., 2003, *The maximum edge biclique problem is NP-complete*, Discrete Applied Mathematics, vol. 131, no. 3, pp. 651-654.

Prelic A., Bleuler S., Zimmermann P., Wille A., Buhlmann P., Gruissem W., Hennig L., Thiele L., and Zitzler E., 2006, *A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data*, Bioinformatics, vol. 22, no. 9, pp. 1122-1129.

Purcaru C., Precup R.-E., Iercan D., Fedorovici L.-O., David R.-C., and Dragan F., 2013, *Optimal robot path planning using gravitational search algorithm*, International Journal of Artificial Intelligence, vol. 10, no.S13, pp. 1-20.

Savio M.D., Sankar A., Nataraj R.V., Nov. 2012, *A Novel Algorithm to Enumerate Maximal Bicliques from a Symmetric Matrix*, Proc. of 3rd International Conference on Emerging Applications of Information Technology, Kolkata, India, pp. 456-467.

Selvan S., Nataraj R.V., 2010, *Efficient Mining of Large Maximal Bicliques from 3D Symmetric Adjacency Matrix*, IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 12, pp. 1797-1802.

Theljani F., Laabidi K., Zidi S., Ksouri M., 2013, *Convex hull based clustering algorithm*, International Journal of Artificial Intelligence, vol. 10, no. S13, pp. 51-70.

Uno T., Kiyomi M., and Arimura H., 2004, *LCM ver.2: Efficient mining algorithms for frequent/closed/maximal itemsets*, Proceedings of IEEE ICDM'04 Workshop FIMI'04, Brighton, UK.

Wang J., Han J., Pei J., 2003, *CLOSET+: Searching for the best strategies for mining frequent closed itemsets*, Proceedings of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 236-245, Washington, DC, USA.

Yazdani D., Saman B., Sepas-Moghaddam A., Mohammad-Kazemi F., Meybodi M.R., 2013, *A New Algorithm Based on Improved Artificial Fish Swarm Algorithm for Data Clustering*, International Journal of Artificial Intelligence, vol. 10, no. A13, pp. 193-221.