ONE TIME ENUMERATION OF MAXIMAL BICLIQUE PATTERNS FROM 3D SYMMETRIC MATRIX

[1]M DOMINIC SAVIO, [2]A SANKAR, [3]R V NATARAJ
[1]Department of Applied Mathematics and Computational Sciences, [2]Department of Computer Applications, [3]Department of Information Technology
[1,2]PSG College of Technology, [3]Bannari Amman Institute of Technology
[1,2]Coimbatore, [3]Sathyamangalam
INDIA
[1]mds@ity.psgtech.ac.in, [2]dras@mca.psgtech.ac.in, [3]rv.nataraj@bitsathy.ac.in

Abstract: - We propose an algorithm Cubeminer-MBC*, to extract maximal biclique patterns from a 3D symmetric adjacency matrix only once. In this paper, we introduce (i) a novel enumeration strategy and (ii) a new pruning strategy, which results 50% reduction in search space and maximal biclique patterns are generated only once, i.e., zero duplicates are generated. On the basis of experiments conducted, we observed Cubeminer-MBC* outperforms Cubeminer in terms of running time.

Key-Words: - Data Mining – Maximal Bicliques – Algorithms – Symmetric matrix – Duplicate pattern

1 Introduction

Finding web communities from online social networks can be modeled by maximal bicliques [1]. Online social networks have connected many people. Interacting people on social network is denoted by vertices and the interaction among them is denoted by edges. The social network can be represented as a 2-dimensional dataset in boolean context. If one more dimension like period (days, week, month or year) is added to the existing 2D dataset, that becomes 3-dimensional dataset, then we are able to extract pattern of communication between these social communities w.r.t. time, which would be commercially more useful [5]. Enumerating maximal bicliques is a highly challenging task and the running time of the algorithm increases exponentially with respect to the number of vertices [4]. The maximal biclique patterns occur twice in a symmetric adjacency matrix. Extensive study has been carried out to enumerate maximal bicliques only once in LCM-MBC [1] and Twinblade [6] algorithms on 2D symmetric adjacency dataset. In 3D symmetric context, Cubeminer-MBC [5] extends Cubeminer [3] to enumerate large maximal bicliques. With the help of a pruning strategy, it eliminates duplicate patterns to a certain extent, i.e., not all duplicate patterns are eliminated. Hence, we propose an efficient algorithm Cubeminer-MBC* in 3D symmetric context, to enumerate maximal biclique patterns only once, i.e., to eliminate duplicate patterns completely. We introduce a novel enumeration strategy and new pruning technique to achieve this. We have compared our algorithm with Cubeminer [3] on synthetic datasets.

Extensive study has been already carried out to enumerate maximal bicliques from boolean adjacency matrix in DCI-Closed [8], LCM [9], and Bimax [10] algorithms. Datapeeler [7] algorithm enumerates maximal biclique patterns in n-dimension context from an n-dimension dataset. Biclustering algorithms find maximal patterns from gene expression matrix [11], and work on real value datasets and not in boolean context.

These algorithms generally fit into any one of the following two straregies namely 1) grouping of ‘1’contained rows and columns and 2) eliminating ‘0’ contained rows and columns. Most of the algorithms like DCI-Closed [8], LCM[9], Bimax [10], Datapeeler [7], LCM-MBC [1], Twinblade [6], Apriori [12], FP-Growth [13] , and Closet+ [14], fall in the first category. DMiner [15] in 2D context, and Cubeminer[3] and Cubeminer-MBC[5] in 3D context fall in the second category. Our proposed algorithm Cubeminer-MBC* also fall in the second category. Anyhow, Bimax[10] algorithm alters row with some fellow rows, or columns with fellow columns so that finally maximal patterns are grouped.

The rest of the paper is organized as follows: section 2 presents the preliminaries, section 3 deals with the novel enumeration technique and pruning methodology adopted, section 4 analyzes the

experimental results and section 5 concludes the paper.

2 Preliminaries

In this section, we present the basic definitions with the problem definition. Let A= (H, R, C) be a 3D dataset. Let R = {r_1, r_2, r_3, . . . , r_m} be set of row vertices, and C = {c_1, c_2, c_3, . . . , c_m} be set of column vertices, where $r_i = c_i$, $1 \le i \le m$. Let H = {h_1, h_2, . . . , h_n}, 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' that represents no interaction, or '1' that represents interaction between the corresponding row vertex and column vertex. We now provide the definitions of bicliques and maximal bicliques in 3D context. *Let $H \subseteq$ H, $R \subseteq$ R and $C \subseteq$ C. (H, R, C) is said to be 3D biclique iff $\forall h \in H$, $\forall r \in R$, $\forall c \in C$, h x r x c is '1' and $R \cap C = null$.* In Table 1, we have an example of 3D 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'. *A 3D biclique (H, R, C) is said to be maximal, iff $\nexists h \in$ H\H, such that $\forall r \in R$, $\forall c \in C$, h x r x c is '1', $\nexists r \in$ R\R, such that $\forall h \in H$, $\forall c \in C$, h x r x c is '1', and $\nexists c \in$ C\C, such that $\forall h \in H, \forall r \in R$, h x r x c is '1'.* 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) forms a biclique. Biclique is also called as complete bipartite subgraph. Let $|R|$ refers to cardinality of the *R*.

The cutter is represented by (h', r', C') where C' is a zero contained columns w.r.t. height $h'$and row r'. (h_1 : r_1: c_1, c_2, c_5) is a cutter, since '0' contained columns are c_1, c_2 and c_5 w.r.t. h_1 and r_1. In symmetric context, each maximal biclique will be generated twice. One among the two is a duplicate pattern and defined as follows: *a 3D maximal biclique (H, R, C) is a duplicate pattern if min(R) > min(C). min(R)* denotes minimum element of row vertices. For example, (h_1, h_2, h_3: r_4 : c_1, c_2, c_5) is duplicate pattern of (h_1, h_2, h_3: r_1, r_2, r_5 : c_4). Readers may refer [5] for Cubeminer-MBC and [3] for Cubeminer algorithms respectively. In this paper, the problem is to enumerate all maximal biclique patterns only once from 3D symmetric matrix without generating duplicates.

Table 1
An Example for 3-D Symmetric Matrix

		c_1	c_2	c_3	c_4	c_5
h_1	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
		c_1	c_2	c_3	c_4	c_5
h_2	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
		c_1	c_2	c_3	c_4	c_5
h_3	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

This section discusses a novel enumeration strategy and a new pruning technique. We strictly follow the elements in its lexicographic order while processing the element from *H*, *R* and *C* [2]. Let *(H, R, C)* be any node. If (h', r', C') is a cutter, then the left son is generated as ($H\backslash h'$, *R*, *C*), middle as (*H*, $R\backslash r'$, *C*) and the right son as (*H*, *R*, $C\backslash C'$) [3] [5]. Consider the example dataset given in Table 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. (h_1: r_1 : c_1, c_2, c_5) is the cutter. Generally, the left node is generated as (h_2, h_3: r_1, r_2, r_3, r_4, r_5 : c_1, c_2, c_3, c_4, c_5), middle as (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 as (h_1, h_2, h_3: r_1, r_2, r_3, r_4, r_5 : c_3, c_4).

3.1 Novel Enumeration Technique

We do not use the subtree pruning of Cubeminer-MBC algorithm, instead introduce a new enumeration strategy to generate zero duplicate patterns. Let *(H, R, C)* be any node. Let us assume *min(R)* is same as *min(C).* Let $min(R)=r_i$ and $min(C)=c_i$, $1 \le i \le m$. Since $r_i=c_i$, there exists a cutter (h', r', C') such that $r_i=r'$ and $c_i \in C'$, for some $h \in H$, because h' x r_i x c_i contains '0'. While enumerating, additionally c_i is removed in both left and middle son. Obviously, c_i is removed in right son generation. Hence, the left son is generated as ($H\backslash h'$, *R*, $C\backslash min(C)$), middle as (*H*, $R\backslash r'$, $C\backslash min(C)$) and right son as (*H*, *R*, $C\backslash C'$), whenever *min(R) = min(C).* This new enumeration strategy takes care

of non occurrence of duplicate patterns and lemma 1 justifies it.

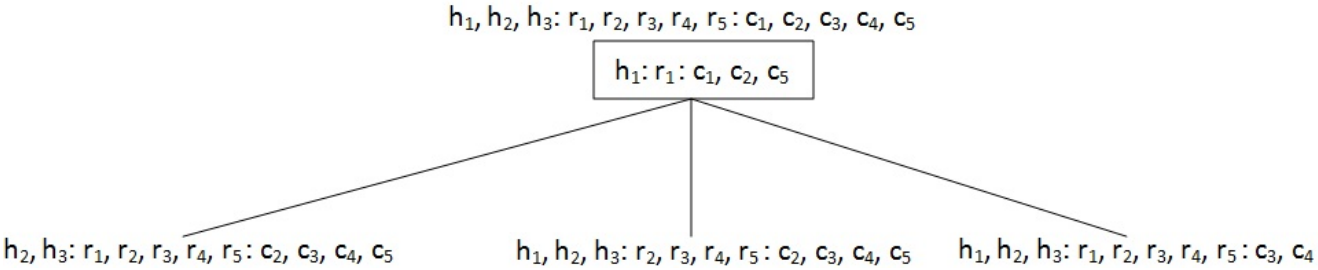

Fig. 1. Tree enumeration at th**e** root level that follows the novel enumeration technique

Lemma 1. *Let O = (H, R, C) be any node such that min(R) = min(C). Let there exists a cutter Z = (h′, r′, C′) such that, h′ ∈ H, min(R) ∈ r′, min(C) ∈ C′. If the cutter Z is applied on O to generate left son as (H\h′, R, C\min(C)) and middle son as (H, R\r′, C\min(C)), then it leads to enumeration of the patterns only with min(R) < min(C).*

Proof. Let *O = (H, R, C)* be any node such that *min(R) = min(C)*. Let there exists a cutter Z = *(h′, r′, C′)* such that, *h′∈H, min(R) ∈r′, min(C) ∈ C′*. The cutter Z is applied on *O* to generate left son as *LS = (H\h′, R, C\min(C))*, say *LS = (H_L, R_L, C_L)*, middle son as *MS = (H, R\r′, C\min(C))*, *say MS = (H_M, R_M, C_M)*, and right son as *RS = (H, R, C\C′), say RS = (H_R, R_R, C_R)*. Considering *LS*, *$min(R_L)$<$min(C_L)$*. Considering *RS*, *$min(R_R)$ < $min(C_R)$*. Considering *MS*, *$min(R_M)$=$min(C_M)$*, then there exists cutter *Z′ = (h′′, r′′, C′′)* such that left son and right son satisfy *min(R) < min(C)*, and middle son as *(H_M, R_M\r′′, C_M\min(C_M))*, *say* O_M = *(H_{M2}, R_{M2}, C_{M2})*. If there exists no more cutter w.r.t. *O_M*, then both *R_{M2}* and *C_{M2}* will be empty. Hence, it leads to enumeration of the patterns only with *min(R) < min(C)*. □

Considering Table 1 dataset, (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. According to lemma 1, (h_2, h_3: r_1, r_2, r_3, r_4, r_5 : c_2, c_3, c_4, c_5) is left son, (h_1, h_2, h_3: r_2, r_3, r_4, r_5 : c_2, c_3, c_4, c_5) is middle and (h_1, h_2, h_3: r_1, r_2, r_3, r_4, r_5 : c_3, c_4) the right son as shown in Fig. 1. min(r_1, r_2, r_3, r_4, r_5) = r_1, min(c_1, c_2, c_3, c_4, c_5) = c_1 and r_1 = c_1 . Hence, c_1 is removed both in left node and middle node w.r.t. the novel enumeration technique. On middle son again lemma 1 will be applied. In this way, no duplicate patterns are generated.

3.2 New Pruning Technique

The new enumeration strategy has led to the occurrence of some additional biclique patterns which are not maximal. These non-maximal biclique patterns occur as left/middle son of a node, and are not eliminated by left track check, right track check, closed height set check and closed row set check. For detailed information on left track check, right track check, closed height set check, closed row set check readers may refer [5] and [3] (Short discussions are provided in section 3.4). Hence, we introduce the new pruning technique, closed column set check in lemma 2. This check is similar to closed height set check/closed row set check, but w.r.t. column elements and performed on left/middle son of any node.

Lemma 2. *Closed Column Set Check: Let O_{LM} = (H_{LM}, R_{LM}, C_{LM}) be the left/middle son of any node O = (H, R, C). If ∃ c′ ∈ (C\C_{LM}) (C is the full column set), such that ∀ h′∈ H_{LM}, ∀ r′ ∈ R_{LM}, h′ x r′ x c′ is '1', then O_{LM} is not a maximal pattern in the column set and can be pruned off.*

Proof. Since ∃ *c′ ∈* (C \C_{LM}), such that ∀ *h′∈* H_{LM}, ∀ *r′∈* R_{LM}, *h′ x r′ x c′* is *'1'*, there exists O_S = *(H_{LM}, R_{LM}, C_{LM} ∪ {c′})*, which is a superset of *(H_{LM}, R_{LM}, C_{LM})*. Hence, O_{LM} is not a maximal pattern in the column set and can be pruned off. □

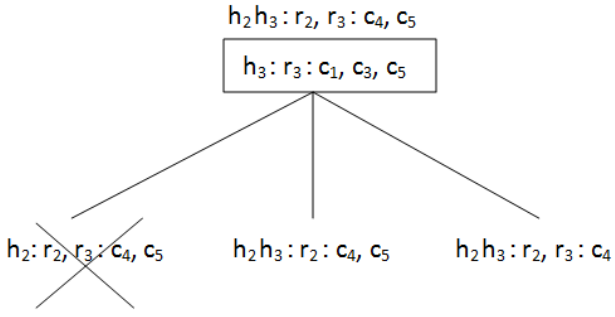

Fig. 2. Left node is pruned due to new pruning technique - closed column set check

For example, if the ternary tree is constructed for the dataset given in Table 1, the node (h_2: r_2, r_3 : c_4, c_5) occur as the left son of a node (h_2 h_3 : r_2, r_3 : c_4, c_5), whose cutter is (h_3 : r_3 : c_1, c_3, c_5). (h_2: r_2, r_3 : c_4, c_5) is not a maximal pattern and pruned by closed column set check w.r.t. c_1, as stated in lemma 2. Fig. 2 clearly depicts the left node (h_2: r_2, r_3 : c_4, c_5) being pruned due to closed column set check w.r.t. c_1, i.e., (h_2: r_2, r_3 : c_4, c_5) is a subset of (h_2: r_2, r_3 : c_1, c_4, c_5). Moreover, (h_2: r_2, r_3 : c_1, c_4, c_5) is a duplicate maximal biclique pattern and has not occurred due to lemma 1. Similarly (h_1, h_2, h_3 : r_3, r_4 : c_5) occur as middle son of the node (h_1, h_2, h_3 : r_3, r_4, r_5: c_5) whose cutter is (h_1 : r_5 : c_1, c_2, c_5) and is pruned w.r.t. c_2, by closed column set check (as depicted in Fig. 3), since as seen in Table 1, h_1 x r_3 x c_2, h_1 x r_4 x c_2, h_2 x r_3 x c_2, h_2 x r_4 x c_2, h_3 x r_3 x c_2, and h_3 x r_4 x c_2, contain '1'.

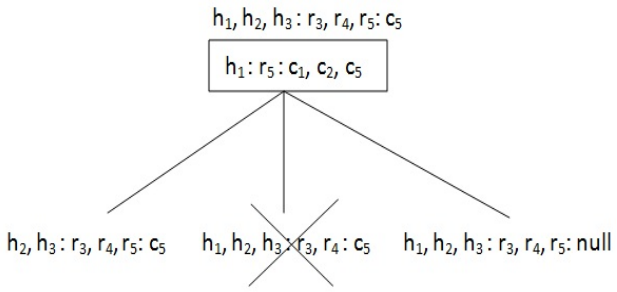

Fig. 3. Middle node is pruned due to new pruning technique - closed column set check

3.3 3D Maximal Biclique Algorithm

Algorithm 1: Cubeminer-MBC*

H- set of heights, R – set of rows, C- set of columns
TL – height elements – meant for left track check
TM – row element – meant for middle track check
h′ – cutter height, r′ – cutter row, C′ – cutter columns
INPUT: 3D Symmetric matrix M with n heights, m rows and m columns
OUTPUT: set of maximal bicliques
TL = TM = null
h′, r′, and C′ are computed from M

```
Cubeminer-MBC*(H, R, C, TL, TM)
1.  if cutter (h′,r′,C′) exists w.r.t. (H, R, C)
2.    if( min(R) = min(C) ) // lemma 1
3.      C = C\min(C)
4.      C′ = C′\min(C)
5.    end
6.    /* left son generation */
7.    if(∄ h′ ∈TL)
8.      if (! closed row set Check)
9.        if (! closed column set Check)  // lemma 2
10.         Cubeminer-MBC*(H\h′, R, C, TL, TM)
11.       endif
12.     endif
13.   endif
14.   /* middle son generation */
15.   if(∄ r′ ∈TM)
16.     if(! closed height set Check)
17.       if (! closed column set Check)//lemma 2
18.         Cubeminer-MBC*(H, R\r′, C, TL ∪ h′, TM)
19.       endif
20.     endif
21.   endif
22.   /* right son generation */
23.   if(! closed height set Check)
24.     if(! closed row set Check)
25.       Cubeminer-MBC*(H, R, C\C′,TL ∪h′, TM ∪ r′)
26.     endif
27.   endif
28. else
29.   Output (H, R, C) as maximal biclique pattern
30. endif
```

3.4 Description

The Cubeminer-MBC* algorithm works in a depth first search manner. It generates a ternary tree, where the left son is generated as *(H\h′, R, C)*, middle son as *(H, R\r′, C)*, and the right son as *(H, R, C\C′)* of a node *(H, R, C)* where *(h′,r′,C′)* is a cutter (line nos.10, 18, & 25). According to lemma 1 (novel enumeration strategy), line nos. 2-5 remove *min(C)* from *C* as well as *C′*, which results in the generation of left son as *(H\h′, R, C\min(C))*, middle son as *(H, R\r′, C\min(C))*, and the right son as *(H, R, C\C′)*, that leads to non-occurrence of duplicate patterns.

Left track check and middle track check are done in line nos. 7 & 15 respectively. Left track check is performed as follows: if the height element *h′* of the cutter does intersect with *TL*, then the left node can be pruned. Middle track check is performed as follows: if the row element *r′* of the cutter does intersect with *TM*, then the middle node can be pruned. *TL* w.r.t. left track check is updated during left and right son generation and *TM* w.r.t. middle track check is updated during right son generation. Closed height set check on middle son and right son, closed row set check on left son and right son are

performed respectively to eliminate biclique patterns which are not maximal. Closed height set check is performed as follows: *Let O_{MR} = (H_{MR}, R_{MR}, C_{MR}) be the middle/right son of any node O = (H, R, C). If* $\exists\ h' \in (\mathrm{H} \backslash H_{MR})$ *(*H *is the full height set), such that* $\forall\ r' \in R_{MR}$, $\forall\ c' \in C_{MR}$, $h' \times r' \times c'$ *is '1', then O_{MR} is not a maximal pattern in the height set and can be pruned off.* Closed row set check is performed as follows: *Let O_{LR} = (H_{LR}, R_{LR}, C_{LR}) be the left/right son of any node O = (H, R, C). If* $\exists\ r' \in (\mathrm{R} \backslash R_{LR})$ *(*R *is the full row set), such that* $\forall\ h' \in H_{LR}$, $\forall\ c' \in C_{LR}$, $h' \times r' \times c'$ *is '1', then O_{LR} is not a maximal pattern in the row set and can be pruned off.* In a similar fashion, as per lemma 2 (new pruning strategy), closed column set check is performed on left son and middle son. Some nodes are not maximal bicliques and still not eliminated by left track check, middle track check, closed height set check, and closed row set check. Hence, the additional closed column set check leads to elimination of non maximal biclique patterns. The cutter generation technique followed is discussed in [5]. Once there are no more cutters, we arrive at leaf node and is a maximal biclique pattern. Hence, due to the novel enumeration strategy, and new pruning techniques maximal biclique patterns occur only once, i.e., zero duplicate patterns are generated.

3.5 User Specified constraints

In order to include user specified constraints, we include the following definition w.r.t. maximal biclique, with an assumption that $p \leq q$ always hold: *A 3D maximal biclique (H, R, C) is said to be (w,p,q)-large if $|H| \geq w$, and $|R| \geq p$ or $|C| \geq p$, and the other is atleast q.* Hence, we enumerate all *(w,p,q)-large* maximal bicliques from 3D symmetric matrix without generating duplicates. In the simplest form *(w,p,q)-large* constraint can be added on all the leaf node.

4 Experimental Results and Analysis

We have implemented and compiled both Cubeminer-MBC* and Cubeminer algorithms using 32-bit Microsoft VC++ compiler, with Windows 7 operating system, in 3 GB RAM and Intel Core i3 processor environment. The datasets used in Table 2 are from [5]. It is very well observed in experiment section of [5] that Cubeminer-MBC generates duplicate patterns for all datasets except the last. It is inappropriate to compare the running time of Cubeminer-MBC* with Cubeminer-MBC, since the later generate duplicate patterns. Moreover, Cubeminer-MBC algorithm works correctly only when the row and column constraints are equal. If they are not equal, it may miss out many patterns. Hence, we compare the running time of Cubeminer-MBC* with Cubeminer.

Table 2
Running time in seconds on various datasets between Cubeminer-MBC* and Cubeminer without minimal size constraints

Dataset	Running time (in seconds)	
	Cubeminer-MBC*	Cubeminer
4-300-300	27.0844	38.0822
3-1000-1000	277.0252	377.7922
9-200-200	84.7764	120.073
5-500-500	694.0542	969.6448

Table 2 compares the running time of two algorithms Cubeminer and Cubeminer-MBC* in seconds without any size constraints, i.e., minimum count of height elements is 1, minimum count of row elements is 1, and minimum count of column elements is 1, and observed that Cubeminer-MBC* outperform the other on all datasets.

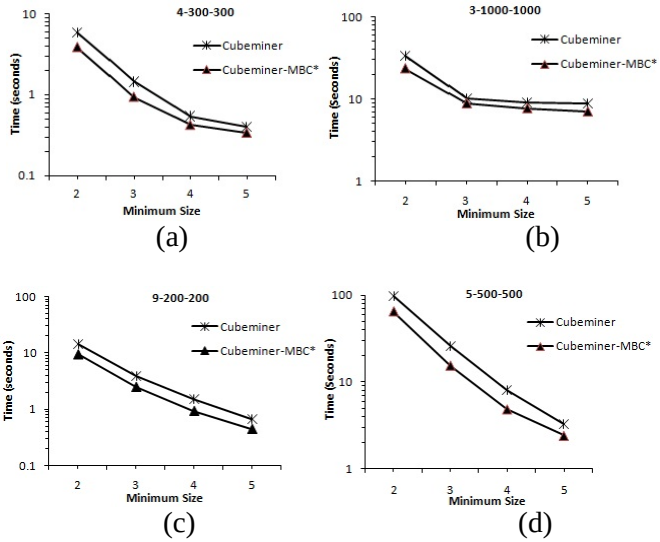

Fig. 4. Running time comparison on synthetic datasets with user specified constraints. a) 4-300-300 b) 3-1000-1000 c) 9-200-200 and d) 5-500-500

Fig. 4 shows the running time of Cubeminer and Cubeminer-MBC* with minimum w = 1, and p and q range from 2 to 6. For all the datasets we compared, Cubeminer-MBC* outperform the other. The comparison was done for the datasets mentioned in Table 2.

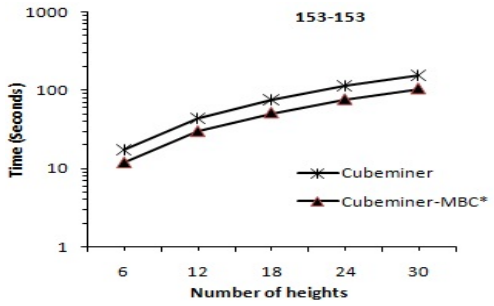

Fig. 5. Running time comparison with number of heights vary from 6 to 30

In Fig. 5, we make the number of heights vary from 6 to 30 and we constrain every maximal 3D pattern to involve atleast 2 heights (w), p = 2 and q = 2 elements. The density of the dataset is 14.9% w.r.t. each height element. The results are represented in Fig. 2, and clearly depict the better performance of Cubeminer-MBC*.

5 Conclusion

We have introduced a novel algorithm Cubeminer-MBC*, inspired by Cubeminer-MBC and Cubeminer, to completely prune the duplicate maximal bicliques in 3D symmetric context. This 100% elimination of duplicate patterns is achieved with the new enumeration technique and corresponding pruning technique proposed in this paper. This pruning has led to fifty percent reduction in search space, and thus performance of Cubeminer-MBC* is better than Cubeminer on symmetric datasets. In this paper, Cubeminer-MBC* is discussed in 3D context, and may also be extended in n-dimensional context.

References:

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

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

[3] L. Ji, K.L. Tan and A.K.H. Tung, "Mining Frequent Closed Cubes in 3D datasets," *Proc. 32nd Intl'. Conference on Very Large Data-bases*, 2006.

[4] R. Peeters, "The maximum edge biclique problem is NP-complete," *Discrete Applied Mathematics*, 131(3), pp. 651-654, 2003.

[5] S. Selvan, R.V. Nataraj, "Efficient Mining of Large Maximal Bicliques from 3D Symmetric Adjacency Matrix," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, iss. 12, pp. 1797-1802, Dec. 2010.

[6] M.D. Savio, A. Sankar, R.V. Nataraj, "A Novel Algorithm to Enumerate Maximal Bicliques from a Symmetric Matrix," *Proc. of the third Int. Conf. on Emerging Applications of Information Technology*, pp. 456- 467, Kolkata, India, Nov. – Dec. 2012.

[7] L. Cerf, J. Besson, C. Robardet, J.F. Boulicaut, "Closed patterns meet n-ary relations," *ACM Transactions on Knowledge Discovery from Data*, 3(1), 2009.

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

[9] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver.2: Efficient mining algorithms for Frequent/closed/maximal itemsets," *In Proc. IEEE ICDM'04 Workshop FIMI'04*, 2004.

[10] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler, "A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data," *Bioinformatics*, 22(9):1122-1129, 2006.

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

[12] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Proc. Int'l Conf. Very Large Data Bases*, pp. 487-499, Sept. 1994.

[13] G. Grahne and J. Zhu, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees," IEEE Trans. Knowledge and Data Eng., vol. 17, no. 10, pp. 1347-1362, Oct. 2005.

[14] J. Wang, J. Han, J. Pei, "CLOSET+: searching for the best strategies for mining frequent closed itemsets," *Proc. of the ninth ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, p. 24-27, Aug. 2003.

[15] J. Besson, C. Robardet, J.F. Boulicaut and S. Rome, "Constraint Based Concept Mining and its Application to Microarray Data Analysis," *Journal of Intelligent Data Analysis*, pp. 59-82, 2005.