National Conference on Emerging Trends in Computer, Electrical & Electronics (ETCEE-2015)
International Journal of Advance Engineering and Research Development (IJAERD)
e-ISSN: 2348 - 4470 , print-ISSN:2348-6406,Impact Factor:3.134

An Effective Hash-Based Algorithm for Mining Association Rules

Shraddha Savaliya[1], Dhaval Nimavat[2]

[1]Post Graduate Student, [2]Assistant Professor
Department of Computer Engineering, Atmiya Institute of Technology and Science, Rajkot, Gujarat, India
[1]shraddhasavaliya91@gmail.com
[2]dhavalnimavat@icloud.com

***Abstract*—Now a day the database is becoming large. The Apriori algorithm is used to find frequent itemsets but in the apriori algorithm there is requirement to scan the database many times this is significant when we work with the small database but we concern with large database so we will use hash based technique which reduce the size of the candidate generation. Here we have used H-bit array hashing algorithm and N-hash algorithm to create new hashing algorithm which is having advantages of both the algorithm. The generated algorithm does not use the lengthy and complex process of mapping items into the bucket by hashing algorithm. The proposed algorithm is efficient and requires less space for hash table and maps items into hash table without collision.**

***Keywords*— Apriori,**
DHP: Direct Hashing and Pruning
PHP: Perfect Hashing and Pruning
PHS: Perfect hashing and database Pruning
MPIP: Multi-Phase Indexing and Pruning
HBMFI-LP: Hash Based Maximal Frequent Itemsets-Linear Probing
HBFI-QP: Hash Based Frequent Itemsets-Quadratic Probing
H-BAH: H-Bit Array Hashing

I. INTRODUCTION

Apriori algorithm is based on the fact that the algorithm uses prior knowledge of frequent item set properties. Apriori employs an iterative approach known as a level wise search. Apriori property suggests that all non empty subset of frequent item set must also be frequent. But Apriori algorithm is having certain limitations that requirement of scanning database many times. This issue is significant when we work with a small database, but when there are large database the idea of reading data repeatedly is very costly and will affect to the accuracy of algorithm. [1]

To improve efficiency of apriori algorithm there are several methods

1. Hash-based technique
2. Transaction reduction
3. Partitioning
4. Dynamic item set counting [2]

Among them here we concern to Hash-based technique. Hash-method often used an array structure to store database. By this deed, we are able to access database directly by using a key element instead of linear search. Our databases are growing quickly day after day while the storage devices are not. So, reading data a lot of times brings us a big difficult to process data when their size exceed limitation provided by hardware devices. Notice that hash method is not only useful to access directly to data, but also help us divide the original database into parts, each part fit in a limited space. We intend to use hash functions to hash item sets into another buckets of a hash table and we could reduce the size and even reduce the total task. [3]

II. RELATED WORK

In DHP (Direct Hashing and Pruning) there are two steps 1.Hashing 2.Pruning. In hashing scan the database count support for items and generate candidate-1 itemsets. Generate Large-1 itemsets according to minsup. To generate Large-2 itemset use the hash function to create hash table,

$$\mathbf{H(x, y) = [(x's\ order*10 + y's\ order]\ mod\ n}$$

Where 'n' is the hash table size and value for n is calculated as (m + c) and it must be prime, 'm' represents total number of items in the database for the hash and 'c' represents a real number which is added to make the prime number .

Repeat until all frequent items found. In pruning step we are removing the items which are less then minsup. First remove transaction then remove itemsets. DHP is having certain limitations that here results were generated quite quickly compare to

Apriori algorithm as size of candidate itemsets was reduced by hash method. However, there is still some weakness due to the collision in hash table and another drawback is that the process to generate candidate itemsets may be omit to filter out itemsets when they laid together in a bucket to make the entry value is greater than minsup.[4]

c. [3]

MPIP (Multi-Phase Indexing and Pruning): If a hashing function can be found that is one to one from the set of keys to the address space, it is a perfect hashing function. If a perfect hashing function maps from the set of keys to an address space with the same size, this perfect hashing function is called the minimal perfect hashing function. Although some researchers have proposed algorithms to generate non-collision hashing functions, the computation cost is very high owing to the use of large amount of prime number multiplications. To cope with this problem, the MPIP Algorithm employs a minimal perfect hashing function to produce no-collision hash tables to reduce the time needed for scanning the entire database and searching data items in a very large hash tree. [6]

c calculated as (2m + 1) and „m" is the number of items in the database.

If collisions take place, Quadratic Probing (QP) technique is employed to avoid it. Quadratic Probing [QP**]** Collisions are avoided by using Quadratic Probing technique. QP function is

$$\mathbf{H(k, i) = (H(k) + i^2) \bmod n}$$

Here in this algorithm there is certain drawback that hashing with Quadratic Probing technique places all items without any collision but the probing sequence volume is high. Because of the lengthy probing sequence, it takes more time to hash the collided itemsets. Secondary clustering occurs while using QP technique. In secondary clustering for the same initial hash value same sequence of numbers are repeated again. [7]

H-BAH (H-Bit Array Hashing) algorithm which is a hash-based technique mines the frequent itemsets without any collision in the hash table. The lengthy probing sequence and secondary clustering that exist with Quadratic Probing technique is avoided by using the H-Bit array in the header bucket of the hash table. The H-Bit array technique also shrinks the size of hash table, which is required for placing itemsets. But when we are generating second level of large item set at that time the hashing function which we are using is complex function and it is also creating the collisions. Then we have to apply the lengthy process of removing collision by H Bit array technique. Here the size of the hash table is also fix so wastage of memory space. [7]

N-Hash algorithm has some advantages. Its rationale is simple and clear, it can be carried out without any Hash function and will not encounter Hash colliding problem. N-Hash algorithm generate the appropriate bucket address just for the bucket content, the number of address is the most but necessary and the least but sufficient number needed to avoid Hash colliding, that is it is the best value. [8]

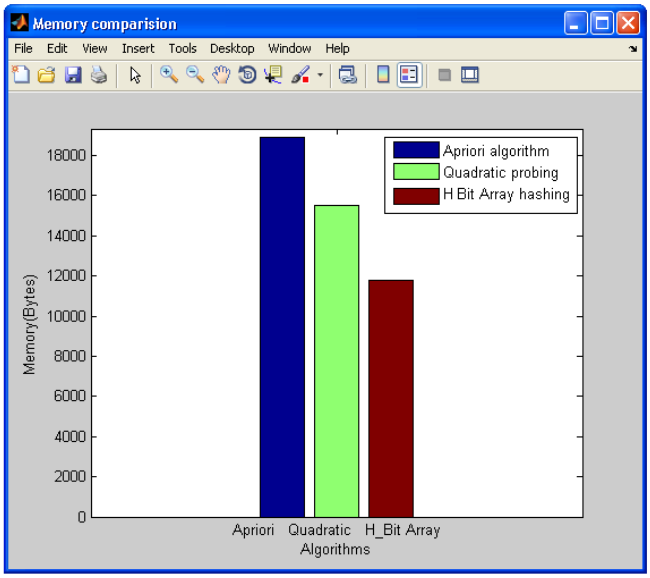

Fig 1 Graph representing the memory of the Retail Dataset

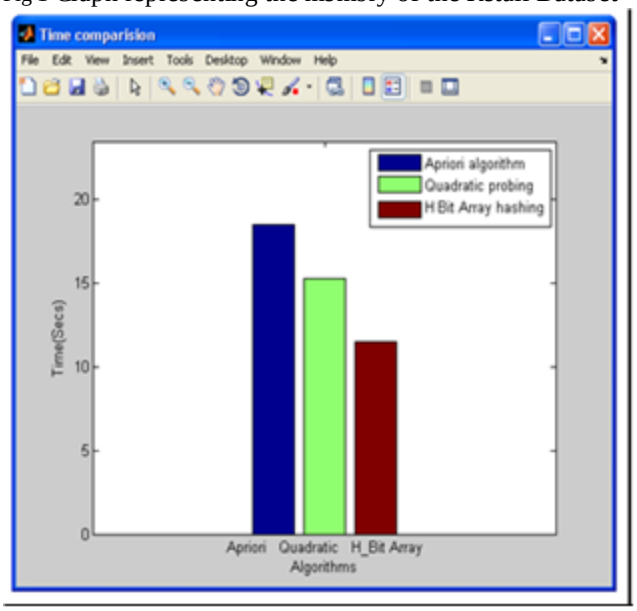

Fig 2 Graph representing the time of the Retail Dataset

III PROPOSED ALGORITHM

STEP 1: Create vertical format (Itemset, Tidset) of the initial transactional database.

STEP 2: Built a hash table based on number of items in the database.

H (k) = (order of item k) mod n

Where 'n' is the hash table size and value for n is calculated as (m + c) and it must be prime, 'm' represents total number of items in the database for the hash and 'c' represents a real number which is added to make the prime number .

STEP 3: Hash the candidate 1-itemset based on the hash function.

STEP 4: Add the H-Bit array to the first slot or bucket of the hash table.

STEP 5: If collision occurs, then use H-Bit array searching technique to map collided items.

STEP 6: Place all the collided itemsets and update its corresponding bit value.

STEP 7: Create the linked list with support count and Tidset for each stored item in the hash table.

STEP 8: Scan the hash table with linked list, generate the first level frequent itemsets which satisfies minimum support threshold.

STEP 9: To create the second level of candidate itemset use N- hash method.

9.1:scan the transaction take out the transaction one by one.(s , t) as bucket address,{Is , It} be itemset, n_{st} be the count of $\{I_s , I_t\}$.then if (s , t) does not belongs to A then $n_{st}=1$ else $n_{st}= n_{st}+1$. Repeat until all items visited.

STEP 10: On using minimum support threshold, generate the second level frequent itemset.

As we have seen H-bit array hashing algorithm is having many advantages but we can increase its efficiency if we use simple method when generating second level of large itemsets. N-hash method is simple and clear so we use this method for the generation of second level of frequent item sets. It can be carried out without hashing function. In the h-bit array hashing algorithm when we are generating the second level of candidate item set at that time hashing function is used but still it creates collision to map the items. So if we use N-Hash algorithm then the collision will not occur in the second level of candidate generation so we do not have to generate the lengthy process of creating H

bit array and searching according to H bit array searching technique. In the H-bit array hashing algorithm the size of the hash table is fixed so wastage of memory space. N-Hash algorithm generate the appropriate bucket address just for the bucket content, the number of address is the most but necessary and the least but sufficient number needed to avoid Hash colliding, that is it is the best value.

IV CONCLUSION

Mining association rules is one of the most used functions in data mining. The traditional Apriori algorithm iteratively defines the candidate and frequent itemsets. Hash-based itemsets counting technique can improve the performance of ARM. The proposed hash-based algorithm which we have created from H-bit array hashing algorithm and N-hash algorithm takes the advantages of both the algorithms. From H-bit array we have used H-bit array which gives the information of the hashed items and it also uses linked list representation to store frequent items so all the items can be searched frequently and from the N-hash algorithm we have used it to generate the second level of frequent item because this process is easy and simple then calculating lengthy process of hash function and N-hash also generate the appropriate bucket address just for the bucket content.

REFERENCES

[1] Tung-Cheng Hsieh and Tzone I Wang ”A mining-based approach on discovering courses pattern for constructing suitable learning path”, science Direct , 2009
[2] Data Mining Concepts and Techniques, Second Edition Jiawei Han *University of Illinois at Urbana-Champaign* Micheline Kamber, ch 5.
[3] Le Kim Thur, “HASH-BASED APPROACHTO DATA MINING”
[4] Jong soo park, ming syan chen, Philip S.Yu,”an efficient hash based algoritham for mining association rules”.
[5] Hassan Najadat, Amani Shatnawi and Ghadeer Obiedat“A New Perfect Hashing and Pruning”2011
[6] Chuang-Kai Chiou, Judy C. R. Tseng “An Incremental Mining Algorithm for Association Rules based on Minimal Perfect Hashing and Pruning”.
[7] Padmavathy,and V.Umarani “An Efficient Association Rule Mining Using the H-BIT Array Hashing Algorithm” International Journal of Advanced Research in Computer Science and Software Engineering.
[8] Chen Yong-ming, Zhu Mei-ling “Sampling Based N-Hash Algorithm for searching frequent items”