Efficient Closure Checking Mechanism for Cube Miner Algorithm

K Shanmuga Priya[1] , R .V.Nataraj [2]
[1]Assistant Professor, Department of Information Technology, PSG College of Technology, Coimbatore, India.
ksp@ity.psgtech.ac.in
[2]Professor, Department of Information Technology, Bannari Amman Institute of Technology, Sathyamangalam, India
rv.nataraj@bitsathy.ac.in

***Abstract*:** This paper, we propose an efficient closure checking mechanism for three dimensional closed pattern mining from 3D datasets. We propose Cube Miner+ which extends the cube miner algorithm and incorporates the proposed closure checking scheme. We also aim at optimizing the existing algorithm by incorporating additional optimization techniques to reduce computation time.

Keywords: *Data mining, closed patterns, mining methods, Algorithms*

INTRODUCTION

Mining frequent patterns or item sets is a fundamental and essential problem in many data mining applications. Frequent item sets play an essential role in many data mining tasks that tries to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more. The task of discovering all frequent item sets is quite challenging. The search space is exponential in the number of items occurring in the database. The support threshold limits the output to a hopefully reasonable subspace. Also, such databases could be massive, containing millions of transactions, making support counting a tough problem. The mining of the complete set of frequent item sets will lead to a number of huge and redundant item sets. Fortunately, this problem can be reduced to the mining of frequent closed item sets, which results in a much smaller number of item sets. A frequent closed item set is either a maximal frequent item set, or a frequent item set whose support is higher than the supports of all its proper supersets.

Frequent pattern mining is a fundamental step to several essential data mining tasks. It is an unsupervised mining technique that identifies all subsets of items or attributes frequently occurring in many database records or transactions. The concept of frequent closed pattern mining is proposed to overcome the problems of redundant patterns and rules and to enumerate feasible patterns from very long frequent patterns. Frequent closed pattern mining is found to be efficient in many application domains like market-basket analysis, www usage mining, gene expression data analysis, etc. In a typical gene expression data analysis, items denote gene expression properties in biological situations. Here, the frequent sets denote the sets of genes that are frequently co-regulated and thus can be suspected to participate to a common function within the cells. Frequent Closed Pattern Mining can reveal patterns about how the expression of one gene may be associated with the expression of a set of genes under a set of environments.

Given such information, we can easily infer that the genes involved participate in some kind of gene networks. Moreover, such association rules can be used to relate the expression of genes to their cellular environments and time periods simultaneously. Such associations can help to detect cancer genes in different cancer developing stages, especially when cancer is caused by a set of genes acting together in-stead of a single gene. It is however possible to use the properties of Galois connection to compute the closed sets on the smaller dimension and derive the closed sets on the other dimension. column set. Several algorithms [2] [5] [6] are proposed to mine frequent closed patterns from 2D space. In 3D frequent closed pattern mining (FCP), patterns are generated from 3D datasets comprising of height set, row set and column set.

Paper Type: Research. Category: Technology
Corresponding Author: K.Shanmuga Priya, Email: *ksp@ity.psgtech.ac.in*

The 2D frequent set mining problem concerns the computation of sets of attributes that are true together in enough transactions, i.e., given a frequency threshold. The patterns are generated from a 2D dataset which comprises of row set and column set. Several algorithms [2] [5] [6] are proposed to mine frequent closed patterns from 2D space coming up with the 3D datasets. The Frequent closed cube (FCC), which generalizes the notion of 2D frequent closed pattern to 3D context with two approaches called *Representative Slice Mining* algorithm *(RSM)* that exploits 2D FCP mining algorithms to mine FCCs and *Cube Miner* that directly mines from 2D Dataset. The process of mining frequent closed pattern consists of three steps:

1 The frequent item sets that have minimum support is selected.
2 Those frequent item sets are used to generate association rules that meet the confidence threshold.
3 Using frequent closed pattern algorithm, the closed pattern are mined from frequent item sets.

Several extensive studies have proposed fast algorithms for mining frequent closed item sets, such as A-close , CLOSET [5], MAFIA (it has an option to generate closed item sets) and CHARM [19]. Various search strategies have been developed, such as depth-first search vs. breadth-first search, vertical formats vs. horizontal formats, tree-structure vs. other data structures, top-down vs. bottom-up traversal, pseudo projection vs. physical projection of conditional database, etc. However, two critical things are missing: (1) there is no systematic study on comparing the strategies and evaluate their pros and cons objectively; and (2) there is no thorough discussion on how to integrate the winning strategies and achieve an even better algorithm. With the research proceeded so far, it is the right time to ask

(1) What are the pros and cons of the strategies?"
(2) What and how can we pick and integrate the best
(3) Strategies to achieve higher performance in general cases

Therefore, we answer the above questions by a systematic study on the search strategies and develop a winning algorithm Cube-Miner+. Cube-Miner+ integrates the advantages of the previously proposed effective strategies as well as some ones newly developed here. A thorough performance study on synthetic and real data sets has shown the advantages of the strategies in terms of runtime and memory usage. The rest of the paper is organized as follows. Section II gives the preliminaries, section III gives an overview of cube miner algorithm, section IV describes our proposed method, Section V gives the implementation and result analysis and section VI concludes the paper.

PRELIMINARIES

In this section we describe about the basic definitions and notations associated with this paper.

Frequent closed item set : An item set Y is a frequent closed item set, if it is frequent and there exists no proper superset Y' ⊃ Y such that sup(Y') = sup(Y). Binary mapping process is done for the given transaction data (Table 1) in order to allow for efficient use of memory. The mapping(Table 2) is done in such a way that, if an item set is present in transaction its support is denoted as '1' and if not, denoted as '0'.

TID	**Items**
T1	i1 i3 i5
T2	i1i2 i4 i5
T3	i1 i2 i3 i4
T4	i1 i2 i5
T5	i1 i2 i3 i4 i5

Table 1 –Transactions Database

Transaction ID	**Items**				
	i1	i2	i3	i4	i5
T1	1	0	1	0	1
T2	1	1	0	1	1
T3	1	1	1	1	0
T4	1	1	0	0	1
T5	1	1	1	1	1

Table 2 – binary mapping for the table 1

A FCC indicates that all its heights, rows, and columns are in "S-contained" relation. Consider the Boolean context given in Table 3. Let *R* denotes set of row vertices, *C* denotes set of column vertices and *H* denotes set of heights vertices.

Table 3 – Sample Boolean Context

*H*1

R/C	c1	c2	c3	c4	c5
r1	0	1	0	1	1
r2	1	1	1	1	1
r3	0	0	1	1	0
r4	1	0	1	0	1
r5	1	1	0	0	1

*H*2

R/C	c1	c2	c3	c4	c5
r1	1	1	1	0	0
r2	1	1	1	1	1
r3	0	0	1	1	0
r4	1	1	1	1	0
r5	1	1	1	0	1

*H*3

R/C	c1	c2	c3	c4	C5
r1	1	1	1	0	1
r2	1	1	1	0	0
r3	1	1	1	1	0
r4	1	1	1	1	1
r5	0	0	1	0	1

Let $R = \{r_1, r_2 \ldots r_n\}$ denote a set of rows, $C = \{c_1, c_2, \ldots, c_m\}$ denote a set of columns, and $H = \{h_1, h_2, \ldots, h_l\}$ denote a set of heights. Then a three-dimension dataset can be represented by a $l \times n \times m$ binary matrix $\mathcal{O} = \mathcal{H} \times \mathcal{R} \times \mathcal{C} = \{o_{k,i,j}\}$ with $k \in [1, l]$, $i \in [1, n]$ and $j \in [1, m]$. Each cell $\mathbf{o_{kij}}$ corresponds to the relationship among height $\boldsymbol{h_k}$, row $\boldsymbol{r_i}$, and column $\mathbf{c_j}$. The value true ("1") denotes the relationship that any two dimensions are "simultaneously contained (S-contained)" in the third one.

The value '0' represents that there exists no-relation and consider as a cutter element for construction of the frequent pattern mining tree. Similarly, all the values of '0' in all transactions of all heights forms a cutter set for the given boolean context. The nodes are generated by excluding the elements of cutter. E.g. if there exists a cutter is (h1 r1 c1), then a left-child node can be generated by excluding the element h1 and the cutter is termed as left-son cutter. Similarly, a right-child node can be generated by excluding the element c1 and the cutter is termed as right-son cutter for this node. The cutter set for the given Boolean context is given in Table 4.

Table 4 – Cutter set (Z) for the Boolean context table

T U V
h1,r1, c1c3
h1,r3,c1c2c5
h1,r4,c2c4
h1,r5,c3c4
h2,r1,c4c5
h2,r3,c1c2c5
h2,r4,c5
h2,r5,c4
h3,r1,c4
h3,r2,c4c5
h3,r3,c5
h3,r5,c1c2c4

Problem Statement

To analyze an efficient methodology and incorporate it into the cube-miner algorithm to improve the performance of frequent closed pattern mining in high- dimensional space.

CUBE MINER ALGORITHM

Cube Miner is a frequent closed pattern mining algorithm that operates directly on 3D dataset to mine FCCs. The construction of FCC tree is done by splitting recursively using cutters. The root node consists of the entire set of height, row and column sets respectively. A cutter (T, U, and V) consists of height, row and column atoms respectively which are generated from the given Boolean context.

The tree construction starts with the whole dataset O (H, R, and C) and then split recursively using the cutters (elements of false values (0)) of Z (Z is a cutter-set obtained from the dataset) until all cutters in Z are used and consequently all cells in each resulting cube have the value '1'. Before generation of child nodes, the cutter checking is done to ensure whether a cutter can be applied to generate nodes. A left son can be generated by including all atoms and removing the left atom of the current cutter from its parent node. A middle son can be generated by including all atoms and removing the middle atom of the current cutter from its parent node. A right son can be generated by including all atoms and removing the right atom of the current cutter from its parent node. Hence a cube miner tree is constructed by splitting each node into three new nodes in the next level if the cutter is applicable.

However, in each level, not all nodes generated are useful for further splitting. Hence, pruning Strategies need to be applied to ensure that we obtain all FCCs and only the FCCs. The strategies are Left Tracking, Middle Tracking and Closeness Checking. The closeness checking is done for Height set and Row set to prune the super-set nodes. The strategies are as follows:

- Left son from a middle or right branch by the cutter whose left atom has cut the node's path before
- Middle son from a right branch by the cutter whose middle atom has cut the node's path before
- Nodes that are unclosed in height set
- Nodes that are unclosed in row set

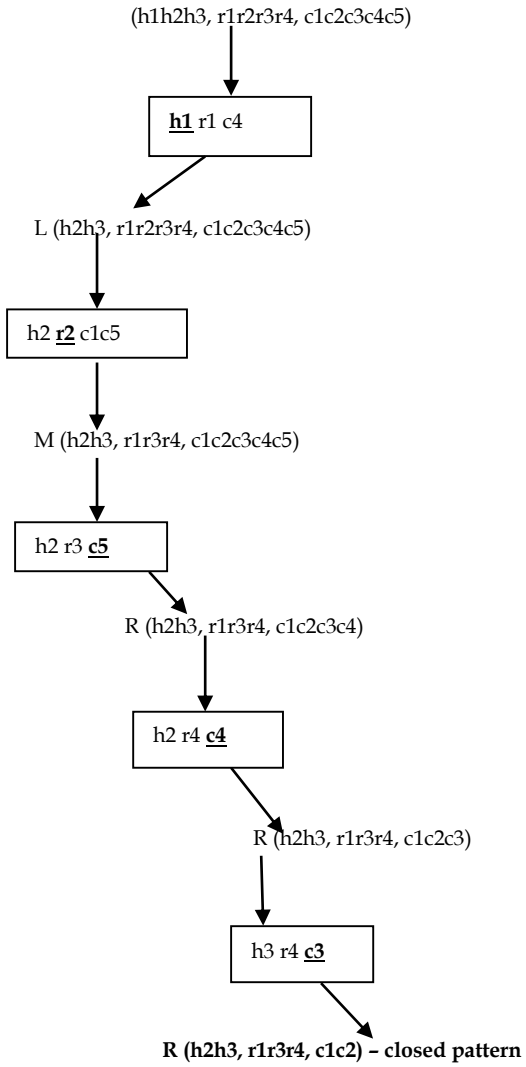
To remove the useless nodes of first two types, left track and middle track checking is done respectively.

CUBE MINER+

Cube-miner+ aims at mining frequent closed patterns from high-dimensional space. The tree construction is done by generating cutters and leaf nodes as described in cube-miner algorithm. The efficiency principle behind the cube-miner+ scheme can be described by comparing the process of closeness checking in cube-miner with that of cube-miner+. The closeness checking for cube miner algorithm is done by comparing the leaf node with all its possible supersets and then verifying it with its corresponding value in the dataset. E.g., consider the leaf node pattern as (h3, r1r4, c1c2). For height-set checking, this pattern is compared with remaining nodes in root i.e. here h1 h2 is left out in pattern. The possible combinations for this pattern are (h1h3, r1r4, c1c2), (h2h3, r1r4, c1c2), (h1h2h3, r1r4, c1c2). Now these are verified with the given Boolean context or the dataset for the value '1' in it. If the superset pattern (h1h2h3, r1r4, c1c2) found to be closed, then the obtained leaf (h3, r1r4, c1c2) can be pruned off. Similar procedure is followed for row-set checking with all its possible combinations of supersets with dataset. If the Boolean context comprises of many rows and columns, then computing the possible combinations for the obtained pattern is a tedious process and also is time-consuming. Hence, we can incorporate an efficient methodology which would follow a novel procedure to reduce computation time without computing the possible combinations for the pattern.

The proposed "Cube-Miner+" is an efficient closure checking mechanism to mine frequent closed patterns. The closeness checking for Cube-Miner+ is performed as follows:

Figure-1 Cube-Miner+ Methodology



The closeness checking for row and height sets can be performed on the leaf node by tracking only the elements of set of cutters of the leaf node's path from the root node. This eliminates the necessity of checking the leaf node with all possible supersets and verifying with its value in the dataset. Hence it avoids the unnecessary computations involved in performing closure checking. The proposed technique is depicted in the figure-1. In the figure, L, M and denotes left child node, middle child node and right child nodes respectively.

The above tree is a part of the whole construction of the frequent pattern mining tree constructed using cube-miner algorithm. This tree traces out a path from the root node to the leaf node for the obtained closed pattern. The rectangular box denotes the cutter set (height, row, and column). The cutters are named as per their child nodes e.g. R – denotes the right child node which is derived from its preceding cutter and it is called as right-son cutter and the same cutter may act as a middle-son cutter for a middle-son child node. The closeness checking is done only for height sets and row sets. From this tree path, we can see that there doesn't exist a superset for the obtained closed pattern. Hence the checking can be done easily in this path from root node to leaf node alone which reduces time in computing combinations for comparison of supersets and thus reduces the memory space for storing and comparing with the dataset values. Similarly, all the parts of the tree can be evaluated and the obtained closed patterns are verified by checking the path from the root node to leaf node.

IMPLEMENTATION AND RESULT ANALYSIS

The Cube-Miner+ technique is incorporated into cube-miner algorithm and implemented using C Language. The code was compiled using 32-bit Microsoft Visual C++ compiler. Boolean data type is used to store the datasets. The experiment is done for market-basket data analysis. The dataset consisting of even 5000 items is included and implemented for generation of frequent closed patterns.

The proposed technique is experimented using synthetic datasets, both generated manually and using IBM Data generator. For our experiment, we have used different types of datasets and their characteristics are given in the table 5. All the experiments are run on Pentium 4 PC with 1GB RAM. While conducting this experiment, we have made sure that no other programs were run in the background. Table 6 shows the comparison between Cube-Miner and Cube-Miner+ in terms of running time (seconds).

Table 5 – Datasets used

Dataset ID	**Dataset name**	**No.of Vertices**
01	Manual1	10
02	IBMGenerated1	100
03	IBMGenerated2	200
04	IBMGenerated3	500

Table 6 – Total running time in seconds for cube-miner and cube-miner+ for the datasets mentioned above

Dataset ID	Minimum Support	Execution Time(seconds)	
		Cube-Miner	Cube-Miner+
01	6	0.000	0.000
02	6	0.921	0.757
03	6	1.456	1.200
04	6	4.145	3.054

As shown in the table 6, Cube-Miner+ takes lesser execution time than Cube-Miner. When the number of vertices is increased, computation time is increased in cube-miner implementation. Whereas, when the number of vertices is increased, computation time taken by cube-miner+ is less, when compared to cube-miner. This table clearly depicts the efficiency of the cube-miner+ technique.

CONCLUSION

In this paper, we have described an efficient closeness checking mechanism for frequent closed pattern mining from a 3D dataset. We conducted an extensive performance study on both real and synthetic datasets. Our results showed that the scheme can mine the frequent closed patterns efficiently. An extensive study on performance of the proposed technique is to be continued for further optimization. The notion of 3D frequent closed pattern mining can be generalized to high-dimensional space as the future work.

ACKNOWLEDGEMENTS

We wish to thank the reviewers for their valuable comments which helped us to enhance our paper. We also wish to thank the authors of Cube-Miner algorithm for responding to our numerous queries.

REFFERENCES

[1] Liping, Ji, Tan, K.L. and Tung, A.K.H. ”Mining Frequent Closed Cubes in 3D datasets,” Proc. 32nd int. conference on very large databases, 2006

[2] Besson J, Robardet C and Boulicaut J.F, "Constraint based mining of formal concepts in transactional data, PAKDD’04 pp. 615-624, 2004.

[3] Liping, Ji “Mining Localized Co-expressed Gene Patterns from Microarray Data,” PhD dissertation, School of Computing., Na-tional University of Singapore, June 2006.

[4] Mohammed J. Zaki and Ching-Jui Hsiao, “Efficient Algorithms for Mining Closed Items sets and Their Lattice Structure”,IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 4, april 2005

[5] Jian Pei, Jiawei Han and Runying Mao, “CLOSET : An Efficient Algorithm for Mining Frequent Closed Item sets”, Intelligent Database Systems research Lab, School of Computing Science, Simon Fraser University, Canada.

[6] Jianyong Wang, Jiawei Han, Jian Pei, “CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Item sets”, Department of Computer Science, New York

[7] Liping Ji, Kian-Lee Tan and Anthony K.H. Tung, “Compressed Hierarchical Mining of Frequent Closed Patterns from Dense Data Sets”, IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 9, sep 2007

[8] Lucchese C, Orlando S and Perego R, "Fast and Memory Efficient Mining of Frequent Closed Item sets", IEEE Transactions on Knowledge and Data Engineering, VOL 18, No 1, pages 21-36, January 2006

[9] Grahne G., Zhu J. “Fast Algorithms for Frequent Itemset Mining Using FP-Trees”, IEEE Transactions on Knowledge and Data Engineering, Vol 17, No 10, pages 1347- 1362, October 2005.

[10] Gao Cong, Kian-Lee Tan, Tung, A. K. H. and Feng Pan, Mining Frequent Closed Patterns in Microarray Data”, ICDM’04, Vol 1, Issue 4, pp: 363-366, Nov 2004

[11] Ben Yahia, S, Hamrouni,T and Mephu Nguifo E, "Frequent Closed item set based Algorithms: A through structural and analytical survey," SIGKDD Explorations, Vol. 8, Issue 1, pages 93-104 2006

[12] Jiawei Han, Jian Pei, Yiwen Yin, Runying Mao, “Mining Frequent Pattern without candidate Generation : A Frequent Pattern Approach” Journal of Data Mining and Knowledge Discovery , Springer, pages 53-87, 2004.

[13] Mingjun Song, Sanguthevar Rajasekaran, "A Transaction Mapping Algorithm for Frequent Itemsets Mining", IEEE Transactions on Knowledge and Data Engineering,, VOL 18, No 4, pages 472-481, April 2006.

[14] D.Burdick,M.Calimlim,J.Flannick,J.Gehrke,Y.Yiu, "MAFIA: A Maximal Frequent Itemset Algorithm", IEEE Transactions on Knowledge and Data Engineering, VOL 17, No 11,Pages 1490 - 1504, November 2005.

[15] R. Agrawal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large databases", In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington, DC, May 1993.

[16] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", In Proceeding of Int. Conf. Very Large Data Bases, pages 487-499, Santiago, Chile, September. 1994.

[17] A. Savasere, E. Omiecinski, and S. Navath, "An efficient algorithm for mining association rules in large databases", In Proc. of Intl. Conf. on Very Large Databases (VLDB), 1995.

[18] N. Pasquier, Y. Bastide, R. Taouil, and L.Lakhal, "Discovering Frequent Closed Itemsets for Association Rules", Proc. 7th Int. Conf. Database Theory (ICDT'99), pages 398-416, January 1999.

[19] M.J. Zaki,C.J.Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining",Proc SIAM Int. Conf. Data Mining, pages 457-473, April 2002.

[20] Dao-l Lin and Zvi M. Kedem, "PINCER-SEARCH: An Efficient Algorithm for Discovering the Maximum Frequent Set", IEEE Trans. on Knowledge and data Engineering, VOL 14, No. 3, pages 553-566, June 2002.

[21] Ahmed Shakil, Frans Coenen, Paul Leng, "Tree based partitioning of data for association Rule Mining", Knowledge and Information Systems, Springer, Vol 10, N0 3, pages 315-331, October 2006.

[22] A. Veloso, M. Otey, S. Parthasarathy, and W. Meira, "Parallel and distributed frequent itemset mining on dynamic datasets", In Proc. of the High Performance Computing Conference, HiPC, Hyderabad, India, December 2003.

[23] Yew-kwong Woon, Wee-keong Ng, Ee-Peng Lim, "A Support-ordered Trie for Fast Frequent Itemset Discovery", IEEE Transactions on Knowledge and Data Engineering, Vol 16, No 7,pages 875-879, July 2004.

[24] Guimei Liu, "Supporting Efficient and Scalable Frequent Pattern Mining", PhD Thesis, Hong Kong University, May 2005