Frequent Patterns Mining from Data Cube Using Aggregation and Directed Graph

Kuldeep Singh, Harish Kumar Shakya and Bhaskar Biswas

Abstract An algorithm has been proposed for mining frequent maximal itemsets from data cube. Discovering frequent itemsets has been a key process in association rule mining. One of the major drawbacks of traditional algorithms is that lot of time is taken to find candidate itemsets. Proposed algorithm discovers frequent itemsets using aggregation function and directed graph. It uses directed graph for candidate itemsets generation and aggregation for dimension reduction. Experimental results show that the proposed algorithm can quickly discover maximal frequent itemsets and effectively mine potential association rules.

Keywords Association rule · Data cube · Frequent itemsets · Support count · Directed graph · Maximal itemsets

1 Introduction

Frequent pattern mining has been a major task in data mining. Frequent pattern mining finds interesting association or correlation among a large number of itemsets. Finding frequent patterns play an important role in association rule mining, correlation. Frequent pattern are the patterns that appear repeatedly in the dataset.

K. Singh (✉) · H.K. Shakya · B. Biswas
Department of Computer Science and Engineering, IIT-BHU, Varanasi, Uttar Pradesh, India
e-mail: kuldeep.rs.cse13@iitbhu.ac.in

H.K. Shakya
e-mail: hkshakya.rs.cse@iitbhu.ac.in

B. Biswas
e-mail: bhaskar.cse@iitbhu.ac.in

S. Das et al. (eds.), *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015*, Advances in Intelligent Systems and Computing 404, DOI 10.1007/978-81-322-2695-6_15

It was first proposed by Agrawal [1] in 1994. In this paper, a frequent pattern mining technique is proposed that use directed graph together with aggregation to discover knowledge from data cube. Aggregation transforms 3D model views to 2D (tabular) view of dataset using sum-based measure. Directed graph provides fast and easy construction of candidate generation. Candidate generation using directed graph saves time and memory consumption because, most candidate itemsets are frequent. Frequent 2-itemsets are required for the directed graph generation. First item of a frequent 2-itemset works as origin and second item is like destination for an edge in the directed graph.

Many algorithms have been developed for searching association rules. The main challenge in mining association rules is developing fast and efficient algorithms that can handle large volume of data, minimum time scans database and find associated rule quickly. Most of the proposed apriori-like algorithms for mining association rules are wasting lots of time to generate candidate itemsets. FP-Growth algorithm is also very useful for finding association rule. The FP-Growth algorithm does not generate candidate itemsets and so takes less time to find frequent itemsets [2]. But it also has limitations with respect to space and time. The proposed algorithms overcome these limitations. Frequent itemsets can be of two types, closed and maximal. A frequent itemset is called maximal if it has no superset that is frequent [3]. An itemset is closed if it is frequent but none of its superset has the same support count [4].

The proposed algorithm uses directed graph for candidate itemsets generation. It saves time and memory consumption because it generates minimum number of candidate itemsets, those are likely to be frequent. Frequent 2-itemsets are required for the directed graph generation. First item of a frequent 2-itemset works as origin and second item like destination for an edge in the directed graph [5]. Directed graph gives us minimum candidate itemsets that are likely to be frequent itemsets.

Proposed algorithms generate frequent patterns using data cube reduction. Data reduction technique used to reduce the dataset that is much smaller in volume. Reduced dataset should maintain the integrity of original dataset. Mining on the reduced dataset produce exactly the same or almost the same results. If reconstruction of original dataset from reduced dataset is without any loss of information, then the reduced dataset is called lossless. Otherwise, we loss the information and produce approximation of original dataset, then the reduced dataset is called loosy. There are many methods of data reduction. Dimensionality reduction and numerosity reduction techniques are the main forms of data reduction. Dimensionality reduction is the process of reducing the number of attributes or variables. There are many dimensionality reduction methods which transform the original dataset into a smaller spaced dataset, e.g., wavelet transformation, principal component analysis, and attribute subset selection [6]. Numerosity reduction techniques replace the original dataset by smaller form of data representation. Numerosity reduction may be parametric of nonparametric. Parametric methods are used to

estimate the data, only the data parameters needs to be stored, instead of the original dataset, e.g., regression and log-linear model. Nonparametric methods are used to reduce representations of the dataset, e.g., histograms, clustering, sampling, and data cube aggregation. A data cube allows data to be viewed in multiple dimensions. The data cube contains three dimensional item, time, and location. It can return the total sales for any combination of the three dimensions. Apex cuboid refers to the case where the group-by is empty. It contains the total sum of all sales. The base cuboid is the least generalized of the cuboids. The apex cuboid is the most generalized of the cuboid. If we start at the base cuboid and explore upward, this is similar to roll-up operation.

2 Frequent Patterns Discovery Using Directed Graph

In this section, we present the proposed algorithm. The proposed algorithm is shown in Fig. 1.

Input: Data-Cube, min_sup;
Output: Set of Maximal frequent itemset.
1. Condense one predicate of Data-Cube using aggregation
2. Find total_item_count of all the items
3. If total_item_count < min_sup than
4. Remove those items
5. Otherwise add 1-itemsets into L_1
6. Sort the items in ascending order according to their total_item_count
7. Generate candidate 2-itemsets using lexicographical order with L_1
8. Calculate total_item_count of candidate 2-itemsets
9. If total_item_count < min_sup than
10. Remove those itemsets
11. Otherwise add 2-itemsets into L_2
12. Generate directed graph using L_2
13. If (Ii,Ij) in L_2 than
14. Draw a directed edge from I_i to I_j in directed graph
15. Traverse the directed graph and generate candidate k-itemsets
16. Calculate total_item_count of k-itemsets
17. If total_item_count < min_sup than
18. Remove those itemsets
19. Otherwise add k-itemsets into L_k
20. Discover Maximal Frequent itemset from L_k
21. Return Maximal Frequent itemset

Fig. 1 The proposed algorithm

2.1 Algorithm Details

The proposed algorithm can be divide into fives main steps on the basis of broad tasks. These main steps are explained below.

Perform Aggregation on Data Cube.
The data cube condenses using aggregation. We simply computed by aggregation the counts from cells contained in the one predicate. The resulting dataset is smaller in volume without loss of information necessary for frequent pattern mining [7–9]. We did this aggregation because memory consumption is reduced. Aggregate function and the group-by operator produce 1D aggregate or more dimensional aggregates. Aggregation functions return a single value.

Generate the set of frequent 1-itemsets L_1.
Calculate total_item_count of items by counting item_freq of items. If total_item_count of I_j item is smaller than min_sup, itemset I_j in not a frequent 1-itemset so removed. Otherwise, itemset I_j is frequent and is added to the set of frequent 1-itemset L_1. Sorting helps us to generate minimum candidate itemsets as in [10]. Sorting has been done of all frequent 1-itemsets by total_item_count in ascending order, these are likely to be frequent.

Generate the set of frequent 2-itemsets.
In this step, we generate candidate 2-itemsets in lexicographical order using L_1. Count the total_item_count of the candidate 2-itemset $\{I_i, I_j\}$. If the total_item_count of candidate 2-itemset $\{I_i, I_j\}$ is greater than min_sup, then itemset $\{I_i, I_j\}$ frequent 2-itemset and added into L_2, otherwise removed.

Construction of Directed Graph using frequent 2-itemsets.
We construct the directed graph by using frequent 2-itemsets L_2. If itemset $\{I_i, I_j\}$ frequent 2-itemset, then draw a directed edge from I_i to I_j in directed graph [11, 12]. First item of 2-itemset is the origin and second item is the destination. We draw all directed edge by using itemset of L_2. After drowning all the edges, we present directed graph.

Generate all the Candidate k-itemsets using directed graph.
This proposed algorithm traverses once to the directed graph and generate candidate itemsets by using the directed neighbor nodes of itemset. Start from I_i and traverse all the possible reachable nodes. After completed traversing it leads to formation of simple paths; these simple paths are our candidate itemsets. Nodes of possible simple paths are the elements of these itemsets.

Find Frequent itemsets.
In the final step, we collect only frequent itemsets. For finding frequent k-itemsets, count the total_item_count of the candidate k-itemset $\{I_i, I_j, \ldots I_k\}$. If the total_item_count of candidate k-itemset $\{I_i, I_j, \ldots I_k\}$ is greater than min_sup, then $\{I_i, I_j, \ldots I_k\}$ frequent k-itemset and added into L_k. Otherwise, remove infrequent

itemsets. Finally, we got the final set of frequent k-itemsets L_k. Now, we verify whether frequent k-itemsets L_k are maximal or not. The itemsets that do not fulfill the property of maximal itemsets are removed; otherwise added into final result set.

2.2 Example

We present a simple example to help illustrate working of the proposed algorithm. The data cube showed in Fig. 2. The user has given minimum support threshold as 5. We can find out the new minimum support threshold by multiplying the number of aggregated cell with old minimum support threshold. So the new minimum support threshold is (min_sup) is (20 = 5*4) 20.

Perform Aggregation on Data Cube.
The data cube is aggregated into a tabular dataset as shown in Fig. 2b. The data cube can be aggregated so that the resulting data summarize the total sales in all locations instead of sale per city, as shown in Fig. 2. The attributes of dimension location are grouped. The data cube consist the sales per year, for the year 2011–2014. We are interested in the total sale per year, rather than the total obtained per city. Aggregation returns the total sales for combination of the dimension location. The dimension attributes are grouped, where locations are grouped. Therefore, we perform aggregation summarization of location predicates. Figure 2a shows the condensed dataset.

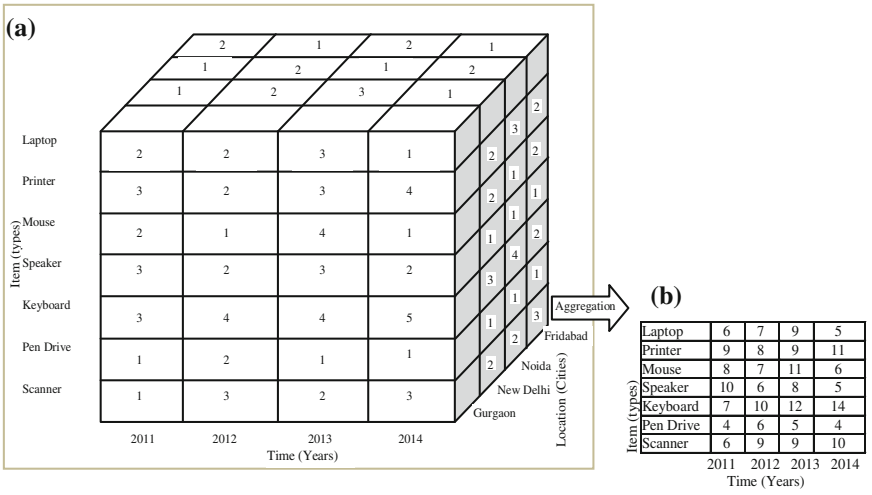

Fig. 2 Sales data cube of computer devices for the year 2011–2014. On the *right*, data are aggregated to provide the total sale of all locations

Generate the set of Frequent 1-itemsets L_1.
Scan the condensed dataset and calculate total_item_count of items by counting values as shown in Fig. 3a. Those items that have total_item_count less than min_sup are removed. Item pen drive is removed because it has less total total_item_count to the min_sup. After removal of item pen drive, the resulted set is shown as Fig. 3b. Then, sort all frequent 1-itemsets by total_item_count in ascending order. Now, the set of sorted itemsets are L_1 = {Laptop, Speaker, Mouse, Scanner, Printer, Keyboard}.

Generate Candidate 2-itemsets:-.
Generate candidate 2-itemsets using sorted 1-itemsets L_1. Count the total_item_count of the candidate 2-itemset $\{I_i, I_j\}$ as shown in Fig. 4a. If the total_item_count of candidate 2-itemset is $\{I_i, I_j\}$ greater than or equal to min_sup, then $\{I_i, I_j\}$ frequent 2-itemset and added into L_2. Otherwise, remove 2-itemset $\{I_i, I_j\}$. Itemset {Laptop, Speaker} and {Speaker, Scanner} are infrequent because total_item_count of these itemsets less than min_sup. Now, the frequent 2-itemsets are shown in Fig. 4b.

Construction of Directed Graph using Frequent 2-itemsets.
Construct the directed graph using L_2. If $\{I_i, I_j\}$ in L_2, then draw a directed edge from I_i to I_j in directed graph. A graph consists of two things node (V) and direct edged {Mouse, Scanner, Keyboard} or $\{L_1\}$ and E = {{Laptop, Mouse}, {Laptop, Scanner}, {Laptop, Printer}, {Laptop, Keyboard}, {Speaker, Mouse}, {Speaker, Printer}, {Speaker, Keyboard}, {Mouse, Scanner}, {Mouse, Printer}, {Mouse, Keyboard}, {Scanner, Printer}, {Scanner, Keyboard}, {Printer, Keyboard}}. For itemset {Laptop, Mouse}, Laptop is the origin point and Mouse is the destination point of the edge. After construction of all the edges, the graph looks like, as shown in Fig. 5.

(a)

Item_type	2011	2012	2013	2014	total_item_count	
Laptop	6	7	9	5	27	
Printer	9	8	9	11	37	
Mouse	8	7	11	6	32	
Speaker	10	6	8	5	29	
Keyboard	7	10	12	14	43	
Pen Drive	4	6	5	4	19	Dropped item
Scanner	6	9	9	10	34	

(b)

Item_type	2011	2012	2013	2014	total_item_count
Laptop	6	7	9	5	27
Printer	9	8	9	11	37
Mouse	8	7	11	6	32
Speaker	10	6	8	5	29
Keyboard	7	10	12	14	43
Scanner	6	9	9	10	34

Fig. 3 The generation of frequent 1-itemsets. **a** Candidate 1-itemsets. **b** Frequent 1-itemsets

(a)

Candidate 2-itemsets	2011	2012	2013	2014	total_item_count
{Laptop, Speaker}	6	6	6	5	23
{Laptop, Mouse}	6	7	9	5	27
{Laptop, Scanner}	6	7	9	5	27
{Laptop, Printer}	6	7	9	5	27
{Laptop, Keyboard}	6	7	9	5	27
{Speaker, Mouse}	8	6	6	5	25
{Speaker, Scanner}	6	6	6	5	23
{Speaker, Printer}	9	6	6	5	26
{Speaker, Keyboard}	7	6	6	5	24
{Mouse, Scanner}	6	7	9	6	28
{Mouse, Printer}	8	7	9	6	30
{Mouse, Keyboard}	7	7	11	6	31
{Scanner, Printer}	6	8	9	10	33
{Scanner, Keyboard}	6	9	9	10	34
{Printer, Keyboard}	7	8	9	11	35

Dropped itemsets

(b)

Frequent 2-itemsets	2011	2012	2013	2014
{Laptop, Mouse}	6	7	9	5
{Laptop, Scanner}	6	7	9	5
{Laptop, Printer}	6	7	9	5
{Laptop, Keyboard}	6	7	9	5
{Speaker, Mouse}	8	6	6	5
{Speaker, Printer}	9	6	6	5
{Speaker, Keyboard}	7	6	6	5
{Mouse, Scanner}	6	7	9	6
{Mouse, Printer}	8	7	9	6
{Mouse, Keyboard}	7	7	11	6
{Scanner, Printer}	6	8	9	10
{Scanner, Keyboard}	6	9	9	10
{Printer, Keyboard}	7	8	9	11

Fig. 4 The generation of frequent 2-itemsets. **a** Candidate 2-itemsets with total_item_count. **b** The frequent 2-itemsets

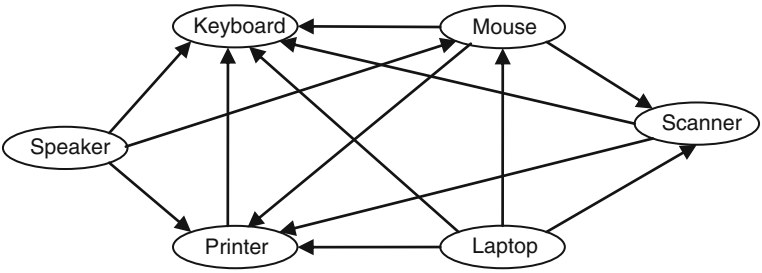

Fig. 5 The directed graph

Generate candidate k-itemsets using directed graph.
Generate candidate k-itemsets by traversal of directed graph. Start from I_i and traverse all the reachable nodes. An I_j is reachable from Ii if there is a simple path from I_i to I_j. After completion of traversing, we found that there are many simple paths; these simple paths are our candidate itemsets. The candidate k-itemsets generated are shown in Table 1. Figure 6 shows the process of candidate itemset generation through traversing the directed graph. Node Keyboard is not having any candidate itemset because it did not have any outgoing edge or node Keyboard; not a source node for any other node.

Table 1 The candidate k-itemsets with source node

Source node	k-candidate itemsets
Laptop	{Laptop, Mouse, Scanner}, {Laptop, Mouse, Scanner, Printer}, {Laptop, Mouse, Scanner, Printer, Keyboard}, {Laptop, Mouse, Scanner, Keyboard}, {Laptop, Mouse, Printer}, {Laptop, Mouse, Printer, Keyboard}, {Laptop, Mouse, Keyboard}, {Laptop, Scanner, Printer}, {Laptop, Scanner, Printer, Keyboard}, {Laptop, Scanner, Printer}, {Laptop, Printer, Keyboard}
Speaker	{Speaker, Mouse, Scanner}, {Speaker, Mouse, Scanner, Printer}, {Speaker, Mouse, Scanner, Printer, Keyboard}, {Speaker, Mouse, Scanner, Keyboard}, {Speaker, Mouse, Printer}, {Speaker, Mouse, Printer, Keyboard}, {Speaker, Mouse, Keyboard}, {Speaker, Printer, Keyboard}
Mouse	{Mouse, Scanner, Printer}, {Mouse, Scanner, Printer, Keyboard}, {Mouse, Scanner, Keyboard}, {Mouse, Printer, Keyboard}
Scanner	{Scanner, Printer, Keyboard}
Printer	{}

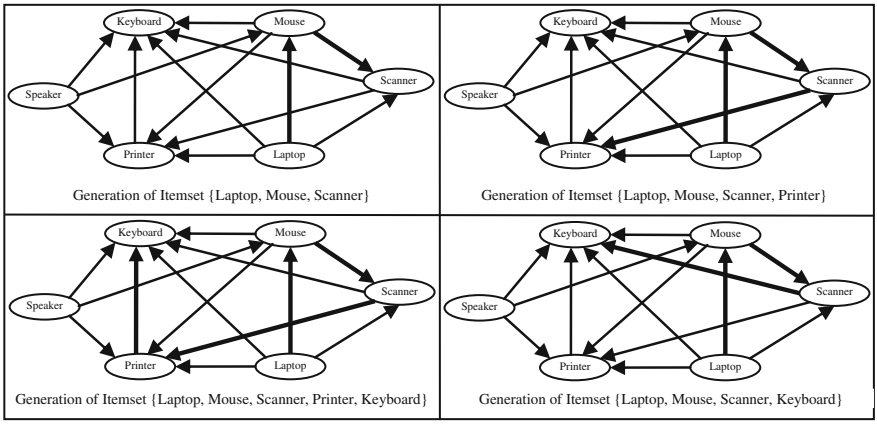

Fig. 6 The generation of candidate k-itemsets through traversing the directed graph

Find frequent itemsets.

At last, we check whether each candidate k-itemsets is frequent or infrequent itemsets. Count the total_item_count of the candidate k-itemset $\{I_i, I_j, \ldots I_k\}$. The total_item_count of itemsets {Speaker, Mouse, Scanner}, {Speaker, Mouse, Scanner, Printer}, and {Speaker, Mouse, Scanner, Printer, Keyboard} and {Speaker, Mouse, Scanner, Keyboard} are less than Sup_count, so remove these itemsets. All other itemsets are having greater total_item_count than min_sup, so these are added into frequent k-itemset L_k.

In the end, we find maximal frequent itemsets from L_k. All frequent k-itemsets are the subsets of the maximal frequent itemset {Laptop, Mouse, Scanner, Printer, Keyboard} and {Speaker, Mouse, Printer, Keyboard}. In maximal itemset we did not include frequent subsets. So, remove the subsets of the maximal itemsets.

3 Experiment

To find experimental results, we have used mushroom and accidents dataset obtained from UCI [13] and synthetic dataset which are created. The algorithms were implemented in Java and tested on a windows platform. Mushroom dataset have total number of instances: 8124, number of attributes: 23, number of items: 119. Dimensions come from the former five attribute of the mushroom dataset. We analyses that as support count increases, the execution time goes down as shown in Fig. 7.

Another dataset used for experiment result is accidents [14]. Accidents dataset have total number of Instances: 3196, number of attributes: 37, number of items: 75. Dimensions come from the former five attributes of the accidents dataset. The result shows that lesser the minimum support threshold takes more time to execute as shown in Fig. 8.

The proposed algorithm takes lesser time when minimum support count threshold is higher. The frequent pattern generated by Apriori and proposed algorithm do not have much difference. Candidate Generation: To better understand the execution time behavior of the proposed algorithm, we explicitly evaluated the number of candidate itemsets generated by the proposed algorithm. The interesting observation here is that the number of candidate itemsets counted by proposed algorithm is less than other traditional algorithms. Mostly, the candidates generated by proposed algorithm are frequent so we can say that candidate itemsets generated by proposed algorithm are likely to be frequent.

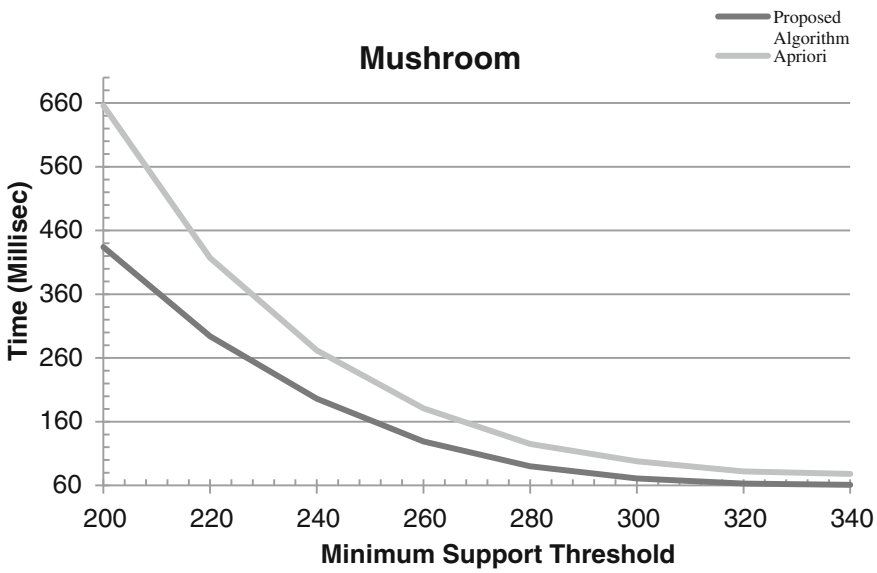

Fig. 7 Execution time with different minimum support thresholds for mushroom dataset

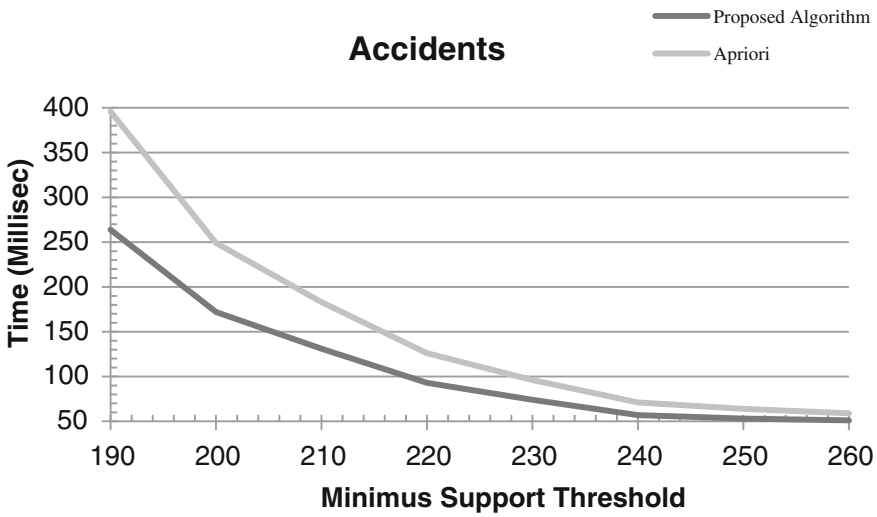

Fig. 8 Execution time with different minimum support thresholds for accidents dataset

Memory consumption: We also monitored the memory consumption overhead of the various algorithms. We found that the memory occupancy of the proposed algorithm is comparable with other algorithms. The consumption was related less in condensed data cube. Condensed Data cube requires less memory because aggregate function is applied on the data cube.

Discovered Patterns: The frequent patterns generated by proposed algorithm and Apriori do not have much difference. The sample result for dataset mushroom and accidents are shown in Fig. 7 and Fig. 8, respectively.

Finally, turning to proposed algorithm, we find that it consistently provides the best performance across all the datasets.

4 Conclusion

The proposed algorithms generate candidate itemset quickly by graph traversing without using join and prune. Proposed algorithm generated minimum candidate itemsets and these candidate itemsets are most likely to be frequent. The proposed algorithm use directed graph for generating candidate itemsets. The algorithm shows that mostly candidate k-itemsets are frequent because sorting of the frequent 2-itemsets. So, the proposed algorithm spends less time to find candidate and maximal frequent itemsets. Directed graph generates candidate itemsets by graph traversing instead of join and prune steps. The proposed algorithm scans the database only once.

5 Future Work

The above described work can be enhanced for directed graph generation. This directed graph can be generated by using 1-frequent itemset that can enhance the performance of the proposed algorithm.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Databases, pp. 487–499. Santiago, Chile (1994)
2. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the ACM-SIGMOD Conference Management of Data, pp. 1–12 (2000)
3. Amir, A., Anumann, y., Feldman, R., Fresko, M.: Maximal association rules: a tool for mining associations in text. J. Intell. Inf. Syst. **25**(3), 333–345 (2005)
4. Zaki, M.K.: Closed itemset mining and non-redundant association rule mining. In: Encyclopedia of Database Systems, Springer (2009). doi: 10.1007/978-0-387-39940-9_66
5. Rosen, K.H.: Discrete Mathematics and Its Application, 7th edn. McGraw-Hill Publication, Columbus (2012)
6. Han, J., Kamber, M.: Data Mining Concepts and Techniques, 3rd edn. Morgan Kaufmann Publishers, San Francisco (2012)
7. Harsola, S.K., Deshpande, P.M., Haritsa, J.R.: IceCube: efficient targeted mining in data cubes. In: ICDM, pp. 894–899 (2012)
8. Gray, J., Chaudhuri, S., Bosworth, A., Layman, D., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Min. Knowl. Discov. Arch. **1**(1), 29–53 (1997)
9. Messaoud, R.B., Rabaseda, S.L.: Enhanced mining of association rules from data cubes. In: Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, pp. 11–18 (2006)
10. Liu, H., Wang, B.: An association rule mining algorithm based on a Boolean matrix. Data Sci. J. **6**, s559–s565 (2007)
11. Liu, N., Ma, L.: Discovering frequent itemsets an improved algorithm of directed graph and array. In: 4th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 1017–1020 (2013)
12. Xu, W.X., Wang, R.J.: A fast algorithm of mining multidimensional association rules frequently. In: International Conference on Machine Learning and Cybernetics, pp. 1199–1203. China (2006)
13. Schlimmer, J.: https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data
14. Shapiro, A.: https://archive.ics.uci.edu/ml/datasets/Accident (KRKPA7)