e-ISSN: 2349-9745
p-ISSN: 2393-8161
Scientific Journal Impact Factor (SJIF): 1.711
International Journal of Modern Trends in Engineering and Research
www.ijmter.com

Frequent Pattern Mining Based on Maximum and Minimum Count Approach for Different Computing Environments

Parag Moteria[1], Dr Y.R.Ghodasara[2]
[1]*Ph.D. Scholar, School of Computer Science, R K University, Rajkot and Assistant Professor, MCA Department, ISTAR, Vallabh Vidyanagar*
[2]*College of Agricultural Information Technology, Anand Agricultural University, Anand*

Abstract—Data mining is the process of finding interesting trends or patterns in transactional datasets to discover knowledge and helps in decision making process. Frequent patterns are frequent data set in transactional dataset. Discover interesting associated frequent items among large itemsets by specifying a user defined support threshold without any domain knowledge leads small or large numbers of uninterested results, is not appropriate. Without specifying minimum user defined support threshold is high cost effective process, but it helps to conclude interesting frequent itemsets without any domain knowledge. We propose new algorithm, RCRS (Reduce Candidate Itemsets and Reduce Scanning of dataset) based on Itemwise Cardinality Matrix, Maximum Item Count, and Minimum Item Count approach implement on three different types of computing environment. This RCRS algorithm has multiple features: only single scan is required to form Itemwise Cardinality Matrix, Maximum Item Count and Minimum Item Count derived from Itemwise Cardinality Matrix, Minimum Item Count table helps to determine implicit minimum support count threshold, and Maximum Item Count helps to reduce items in candidate set and numbers of scan to determine frequent itemsets in transactional dataset both. Centralized transactional dataset computing on demand by distributed node and achieve scalability with faster result using distributed computing environment. Experiment shows RCRS is more efficient than Apriori.

Keywords- Attribute/Characteristic Table, Distributed Data Mining, Frequent Itemset Mining, Itemwise Cardinality Matrix, Maximum Item Count, Minimum Item Count, With and Without User Defined Minimum Support Count Threshold

I. INTRODUCTION

Data mining refers to extracting knowledge from large amounts of data that also carries similar or slightly different meaning to data mining, such as knowledge mining from data, frequent pattern analysis, and data archaeology [1]. Frequent pattern analysis, as the name suggests, are patterns that can occur frequently in data [1]. The concept of frequent itemset was first introduced for mining transaction datasets (Agrawal et al. 1993) [2]. Agrawal and Srikant (1994) observed an interesting downward closure property, called Apriori, among frequent k-itemsets: A k-itemset is frequent only if all of its sub-itemsets are frequent [3]. This process iterates until no more frequent k-itemsets can be generated for some k. Let $I = \{i_1, i_2, \dots, i_n\}$ be a set of all items. A k-itemset α, which consists of k items from I, is frequent if α occurs in a transaction dataset D no lower than $\theta |D|$ times, where θ is a user specified mining support threshold and $|D|$ is total number of transaction in D. Section (III-A) shows single pass process to collect itemsets in item wise cardinality matrix format that helps to determine frequent pattern itemsets in D. Hence, frequent pattern itemsets that appear in data set with frequency not less than minimum support threshold [4]. However, selection of such minimum support threshold is typically hard [5]. Because of without domain knowledge, threshold may be small or large, as a result of frequent pattern itemsets may be too large or small number of result or no result respectively. Implementation of Section (III-B) and Section (III-C) help to determine reasonable minimum support threshold and interesting k itemsets. Section (III-D) interprets to reduce itemsets in candidate set and scanning process to validate itemset with prior knowledge using Table

3 & 4. Section (III-E) depicts implicit and explicit minimum support count threshold. Frequent pattern mining algorithm under distributed computing environment with centralized transactional datasets provides an efficient use of multiple processors to speed up the process to determination of frequent pattern itemsets. Section (III-F) discuss distributed frequent pattern mining for n numbers of nodes spread over network, computing centralized transactional dataset on demand using remote method invocation.

II. RELATED WORK

A. Apriori Algorithm

Large number of distinct single items in synthetic order transactional dataset, may form huge numbers of combinations of itemsets [2], [3], [6]. Agrawal and Srikant (1994) developed scalable methods for frequent pattern mining based on interesting downward closure property, called Aprior, among frequent k – itemset [03]. Apriori scans the dataset k times to mine frequent k – itemset. This is the essence of the Apriori algorithm (Agrawal and Srikant 1994) and its alternative (Mannila et al. 1994) [4], [6].

B. Minimum Support Count Threshold

Assumed minimum support threshold without domain knowledge is typically hard to determine interested frequent pattern itemsets [05]. Either assumed minimum support threshold is large or small, due to assumed minimum support threshold may form no result or very large frequent pattern uninterested itemsets respectively. Section–3.5 is helpful to determine appropriate minimum support count threshold for interested frequent pattern itemsets, while appropriate support threshold is not provided.

III. PROPOSED WORK

Proposed algorithm determines k interesting frequent itemsets regarding either implicit or explicit minimum support threshold. Following steps achieve k interesting frequent itemsets (Consider, synthetic order transactional dataset – Table 1):

TABLE 1.

TID	Items
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2, I3, I5
9	I1, I2, I3

A. Generate Itemwise Cardinality Matrix

Scan this synthetic order transactional dataset single time, proposed algorithm forms item wise cardinality matrix as per the following:

TABLE 2.

Cardinality	**Items**				
	I1	**I2**	**I3**	**I4**	**I5**
1	0	0	0	0	0
2	2	3	4	1	0
3	3	3	1	1	1
4	1	1	1	0	1

In Table 2, cardinality depicts numbers of items per transactional itemset, for example, interpretation of third row and second column:
Cardinality = 2, Items = I1 and Cell Value = 2 represents count of I1 is two times in transactions containing I1, those have total two items in itemset, i.e. TID=5 and TID=7.

B. Minimum Item Count Table

Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Apriori property follows two actions: join and prune [01]. To find L_k, make a join of L_{k-1} ($k \geq 2$) and (k-2) items are in common.
To form Table 3, go through following operation on Table 2.
Consider that, if it is L_k then ${}^{k}C_{(k-1)}$ unique combinations with (k-1) – itemset. Join operation for any two unique (k-1) – itemsets form L_k. Minimum count of each item in unique combinations of (k-1) – itemset to achieve L_k, is always one per k - itemset.

TABLE 3.

Cardinality	Items				
	I1	**I2**	**I3**	**I4**	**I5**
1	6	7	6	2	2
2	6	7	6	2	2
3	4	4	2	1	2
4	1	1	1	0	1

Derivation of Table 3:
Consider TID=9, Itemset is {I1, I2, I3, I5}. With reference of Apriori Join and Prune operations, to generate this itemset, it requires join operation between following possible uniqe combinations: {I1, I2, I3} , {I1, I2, I5} , {I1, I3, I5} and {I2, I3, I5}. From these unique combinations minimum count for each item is one.

C. Maximum Item Count Table

If it is L_k, add (k-1) – itemset in L_{k-1} to L_1, per one itemset of k – itemset.

TABLE 4.

Cardinality	Items				
	I1	**I2**	**I3**	**I4**	**I5**
1	6	7	6	2	2
2	11	12	9	3	5
3	6	6	4	1	4
4	1	1	1	0	1

Derivation of Table 4:
Consider TID=9, Itemset is {I1, I2, I3, I5}. With reference of Apriori Join and Prune operations, {I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, and {I2, I3, I5} unique combinations are derived using this itemset. Therefore, in Cardinality = 3, Item=I1, Maximum Count = 3+(1*3) = 6.

D. Interpretation of Table 3 and Table 4

In Table 3 & Table 4, rows represent cardinality and columns represent unique items occur in transactional dataset.
Table 3 & 4, row cardinality = 1, higheset item count=7. If minimum support count threshold is greater than one less than highest item count, no frequent itemset will be generated.
Table 3 & 4 show minimum and maximum item count range. Either Table 3 or Table 4 helps to reduces items in candidate item set, to find k interesting itemsets. For example: in Table 4, consider minimum support count threshold is two, using cardinality row = 3, count of item I4 = 1, there is no need to include I4 into L3. Hence, as a result our proposed algorithm reduces items in candidate

itemset, achieved through following pseudo code.

```
Input:
D, a transactional dataset
min_sup, the minimum support count
                threshold [01]
Output:
L, frequent itemsets in D
procedure candidate_gen (Lk-1:frequent (k-1) – itemsets)
for each itemset l1 Є Lk-1
    for each itemset l2 Є Lk-1
    if(l1[1]= l2[1]) Λ … Λ (l1[k-1] < l2[k-1]) Λ  (l2[k-1] ≥ min_sup) then
       // use prior knowledge
      // reduce itemset in Ck
     {
          c = l1 X l2;
          add c to Ck;
     }
return Ck;
```

After generating candidate set; scan transaction dataset for validating occurrence of candidate itemsets within transactions in transactional dataset for interesting k-itemset. For example: consider cardinality row = 2 and minimum support count threshold is two, (a) if itemset is {I3, I4}, but there is no occurrence of this 2-itemset and (b) if itemset is {I1, I2}, to validate this 2-itemset, scan transactional dataset, TID=1 and 4 consists {I1, I2}, it satisfies minimum support count threshold, hence this 2-itemset is frequent and stop scanning of transactional dataset. As a result, our proposed algorithm reduces scanning of transactional dataset, achieved through following pseudo code.

```
Input:
D, a transactional dataset
min_sup, the minimum support count
                threshold [01]
Output:
L, frequent itemsets in D
L1 = find_frequent_1-itemsets(D);
for(k=2;Lk-1 ≠ Ø;k++){
    Ck = candidate_gen(Lk-1);
    for each transaction t Є D{
        Ct = subset(Ck, t);
        for each candidate c Є Ct{
            c.count++;
            if(c.count ≥ min_sup)
                break;
            // reduce scanning of whole D
        }
    }
    Lk = {c Є Ct | c.count ≥ min_sup}
}
return L = UkLk;
```

Convert Table 4 into Attribute/Characteristic table:

TABLE5.

Cardinality	Items					Total
	I1	I2	I3	I4	I5	
1	6	7	6	1	3	23
2	11	12	9	2	5	40
3	6	6	4	1	4	21
4	1	1	1	0	1	4
Total	24	26	20	6	12	

Future aspect of Table V determines joint probability, marginal probability, conditional probability and transitivity property in conditional probability can reduce scanning of transactional dataset.

E. With and Without User Defined Minimum Support Threshold

Discover interesting associated k frequent items among huge numbers of itemsets by specifying a user defined support threshold without any domain knowledge leads small or large numbers of uninterested results, is not appropriate. Without specifying minimum user defined support threshold is high cost effective process, but it helps to conclude interesting frequent itemsets without any domain knowledge.

One of feature of our proposed algorithm is to find minimum support threshold while domain knowledge is not suffice to seek k frequent pattern itemset using following procedure:

Consider Table 3, determine row wise minimum, is greater than one, because of definition of frequent item. In each row minimum count greater than one is {3, 2, 2}. Now, determine lowest value from row minimum, it depicts final minimum support threshold to determine frequent pattern itemset within transactional dataset.

F. Different Computing Environment

Implementation of our proposed algorithm on three different computing systems:

- Single System Single Core
- Single System Multi Core
- Distributed Computing Environment

Objective of a distributed computing system is to connect users and resources in a transparent, open and scalable way [07]. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more networked computers, which communicating and coordinating work among concurrent processes by passing messages. Tightly coupled parallel computing and loosely coupled distributed computing may be seen in the same system. The processors in a typical distributed system run concurrently in parallel [08].

Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts [09].

Consider client-server model in Figure 1, with logically n numbers of clients, to determine frequent pattern mining using distributed computing environment.

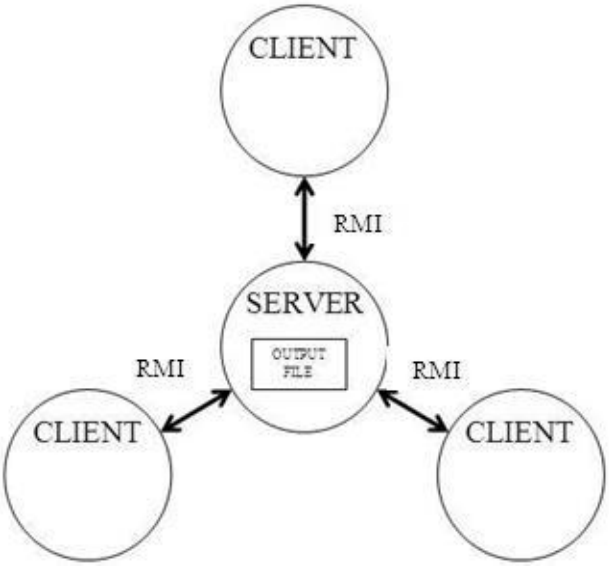

Figure 1. n Nodes are communicate with each other in Distributed Computing Environment using RMI

IV. RESULT

This paper shows computing time results for Apriori Algorithm and RCRS Algorithm, implements on Single System Single Core, Single System Multi Core and Distribute Computing Environment. Experiments are performed on two datasets:

1. Consumer Complaint Dataset with one lack fifty thousand records [10]
2. T40I10D100K dataset was generated using the generator from the IBM Almaden Quest research group with one lack records [11].

All system specifications for different computing environments are listed in Table 6.

TABLE 6.

Dataset	System Specifications
Consumer Complaint & T40I10D100K Single System Single Core	Window 7 Professional, Intel Pentium (R) Core (TM) i-3 CPU 370@2.4 GHz 2 GB RAM, 1.8 GB usable Java 1.6.0
Consumer Complaint & T40I10D100K Single System Multi Core	Window 7 Professional, Intel Pentium (R) Core (TM) i-3 CPU 370@2.4 GHz 2 GB RAM, 1.8 GB usable Java 1.6.0
Consumer Complaint & T40I10D100K Distributed Computing	Server: Window 7 Professional Intel Pentium (R) Core (TM) i-3 CPU 370@2.4 GHz 2 GB RAM, 1.8 GB usable Java 1.6.0 Workstation-1: Window 7 Enterprise Intel Pentium (R) CPU B960@2.2 GHz 4 GB RAM, 2.58 GB usable Java 1.6.0_12 Workstation-2: Window 7 Ultimate Pentium (R) Duo-Core CPU T4400@2.2 GHz 1 GB RAM Java 1.7.0_25
Consumer Complaint Distributed Computing	Server: Window 7 Professional Intel Pentium (R)

	Core (TM) i-3 CPU 370@2.4 GHz 2 GB RAM, 1.8 GB usable Java 1.6.0 Workstation-1: Window 7 Enterprise Intel Pentium (R) CPU B960@2.2 GHz 4 GB RAM, 2.58 GB usable Java 1.6.0_12 Workstation-2: Window 7 Ultimate Pentium (R) Duo-Core CPU T4400@2.2 GHz 1 GB RAM Java 1.7.0_25 Workstation-3: Window 7 Ultimate Pentium (R) Duo-Core CPU T4400@2.2 GHz 1 GB RAM Java 1.7.0_25 Workstation-4: Window XP Intel Core 2 Duo CPU T6570@2.1 GHz 3 GB RAM Java 1.6.0_17
T40I10D100K Distributed Computing	Server: Window 7 Professional Intel Pentium (R) Core (TM) i-3 CPU 370@2.4 GHz 2 GB RAM, 1.8 GB usable Java 1.6.0 Workstation-1: Window 7 Enterprise Intel Pentium (R) CPU B960@2.2 GHz 4 GB RAM, 2.58 GB usable Java 1.6.0_12 Workstation-2: Window 8 Enterprise Intel Pentium (R) Core (TM) i-5 3210M CPU @2.5 GHz 4 GB RAM, 3.79 GB usable Java 1.6.0_26 Workstation-3: Window 7 Ultimate Pentium (R) Duo-Core CPU T4400@2.2 GHz 1 GB RAM Java 1.7.0_25 Workstation-4: Window XP Intel Core 2 Duo CPU T6570@2.1 GHz 3 GB RAM Java 1.6.0_17

In experiments, percentage support count ranges from 0.1 to 90 of total records. Following resultant graphs show computation time for percentage support count ranges from 0.1 to 10 and percentage support count ranges from 15 to 90.

A. Consumer Complaint dataset results

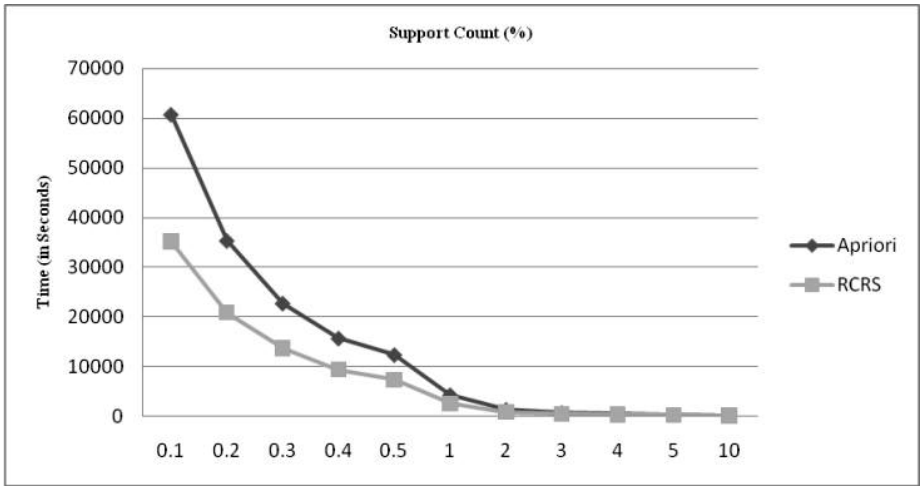

Figure 2.

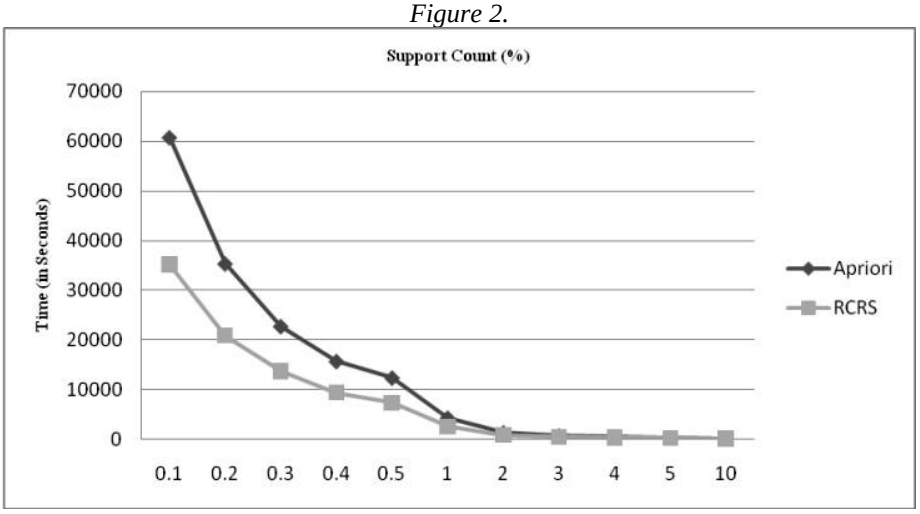

Figure 3.

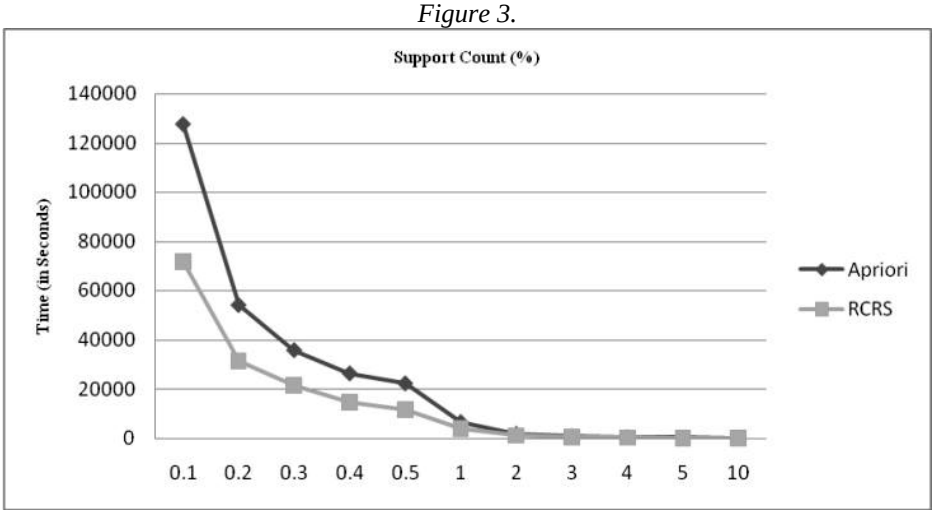

Figure 4.

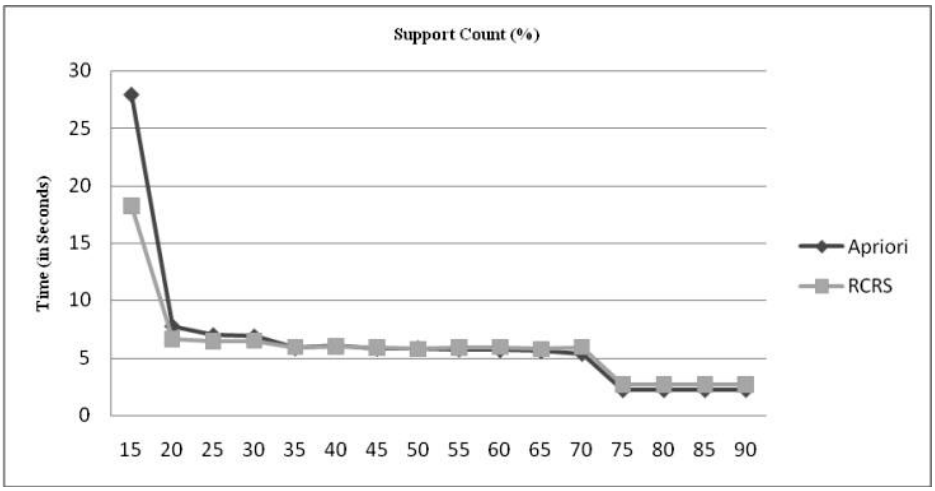

Figure 5.

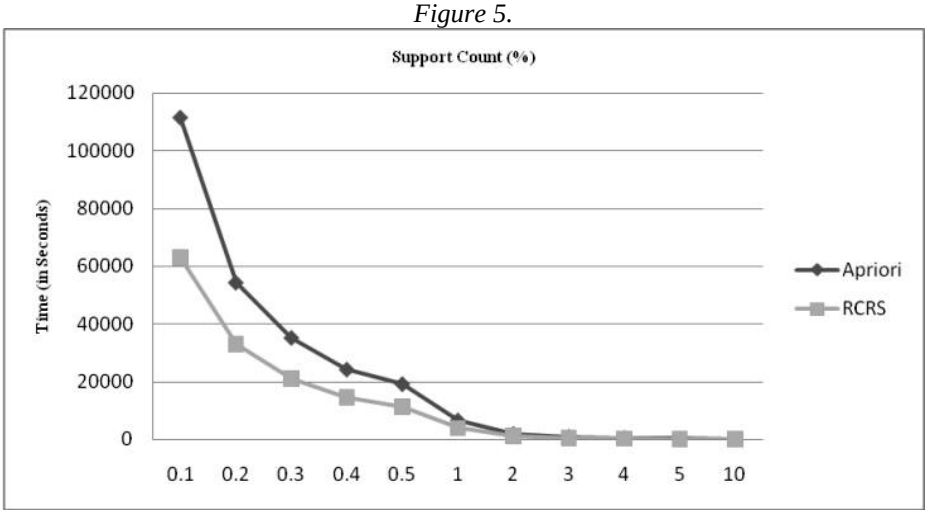

Figure 6.

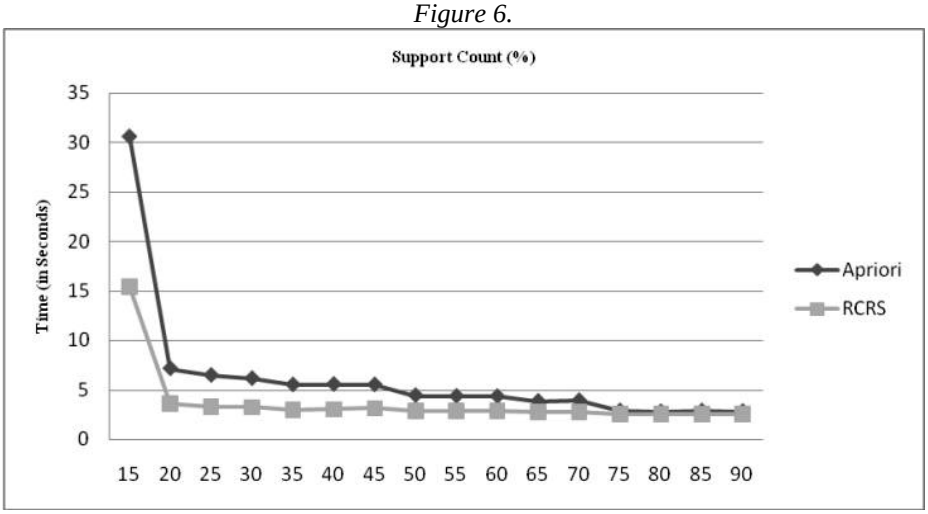

Figure 7.

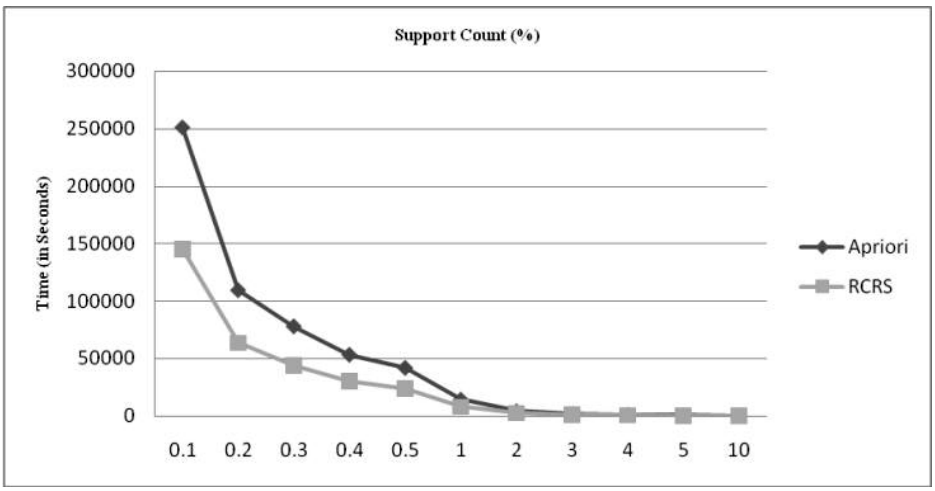

Figure 8.

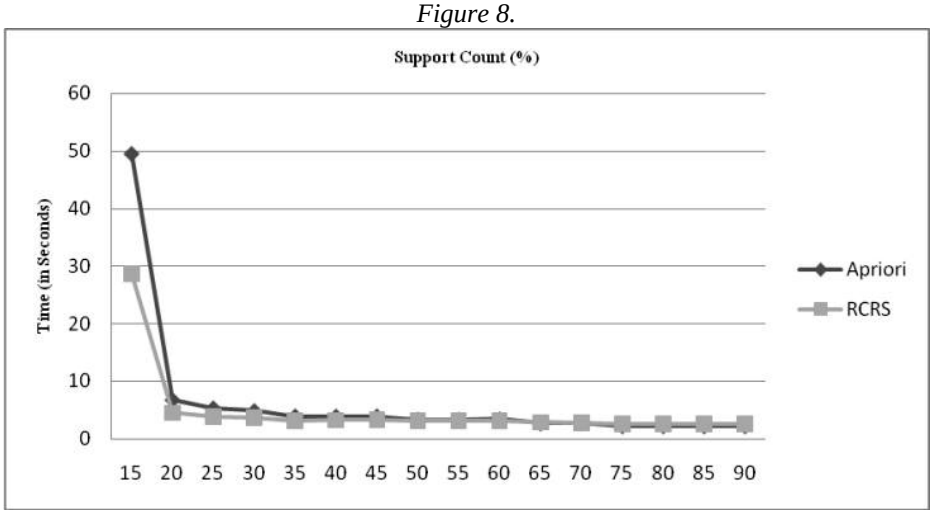

Figure 9.

B. T10I4D100K dataset results

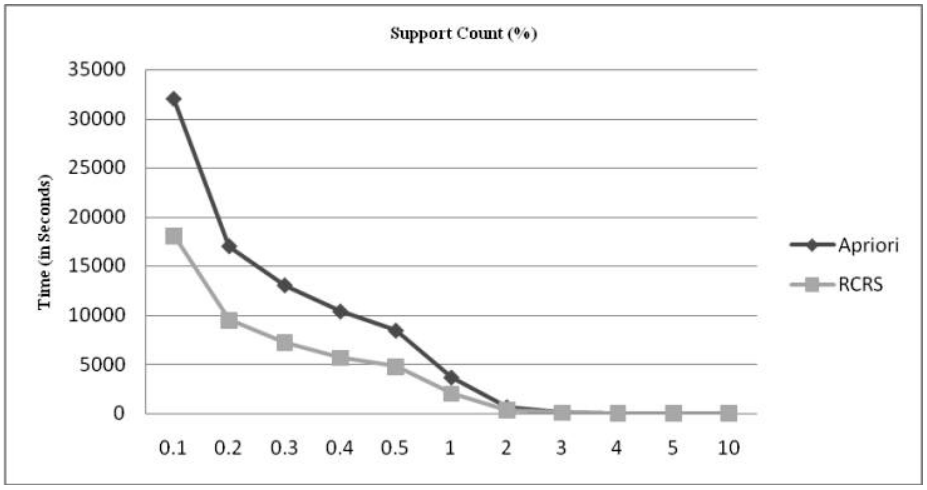

Figure 10.

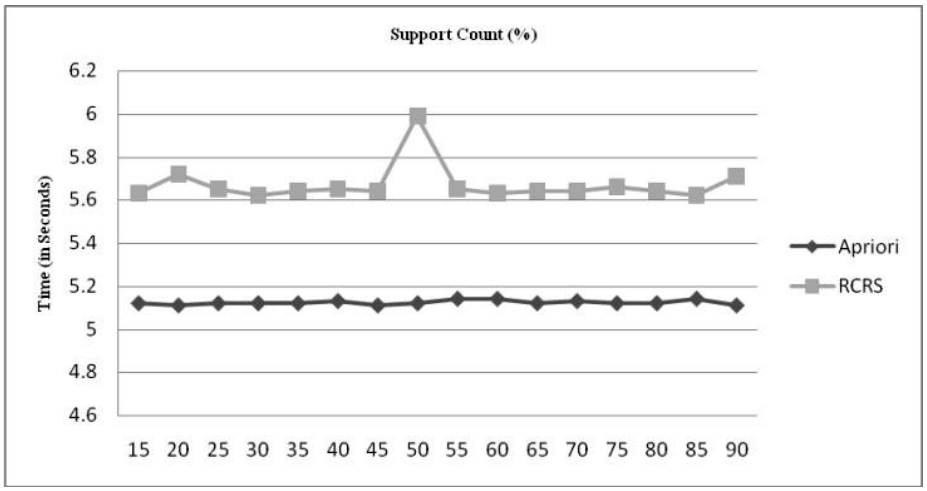

Figure 11.

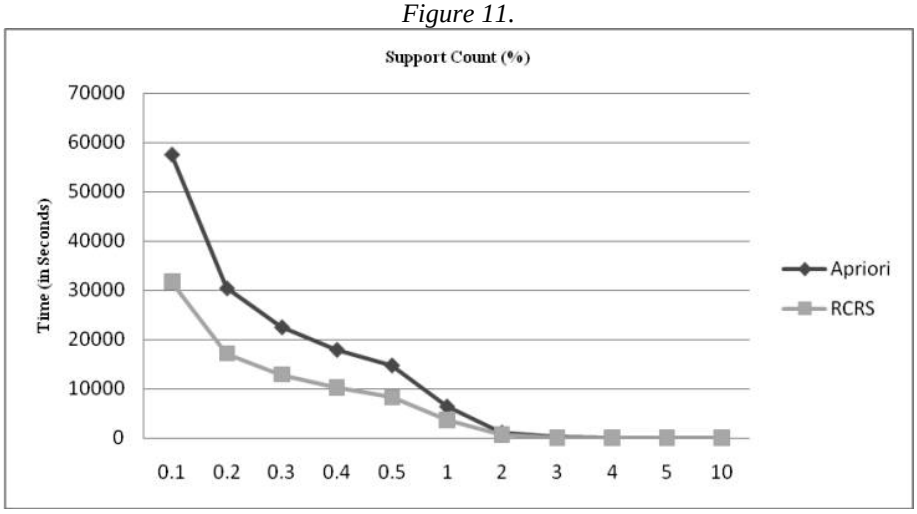

Figure 12.

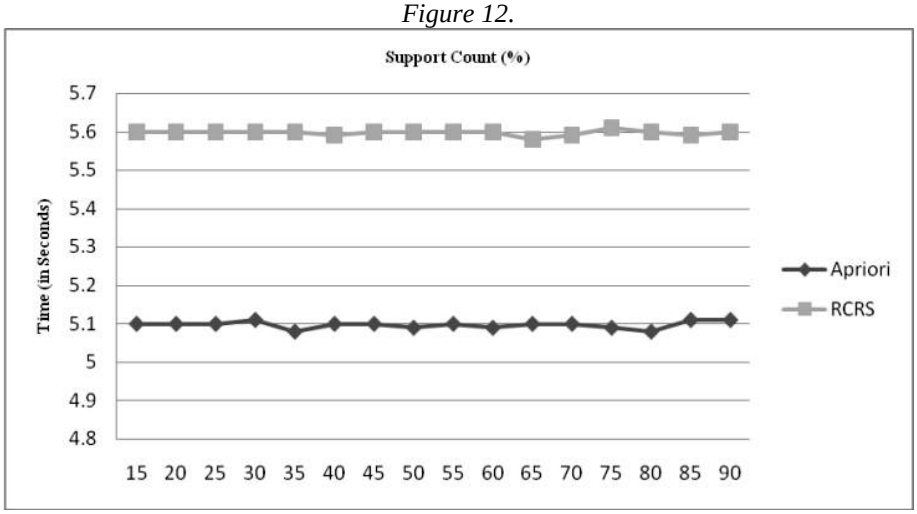

Figure 13.

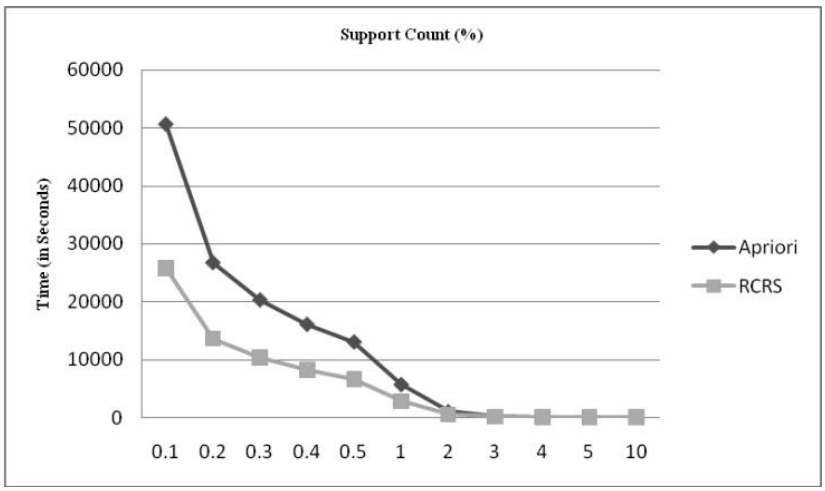

Figure 14.

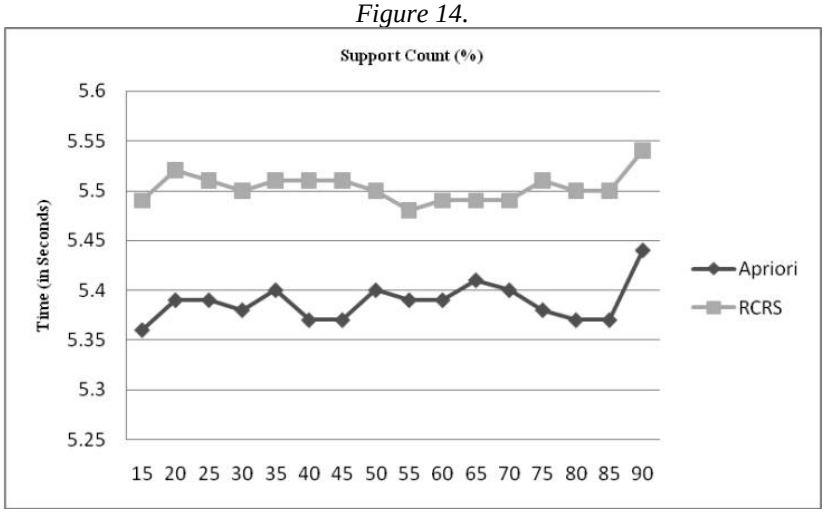

Figure 15.

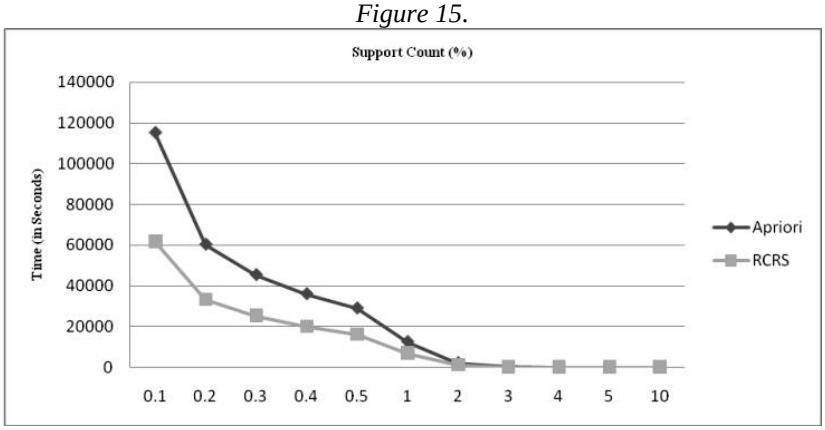

Figure 16.

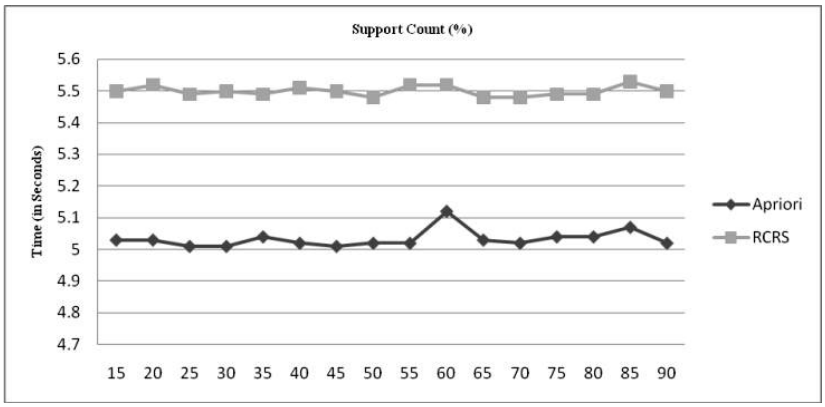

Figure 17.

V. CONCLUSION

Experiments show RCRS is more efficient than Apriori Algorithm. Computation time for RCRS and Apriori Algorithm is almost equal for support count (%) from 15 to 90. It can reduce this difference, to form Itemwise Cardinality Count Matrix and store it for future until transactional dataset is unchanged. Hence, only one time computation requires for Itemwise Cardinality Count Matrix for same transactional dataset.

REFERENCES

[1] Jiawei Han and Micheline Kamber, "Introduction" and "Mining Frequent Patterns, Associations and Correlations", in *Data Mining: Concepts and Techniques*, 2nd ed., Morgan Kaufmann Publishers, pp. 1 – 39, 227 – 240

[2] Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993ACM SIGMODinternational conference on management of data (SIGMOD'93), Washington, DC, pp 207–216

[3] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 1994 international conference on very large data bases (VLDB'94), Santiago, Chile, pp 487–499

[4] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan, "Frequent Pattern Mining: current status and future directions", Springer Science+Business Media LLC 2007, Published online: 27 January 2007

[5] Yin-Ling Cheung, Ada Wai-Chee Fu, "Mining Frequent Itemsets without Support Threshold: With and without Item Constraints", *IEEE Tran. On Knowledge and Data Engineering*, vol. 16, pp. 1 – 18 , June 2004

[6] Mannila H, Toivonen H, Verkamo AI (1994) Efficient algorithms for discovering association rules. In: Proceeding of the AAAI'94 workshop knowledge discovery in databases (KDD'94), Seattle, WA, pp 181–192

[7] http://www.b-u.ac.in/sde_book/distrib_computing.pdf

[8] http://en.wikipedia.org/wiki/Distributed_computing

[9] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

[10] https://catalog.data.gov/dataset?q=Consumer+Complaints&sort=score+desc%2C+name+asc&ext_location=&ext_bbox=&ext_prev_extent=-254.53125%2C-75.84516854027044%2C54.84375%2C83.19489563661588

[11] http://fimi.ua.ac.be/data/T40I10D100K.dat

