An Improved Distributed Association Rule Algorithm

Dr. Saad K. Majeed*& Dr. Hussein K. Abbas*

Received on: 23/2/2010
Accepted on: 2 /9/ 2010

Abstract

All Distributed association rules mining (DARM) algorithms which bases on Apriori algorithm don't have an efficient message optimization technique, so they exchange numerous messages during the mining process which needs several distributed scan operations to the distributed warehouses or distributed databases to get the support values, also the performance of these DARM algorithms decreased with increasing communication cost especially when increasing the number of distributed mining sites as well as the itemsets to be mined become more larger .

The aim of this work is to improve association rules in distributed data mining by proposing a new efficient method of distributed association rule mining, which reduce the average size of records transferred, datasets and messages transferred without need to any distributed scan to the distributed data warehouses or distributed databases to retrieve the values of the support values of these datasets. The results obtained from the proposed method prove that the proposed method is better than the existing algorithms by reducing communications costs, centralstorage requirements, enhance performance and achieves high degree of scalability compared with the existing algorithms.

Keywords: distributed association rule mining, datasets, Apriori algorithm, support.

خوارزمية علاقة ترابطية موزعة محسنة

الخلاصة

كل خوارزميات تعدين االعلاقات الترابطية الموزعة والمعنمدة على مبدأ خوارزمية الابرايوري لاتمتلك تقنيات كفوءة لتحقيق امثلية تناقل الرسائل، لذلك فانها تتطلب تبادل العديد من الرسائل اثناء عملية التعدين والتي تحتاج الى القيام بالعديد من عميات المسح الموزع لمخازن البيانات الموزعة او قواعد البيانات الموزعة لاسترجاع قيم الدعم ، كذلك فان اداء هذه الخوارزميات يتناقص مع زيادة كلف الاتصال وخصوصا عند زيادة عدد مواقع التعدين الموزعة بالاضافة الى ذلك عندما يصبح حجم العناصر المراد تعدينها اكبر.

الهدف من البحث هو تحسين العلاقات الترابطية في تعدين البيانات الموزعة عن طريق استحداث طريقة كفوءة لتعدين العلاقات الترابطية الموزعة، تتولى تقليل حجم معدل القيود المنقولة، مجاميع البيانات والرسائل المتبادلة دون الحاجة الى اجراء مسح موزع لمخازن البيانات الموزعة او قواعد البيانات الموزعة لاسترجاع قيم الدعم الخاص بمجموعة البيانات. النتائج المستحصلة من الطريقة المقترحة تبين افضلية عملها مقارنة بما هو موجود من الخوارزميات الموزعة وذلك بتحقيقها تقليل

*Computer Science Department, University of Technology/ Baghdad

https://doi.org/10.30684/etj.28.18.7
2412-0758/University of Technology-Iraq, Baghdad, Iraq

لكلفة الاتصال، متطلبات الخزن المركزي، وقت الحسابات، تحسين الاداء وتطبيق درجة عالية مـــن
التوسعية مقارنة بما هو موجود من الطرق.

1- Introduction

Association rule mining, one of the most important and well researched techniques of data mining, was first introduced by Agrawal, Imielinski, and Swami [1].

The discovery of “association rules” in databases may provide useful background knowledge to decision support systems, selective marketing, financial forecast, medical diagnosis, and many other applications. As the size of a database to be mined can be very large, parallel computation techniques have also been explored. Consider that in a distributed organization, the database may be allocated through a computer network. This leads to a real demand for developing distributed computation techniques in data mining [2].

Mining association rules is an important data mining problem. Association rules are usually mined repeatedly in different parts of a database. Current algorithms for mining association rules work in two steps.

1. Discover the large itemsets, i.e. the sets of itemsets that have support above a predetermined minimum support σ.

2. Use the large itemsets to generate the association rules for the database.It is noted that the overall performance of mining association rules is determined by the first step, which usually requires repeated passes over the analyzed database and determines the overall performance. After the large itemsets are identified, the corresponding association rules can be derived in a straightforward manner [2]. One of the most popular techniques is association rule mining (ARM), which is the automatic discovery of pairs of element sets that tend to appear together in a common context.

Unfortunately, most Association Rule Mining (ARM) algorithms focus on a sequential or centralized environment where no external communication is required. Distributed ARM algorithms, on the other hand, aim to generate rules from different data sets spread over various geographical sites; hence, they require external communications throughout the entire process [3].

2- Distributed Association Rules Mining

2-1 Association rules concept

An association rule is a simple probabilistic statement about the co-occurrence of certain events in a database, and is particularly applicable to sparse transaction data sets [4]. An association rule is a rule, which implies certain association relationships among a set of objects (such those which occur together or one implies the other”), in a database [5]. Association mining works as follows:

Let I be a set of items and D a database of transactions, where each transaction has a unique identifier (t_{id}) and contains a set of items called an itemset. An itemset with k items is called a k-itemset. The support of an itemset X, denoted S(X), is the number of transactions in which that itemset occurs as a subset. A k-subset is a k-length subset of an itemset. An itemset is frequent or large if its support is more than a user-specified minimum support (min_sup) value. F_k is the set of frequent k-itemsets. A frequent itemset is maximal if it is not a subset of any other frequent itemset.

An association rule is an expression A ⇒ B, where A and B are itemsets. The rule's support (S) is the joint probability of a transaction containing both A and B, and is given as S(A ⇒ B). The confidence of the rule is the conditional probability that a transaction contains B, given that it contains A and is given as $S(A \cup B)/S(A)$. A rule is frequent if its support is greater than min_sup and strong if its confidence is more than a user-specified minimum confidence (min_conf).

Data mining involves generating all association rules in the database that have a support greater than min_sup (the rules are frequent) and that have a confidence greater than min_conf (the rules are strong) [6].

The important measures for association rules, support (S) and confidence (C) can be defined as:

<u>Definition1</u>: Support (S)

$\text{Support}(X, Y) = \Pr(X \cup Y) = \text{count of } (X \cup Y) / \text{Total transactions}$ [8].

The support (S) of an association rule is the ratio (in percent) of the records that contain ($X \cup Y$) to the total number of records in database. Therefore, if we say that the support of a rule is 5% then it means that 5% of the total records contains ($X \cup Y$) [7].

<u>Definition2</u>: Confidence (C)

$\text{Conf}(X \Rightarrow Y) = \Pr(Y \setminus X) = \Pr(X \cup Y)/\Pr(X) = \text{support}(X , Y)/\text{support}(X)$ [8].

For given number of records, confidence (C) is the ratio (in percent) of the numbers of records that contain ($X \cup Y$), to the number of records that contain X. thus, if we say that a rule has a confidence of 15% it means that 85% of the records containing X also contain Y. The confidence of rule indicates the degree of correlation in the database between X and Y. Confidence is also a measure of rules strength. Mining consists of finding all rules that meet the user-specified threshold support and confidence [7]. As there are two thresholds, we need two processes to mine the rules [8]. The first step is to get the large itemsets. It finds all the itemsets whose support is larger than the support threshold. An itemset is the set of the items. Based on the large itemsets, we can generate the rules from the large itemsets, which is the second step. Rules that satisfy both a minimum support threshold (minsup) and a minimum confidence threshold (minconf)*are called strong* [9].

2-2 Background

Mining Association Rules is one of the most used functions in data mining. Association rules are of interest to both database researchers and data mining users. Since 90s, different approaches of data mining have been proposed for discovering useful knowledge from very large datasets [10]. A survey of previous research in the area is provided below.

- In 2001 Schuster and Wolff proposed a distributed algorithm called The Distributed Decision Miner (DDM), this algorithm belongs to the group of Apriori-based algorithms assuming a shared-nothing architecture as well. Here, after local frequency counts are computed on each node, the nodes perform a distributed decision protocol in each round in order to determine the set of globally frequent itemsets [11].
- In 2001 Zaiane et al. proposed a parallel algorithm that is based on frequent pattern –grouth algorithm(fp-growth) .The algorithm is called MLFPT (Multiple Local Frequent Pattern Tree). It assumes shared-memory architecture. Just like the centralized fp-growth algorithm, MLFPT does not generate candidates for frequent itemsets but instead builds multiple frequent pattern trees (FP-trees) [12].
- In 2003 Otey et al. proposed an algorithm named ZigZag. This algorithm assumes a shared-nothing architecture and a setting where the data is initially distributed on different sites (like network data for intrusion detection) [13].
- In 2003 Schuster, Wolff, et al. proposed a distributed sampling algorithm called D-Sampling. This algorithm is a combination of a centralized sampling algorithm and the DDM algorithm Schuster and Wolff presented in 2001. It assumes a shared-nothing architecture. D-Sampling assumes a centralized dataset and distributes it during runtime. Each node gets the"responsibility" for a set of items. The algorithm loads a sample of the dataset into memory. This sample is then distributed according to the responsibility of the different nodes, fragmenting the dataset vertically [14].
- In 2005 Emad Kadum Jabbar proposed algorithm called Association Rule with logical AND operation, which aims to produce association rules depending on logical AND operation by convert the database transaction into binary representation and neglecting any sum (column) less than threshold to find identical column in (k-1)-itemset table with column in k-itemset table which represents the association rules [15].
- In 2005 Claudio Silvestri, Salvatore Orlando proposed algorithm called Distributed Approximate Mining of Frequent Patterns. The proposed algorithm consists in the distributed exact computation of locally frequent itemsets and an effective method for inferring the local support of locally unfrequented itemsets [16].
- In 2007 Rawia Tahrir Salih proposed a new algorithm for distributed association rules called

Extracting Association Rules for Distributed Association Rules (EAR4DAR) Algorithm; which aims to extract association rules for distributed association rules instead of extracting association rules from huge quantity of distributed data at several sites, and that is through collecting the local association rules from each site and storing them, these Local Association Rules turn in series of operations to produce global association rules over distributed systems [17].

- In 2007 Lamine M. Aouad, Nhien-An Le-Khac and Tahar M. Kechadi, proposed a distributed algorithm for frequent itemsets generation on heterogeneous clusters and grid environments called Distributed Frequent Itemsets Mining in Heterogeneous Platforms. The proposed approach uses a dynamic workload management through a block-based partitioning, and takes into account inherent characteristics of the Apriori algorithm related to the candidate sets generation [18].

2-3 Count Distribution algorithm (CD)

It is an adaptation of the Apriori algorithm in the parallel case. At each iteration, it generates the candidate sets at every site by applying the Apriori-gen function to the set of large itemsets found at the previous iteration. Every site then computes the local support counts of all these candidate sets and broadcasts them to all the other sites. Subsequently, all the sites can find the globally large itemsets for that iteration, and then proceed to the next iteration. This algorithm has a simple communication scheme for count exchange. However, it also has the similar problems of higher number of candidate sets and larger amount of communication overhead. CD algorithm Divide the database evenly into horizontal partitions among all site and each site scans its local database partition to collect the local count of each item, then all sites exchange and sum up the local counts to get the global counts of all items and find frequent 1-itemsets, then all sites generate candidate k-itemsets from the mined frequent (k-1) itemsets and each site scans its local database partition to collect the local count of each candidate k-itemset and all sites exchange and sum up the local counts into the global counts of all candidate k-itemsets and find frequent k-itemsets among them. The process will be Repeated with k = k + 1 until no more frequent itemsets are found [19].

2-4 The Proposed System

We present a new method that addresses the issue of discovering the most frequently occurring sets of items. Our method divides the database into partitions and discovers all large itemsets inside each partition. The following steps represent the proposed distributed association rule algorithm:

Steps 1: For each partition (site) of distributed data warehouse find all unique itemsets with their frequency of occurrence and store them together in a table named“Local_House”.

Step2: For each partition (site) of distributed data warehouse find all local-supersets, store it in a table named local-super-set. These local-supersets must apply the following condition:

$Super\text{-}set(S(X)) = \{\ itemset(X) \in S \mid$ for all $itemsets(Y) \in S$, we have $itemset(X) \not\subset itemsets(Y)\ \}$.

Step3: Merge all unique itemsets found from step1, store them in a table named Global-House in the warehouse mining server. Then if any of these unique itemsets appear more than one time you have to select distinct values and compute the sum of their frequent of occurrence.

Step 4: Merge all of local-supersets found from step2, store them in a table named "Global-supersets" in the warehouse mining server. Then if any of these local-supersets appear more than one times you have to select distinct values of them.

Step 5: Find global supersets from table "Global-supersets" by applying the condition of step2, and this will eliminate any subsets existing in the above table and store only the global supersets.

Step 6: Perform first pruning operation by removing all of the one-itemsets that have support < global_min_sup from global supersets containing them; where the Support values of these One-Item-Set are computed from Global-House table generated in step3.

Step 7: Re-find global supersets after first pruning operation from table "Global-supersets" by applying condition of step2; by this step you will get the final global supersets.

Step 8: Generate all subsets for each of final global-supersets found in step seven, store distinct values of these subsets in a table named "Global-Itemsets".

Step 9: Compute the support value for the generated subsets in step8.

Step 10: Perform second pruning operation by deleting all itemsets that contain any of these subsets that have support < global_min_sup; where the support values for these subsets are computed from Global_house table. Store the remaining itemsets with their support in the "Global-Itemsets" table (the process of finding subset's support does not need any remote scan from any site).

Step 11: Generate the association rules from table "Global-Itemsets" and store the strong rules (that have confidence >= min_conf) in a table named "distributed-association-rules".

Figure (2-1) shows the flowchart of the proposed method.

3- Implementation and result

This section test the effectiveness of our new method for finding association rules mining in the distributed databases and compare result with another distributed association rule algorithm which is called count distribution (a distributed version of a apriori algorithm). A series of experimental and comparisons have been conducted to show the efficiency and outperformance of our new method (see tables (3-1) and (3-2). The experiments were run for several minimum support values for each dataset. In particular, except when showing the effects of minimum support and number of partition change, we reported results corresponding to two, four and six partitions and to the different minimum support thresholds used, usually characterized by a difference of about one order of magnitude in execution time as shown in figures (3-1), (3-2). In our experimental we take an example based on medical diagnosis information system for distributed health care systems consisting of databases that contain patient data and mined knowledge from health care institutions. The file resembles transaction data: the first column consists of the patient's visit identification number, the second column contains a visit date, the third column contains a medical test identifier for the symptoms, the fourth column contains the patient gender, and the fifth column contains the age of the patients as shown in Table (3-3). It is obvious that the number of tables used by our new method is fixed and only one table is required, while in the CD algorithm the number of tables' increases linearly with the number of global itemsets-subsets increase, the chart (3-3) shows the total number of storage tables required by our new method and CD algorithms for different global itemset-subsets.

4- Conclusions

The proposed method has the following characteristics:

Ø It reduces *Communication cost.* Since the transfer of huge data volumes over network might takes extremely much time and also requires an unbearable financial cost this is avoided by the method. Also the algorithm utilizes the network resources by minimizing message transfer among sites, so it needs only O(n) messages transfer for transferring local supersets and unique itemsets with their frequent of occurrence to the Warehouse Mining Server (WMS), where n is the number of distributed data warehouse sites . The process of finding the support of all itemsets in the data mining server does not need any remote scan (zero remote scan) or message transfer to any of distributed sites, because the support is computed locally inside WMS.

Ø It achieves high performance. Since the computational cost of mining a central data warehouse is much bigger than the sum of the cost of analyzing smaller parts of the data warehouses, which could also be done in parallel. The efficiency of the algorithm increases with when the number of distributed sites increase but this leads to increases in the size of main memory used by it. Also the pruning process done by using the algorithm which adds extra improvement to the performance of the algorithm by using only two global pruning processes that enhance the search space to find the interesting association rules; especially when the number of itemsets becomes too huge by eliminating all of the uninteresting itemsets that lead to weak rules from search space of the algorithm.

Ø It is scalable and flexible. It achieves high scalability compared to classical Apriori-based implementations. However it can extend the distributed data warehouse by adding unlimited new sites to it easily; also the algorithm can be executed on the whole distributed warehouse, set of sites or even on a single site. Another important feature of the proposed method is the ability to easily maintain the supersets for the database partitions. When a database or data warehouse is updated, the supersets for the updated database partitions should be updated too. When new partitions are appended to a database or data warehouse, then the supersets for the new partitions must be computed, however, none of the previously supersets need to be updated. Besides, computing the positive borders for a partition can be done fast, because the whole partition is likely to fit in main memory.

Ø It utilizes the storage resources; since it uses smart subset generator function which needs only one table for storing all K-itemsets subsets (i.e. one-itemset subsets, two-itemset subsets,….n-itemsets subsets), while the other traditional technique needs a separate table for each K-itemset subset (i.e. one-itemset subsets table, two-itemset subsets table,… n-itemset subsets table), in one table; also this smart function eliminates generating of unrelated subsets rather than other DARM algorithms that generate unrelated subsets.

Ø It does not exhaust the processor by complex processes. So the whole process requires first: finding global supersets, second: generating the subsets with their support, finally finding the strong association rules from them.

References

[1] R. Agrawal, T. Imielinski and A. Swami "Database Mining: A Performance Perspective". IEEE Trans. Knowledge and Data Engineering, England, 1997.

[2] Yijun, Xuemin Lin, and C. Tsang, "An Efficient Distributed Algorithm for Computing Association Rules". Springer-Verlag Berlin Heidelberg 2000.

[3] Ran Wolff Assaf Schuster ," A High-Performance Distributed Algorithm for Mining Association Rules". IEEE Conference on Data Mining (ICDM), Florida, 2003.

[4] David Hand, Heikki Mannila and Padhraic Smyth, "Principles of Data Mining". The MIT Press,Cambridge, London England,2001.

[5] Pieter Adriaans , Dolf Zantinge " Data Mining" Addision_Wesley, 1998.

[6] Mohammed Javeed Zaki, Mitsunori Ogihara, Srinivasan Parthasarathy, and Wei Li, "Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors". Technical Report TR 618, University of Rochester, Computer Science Department,1999.

[7] Sergy Brin, Rajeev Motwani, Jeffery D. Ullman and Sergy Tsur."Dynamic Itemset Counting And Implication Rules For

Market Basket Data". In proceeding of data (SGMOD97) Tucson, Arizona USA May 1997.

[8] R. Agrawal and R. Srikant. "Fast algorithms For Mining Association Rules". In Proceedings of the 20th VLDB Conference, pages 487-499, 1994.

[9] Jiawei Han, Micheline Kanmber, "Data Mining Concepts and Techniques". Academic press, USA, 2001.

[10] Nhien An Le Khac, Lamine M. Aouad and M-Tahar Kechadi," Distributed Knowledge Map for Mining Data on Grid Platforms". IJCSNS International Journal of Computer Science and Network 98 Security, VOL.7 No.10, October 2007.

[11] Assaf Schuster and Ran Wolff. "Communication-Efficient Distributed Mining of Association Rules". In SIGMOD '01: Proceedings of the 2001 ACM Sigmod International Conference on Management of Data, pages 473–484, New York, NY, USA, ACM Press.May 2001.

[12] Osmar R. Zaiane, Mohammad El-Hajj, and Paul Lu. "Fast Parallel Association Rule Mining without Candidacy Generation". In ICDM, pages 665–668, 2001.

[13] M.E. Otey, C. Wang, S. Parthasarathy, A. Veloso, and Jr. W. Meira. "Mining Frequent Itemsets in Distributed and Dynamic Databases". In ICDM 2003: Third IEEE International Conference on Data Mining, pages 617– 620, Nov. 2003.

[14] Assaf Schuster, Ran Wolff, and Dan Trock. "A high-Performance Distributed Algorithm for Mining Association Rules". In ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining, page 291, Washington, DC, USA, 2003.

[15] Emad Kadum Jabbar, "New Algorithms for Discovering Association Rules". PHD. thesis, Department of Computer Sciences of the University of Technology, 2005.

[16] Claudio Silvestri, Salvatore Orlando, "Distributed Approximate Mining of Frequent Patterns". ACM Symposium on Applied Computing, Italy, 2005.

[17] Rawia Tahrir Salih Kadoori," Extracting Association Rules From Distributed Association Rules".MSc. Thesis Computer Science, University of Technology, 2002.

[18] Lamine M. Aouad, Nhien-An Le-Khac and Tahar M. Kechadi, "Distributed Frequent Itemsets Mining in Heterogeneous Platforms". Journal of engineering, computing and architecture, Volume 1, Issue 2, School of Computer Science and Informatics University College Dublin, 2007.

[19] R. Agrawal and J. C. Shafer. "Parallel Mining of Association Rules". IEEE Transactions On Knowledge And Data Engineering, 8:962–969, 1996.

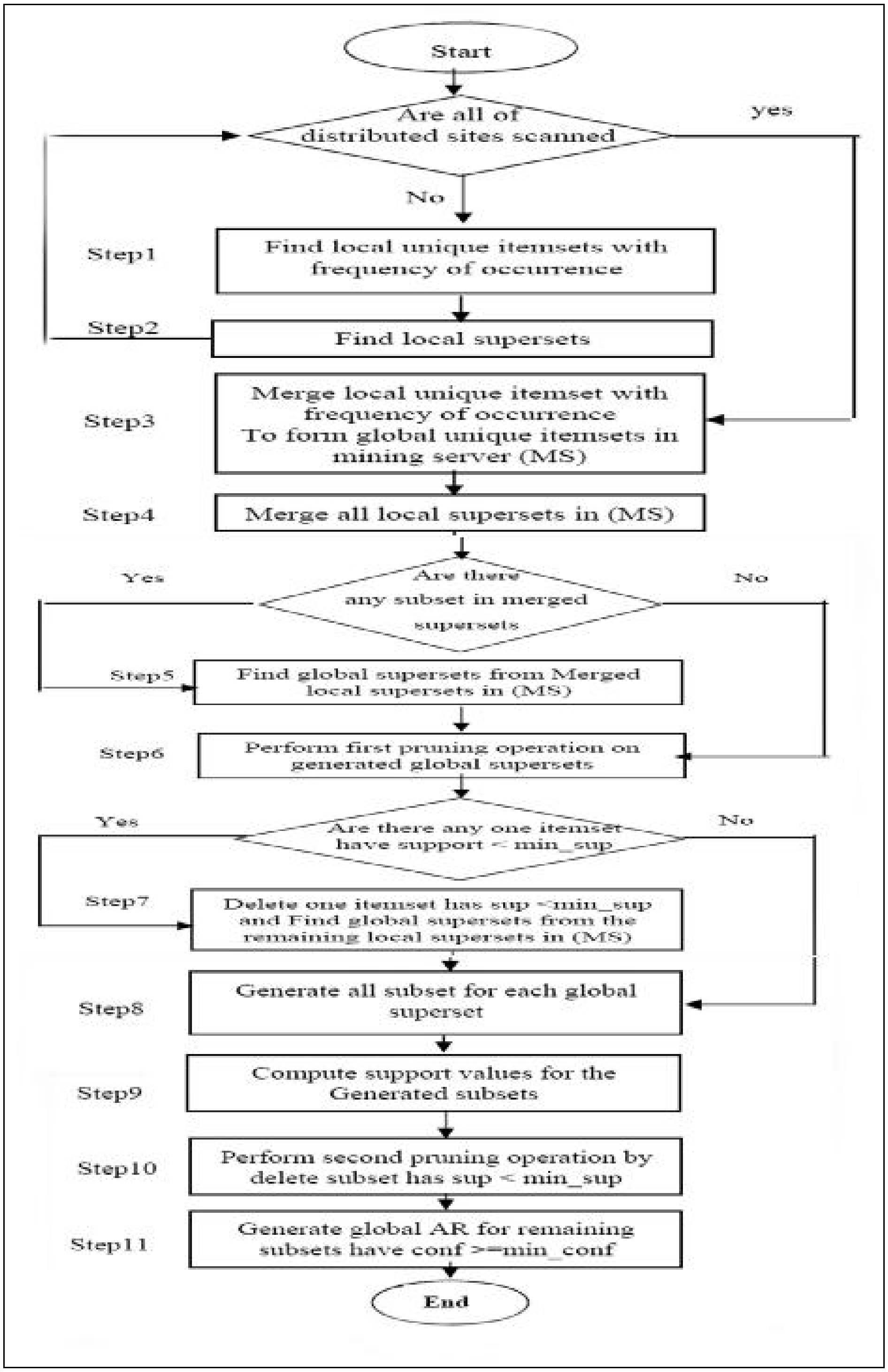

Start
Are all of distributed sites scanned
yes
No
Step1
Find local unique itemsets with frequency of occurrence
Step2
Find local supersets
Step3
Merge local unique itemset with frequency of occurrence To form global unique itemsets in mining server (MS)
Step4
Merge all local supersets in (MS)
Yes
Are there any subset in merged supersets
No
Step5
Find global supersets from Merged local supersets in (MS)
Step6
Perform first pruning operation on generated global supersets
Yes
Are there any one itemset have support < min_sup
No
Step7
Delete one itemset has sup <min_sup and Find global supersets from the remaining local supersets in (MS)
Step8
Generate all subset for each global superset
Step9
Compute support values for the Generated subsets
Step10
Perform second pruning operation by delete subset has sup < min_sup
Step11
Generate global AR for remaining subsets have conf >=min_conf
End

Figure (2-1) flowchart of the proposed approach.

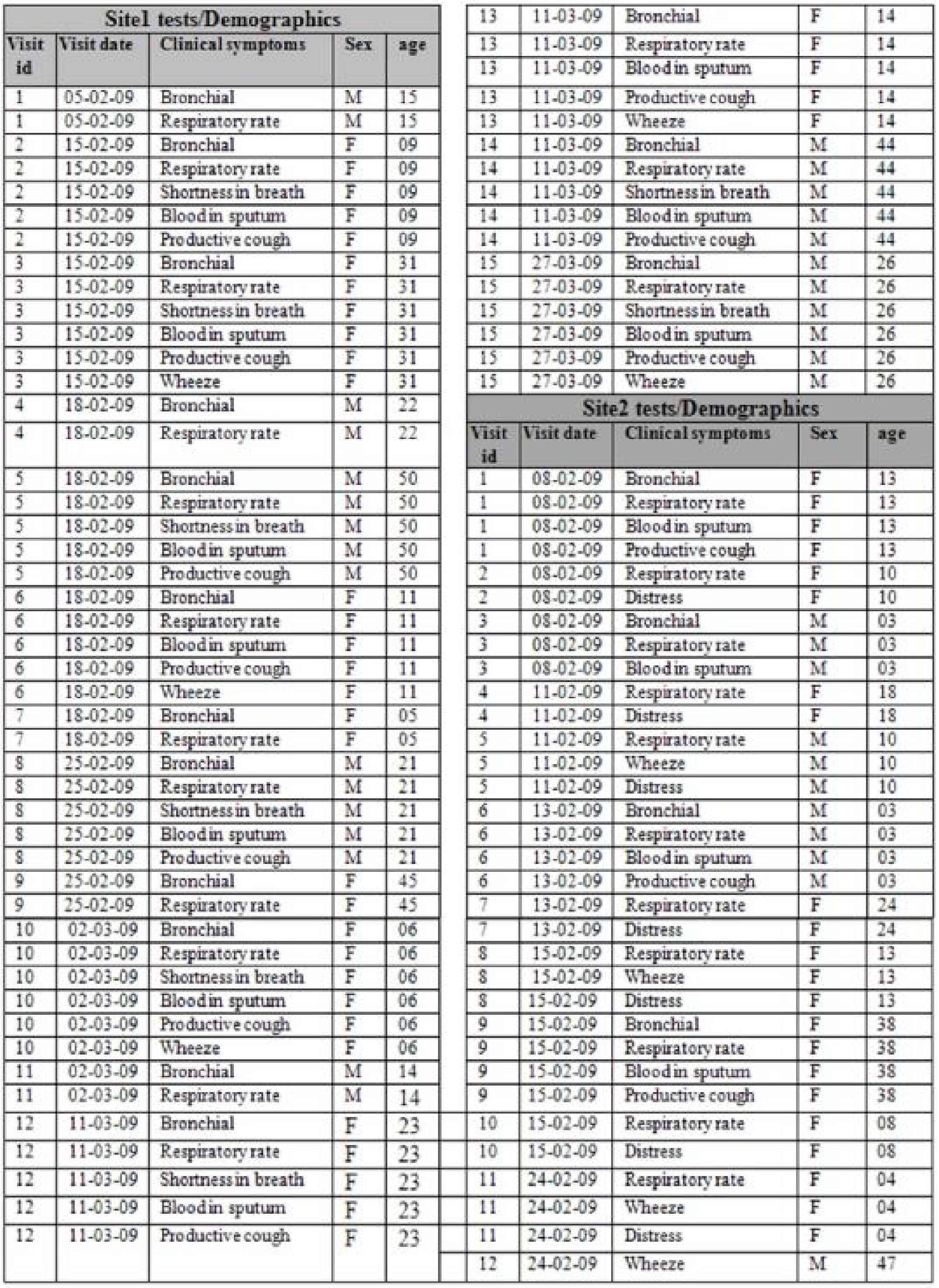
Table (3-1) execution time of the proposed method on two sites

No. Of Items	No. Of Item Set	Global Min. Sup. %	Exec.time For Distrib. 100,000 Records hh:mm:ss	Exec.time For Distrib. 250,000 Records hh:mm:ss	Exec.time For Distrib. 500,000 Records hh:mm:ss	Exec.time For Distrib. 750,000 Records hh:mm:ss	Exec.time For Distrib. 1000,000 Records hh:mm:ss
5	31	20	00:01:00	00:01:11	00:01:26	00:02:10	00:02:58
		30	00:00:51	00:01:00	00:01:08	00:01:46	00:02:37
		40	00:00:37	00:00:39	00:00:53	00:01:37	00:02:04
		50	00:00:29	00:00:31	00:00:36	00:01:12	00:01:41
		60	00:00:16	00:00:21	00:00:27	00:01:02	00:01:19
7	127	20	00:02:47	00:03:13	00:03:47	00:04:11	00:04:33
		30	00:02:05	00:02:36	00:03:14	00:03:34	00:04:00
		40	00:01:13	00:01:49	00:02:30	00:03:15	00:03:38
		50	00:00:56	00:01:23	00:02:09	00:02:26	00:02:53
		60	00:00:33	00:01:00	00:01:36	00:02:02	00:02:28
9	511	20	00:03:45	00:04:07	00:04:38	00:05:15	00:05:25
		30	00:03:07	00:03:15	00:03:42	00:04:18	00:04:55
		40	00:02:36	00:02:44	00:03:03	00:03:21	00:04:06
		50	00:01:43	00:01:52	00:02:14	00:02:40	00:03:00
		60	00:01:05	00:01:23	00:01:55	00:02:07	00:02:39
12	4095	20	00:06:10	00:08:21	00:09:57	00:10:46	00:12:14
		30	00:05:36	00:07:45	00:08:22	00:09:03	00:11:05
		40	00:04:57	00:06:14	00:06:37	00:07:20	00:08:09
		50	00:03:51	00:05:09	00:05:25	00:06:41	00:07:00
		60	00:03:22	00:04:37	00:05:03	00:05:47	00:06:36
15	32767	20	00:15:00	00:17:24	00:22:04	00:25:09	00:27:00
		30	00:13:35	00:16:00	00:19:31	00:21:00	00:23:39
		40	00:08:11	00:09:22	00:12:12	00:15:30	00:17:43
		50	00:06:49	00:07:18	00:08:24	00:12:13	00:14:08
		60	00:05:37	00:06:03	00:07:11	00:10:05	00:13:11

Table (3-2) execution time of Count Distribution Algorithm on two sites

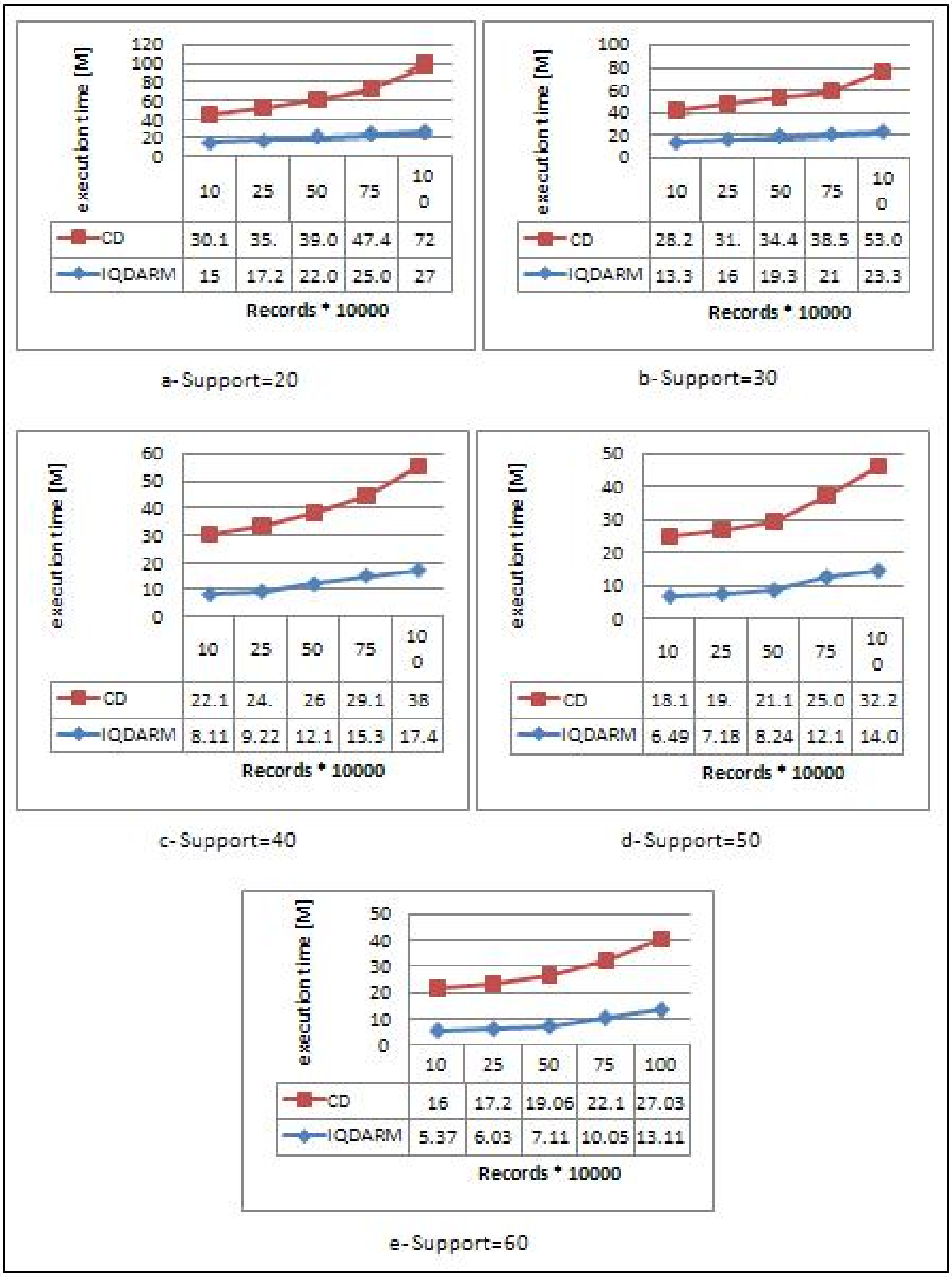
No. Of Items	No. Of Item Set	Global Min. Sup. %	Exec.time For Distrib. 100,000 Records hh:mm:ss	Exec.time For Distrib. 250,000 Records hh:mm:ss	Exec.time For Distrib. 500,000 Records hh:mm:ss	Exec.time For Distrib. 750,000 Records hh:mm:ss	Exec.time For Distrib. 1000,000 Records hh:mm:ss
5	31	20	00:05:14	00:07:28	00:08:05	00:11:04	00:13:27
		30	00:03:58	00:05:29	00:06:29	00:09:13	00:10:24
		40	00:02:41	00:04:41	00:05:17	00:06:27	00:08:12
		50	00:02:13	00:02:42	00:04:00	00:04:46	00:06:31
		60	00:01:22	00:02:10	00:02:31	00:04:04	00:05:10
7	127	20	00:08:38	00:12:16	00:16:20	00:22:42	00:24:12
		30	00:06:14	00:09:00	00:12:31	00:18:31	00:20:05
		40	00:04:38	00:05:11	00:07:10	00:10:03	00:13:22
		50	00:03:45	00:04:16	00:05:04	00:07:11	00:09:51
		60	00:02:18	00:03:22	00:04:24	00:05:50	00:07:13
9	511	20	00:18:26	00:23:21	00:28:26	00:31:00	00:34:14
		30	00:16:14	00:19:50	00:21:17	00:26:05	00:29:10
		40	00:11:10	00:13:41	00:15:13	00:18:20	00:20:12
		50	00:08:23	00:10:23	00:12:10	00:14:44	00:16:11
		60	00:06:42	00:08:00	00:10:05	00:12:05	00:14:00
12	4095	20	00:22:16	00:26:40	00:30:07	00:35:23	00:37:00
		30	00:19:24	00:23:00	00:27:10	00:31:19	00:36:05
		40	00:15:06	00:17:48	00:20:15	00:23:30	00:25:43
		50	00:13:00	00:16:04	00:18:00	00:19:52	00:20:16
		60	00:11:03	00:13:50	00:15:17	00:17:00	00:17:24
15	32767	20	00:30:19	00:35:00	00:39:04	00:47:49	01:12:00
		30	00:28:26	00:31:28	00:34:45	00:38:55	00:53:05
		40	00:22:17	00:24:09	00:26:00	00:29:16	00:38:00
		50	00:18:19	00:19:25	00:21:12	00:25:05	00:32:21
		60	00:16:00	00:17:21	00:19:06	00:22:10	00:27:03

Table (3-3) sample of distributed medical data records used by the proposed method and CD Algorithm.



Site1 tests/Demographics				
Visit id	Visit date	Clinical symptoms	Sex	age
1	05-02-09	Bronchial	M	15
1	05-02-09	Respiratory rate	M	15
2	15-02-09	Bronchial	F	09
2	15-02-09	Respiratory rate	F	09
2	15-02-09	Shortness in breath	F	09
2	15-02-09	Blood in sputum	F	09
2	15-02-09	Productive cough	F	09
3	15-02-09	Bronchial	F	31
3	15-02-09	Respiratory rate	F	31
3	15-02-09	Shortness in breath	F	31
3	15-02-09	Blood in sputum	F	31
3	15-02-09	Productive cough	F	31
3	15-02-09	Wheeze	F	31
4	18-02-09	Bronchial	M	22
4	18-02-09	Respiratory rate	M	22
5	18-02-09	Bronchial	M	50
5	18-02-09	Respiratory rate	M	50
5	18-02-09	Shortness in breath	M	50
5	18-02-09	Blood in sputum	M	50
5	18-02-09	Productive cough	M	50
6	18-02-09	Bronchial	F	11
6	18-02-09	Respiratory rate	F	11
6	18-02-09	Blood in sputum	F	11
6	18-02-09	Productive cough	F	11
6	18-02-09	Wheeze	F	11
7	18-02-09	Bronchial	F	05
7	18-02-09	Respiratory rate	F	05
8	25-02-09	Bronchial	M	21
8	25-02-09	Respiratory rate	M	21
8	25-02-09	Shortness in breath	M	21
8	25-02-09	Blood in sputum	M	21
8	25-02-09	Productive cough	M	21
9	25-02-09	Bronchial	F	45
9	25-02-09	Respiratory rate	F	45
10	02-03-09	Bronchial	F	06
10	02-03-09	Respiratory rate	F	06
10	02-03-09	Shortness in breath	F	06
10	02-03-09	Blood in sputum	F	06
10	02-03-09	Productive cough	F	06
10	02-03-09	Wheeze	F	06
11	02-03-09	Bronchial	M	14
11	02-03-09	Respiratory rate	M	14
12	11-03-09	Bronchial	F	23
12	11-03-09	Respiratory rate	F	23
12	11-03-09	Shortness in breath	F	23
12	11-03-09	Blood in sputum	F	23
12	11-03-09	Productive cough	F	23
13	11-03-09	Bronchial	F	14
13	11-03-09	Respiratory rate	F	14
13	11-03-09	Blood in sputum	F	14
13	11-03-09	Productive cough	F	14
13	11-03-09	Wheeze	F	14
14	11-03-09	Bronchial	M	44
14	11-03-09	Respiratory rate	M	44
14	11-03-09	Shortness in breath	M	44
14	11-03-09	Blood in sputum	M	44
14	11-03-09	Productive cough	M	44
15	27-03-09	Bronchial	M	26
15	27-03-09	Respiratory rate	M	26
15	27-03-09	Shortness in breath	M	26
15	27-03-09	Blood in sputum	M	26
15	27-03-09	Productive cough	M	26
15	27-03-09	Wheeze	M	26

Site2 tests/Demographics				
Visit id	Visit date	Clinical symptoms	Sex	age
1	08-02-09	Bronchial	F	13
1	08-02-09	Respiratory rate	F	13
1	08-02-09	Blood in sputum	F	13
1	08-02-09	Productive cough	F	13
2	08-02-09	Respiratory rate	F	10
2	08-02-09	Distress	F	10
3	08-02-09	Bronchial	M	03
3	08-02-09	Respiratory rate	M	03
3	08-02-09	Blood in sputum	M	03
4	11-02-09	Respiratory rate	F	18
4	11-02-09	Distress	F	18
5	11-02-09	Respiratory rate	M	10
5	11-02-09	Wheeze	M	10
5	11-02-09	Distress	M	10
6	13-02-09	Bronchial	M	03
6	13-02-09	Respiratory rate	M	03
6	13-02-09	Blood in sputum	M	03
6	13-02-09	Productive cough	M	03
7	13-02-09	Respiratory rate	F	24
7	13-02-09	Distress	F	24
8	15-02-09	Respiratory rate	F	13
8	15-02-09	Wheeze	F	13
8	15-02-09	Distress	F	13
9	15-02-09	Bronchial	F	38
9	15-02-09	Respiratory rate	F	38
9	15-02-09	Blood in sputum	F	38
9	15-02-09	Productive cough	F	38
10	15-02-09	Respiratory rate	F	08
10	15-02-09	Distress	F	08
11	24-02-09	Respiratory rate	F	04
11	24-02-09	Wheeze	F	04
11	24-02-09	Distress	F	04
12	24-02-09	Wheeze	M	47



a-Support=20

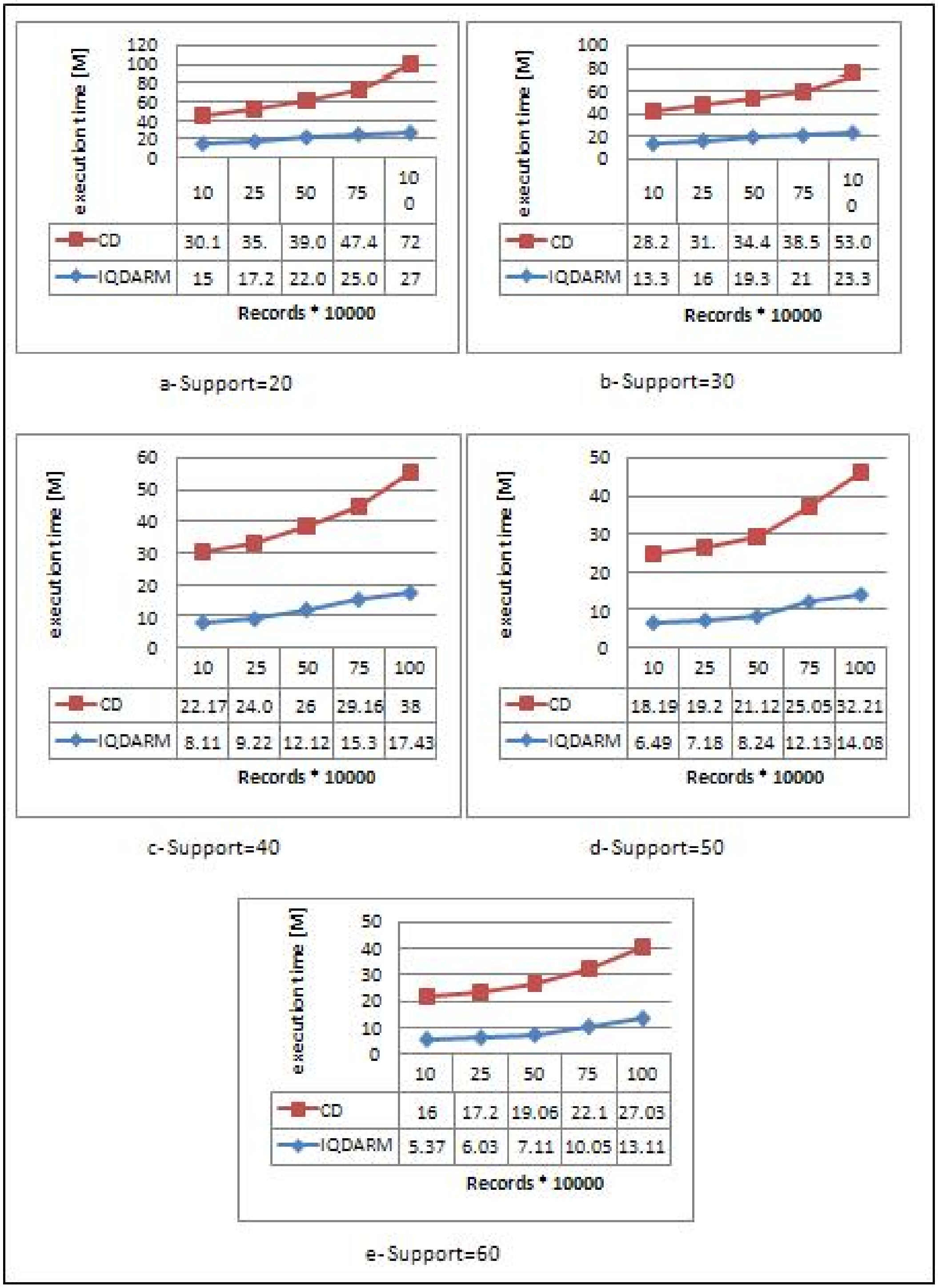
b-Support=30

c-Support=40

d-Support=50

e-Support=60

Figure (3-1) Chart of proposed method, Count Distribution Algorithms for 15 items on two sites with 5 different supports



a-Support=20

b-Support=30

c-Support=40

d-Support=50

e-Support=60

Figure (3-2) Chart of proposed method, Count Distribution Algorithms for 15 items on four sites with 5 different supports.

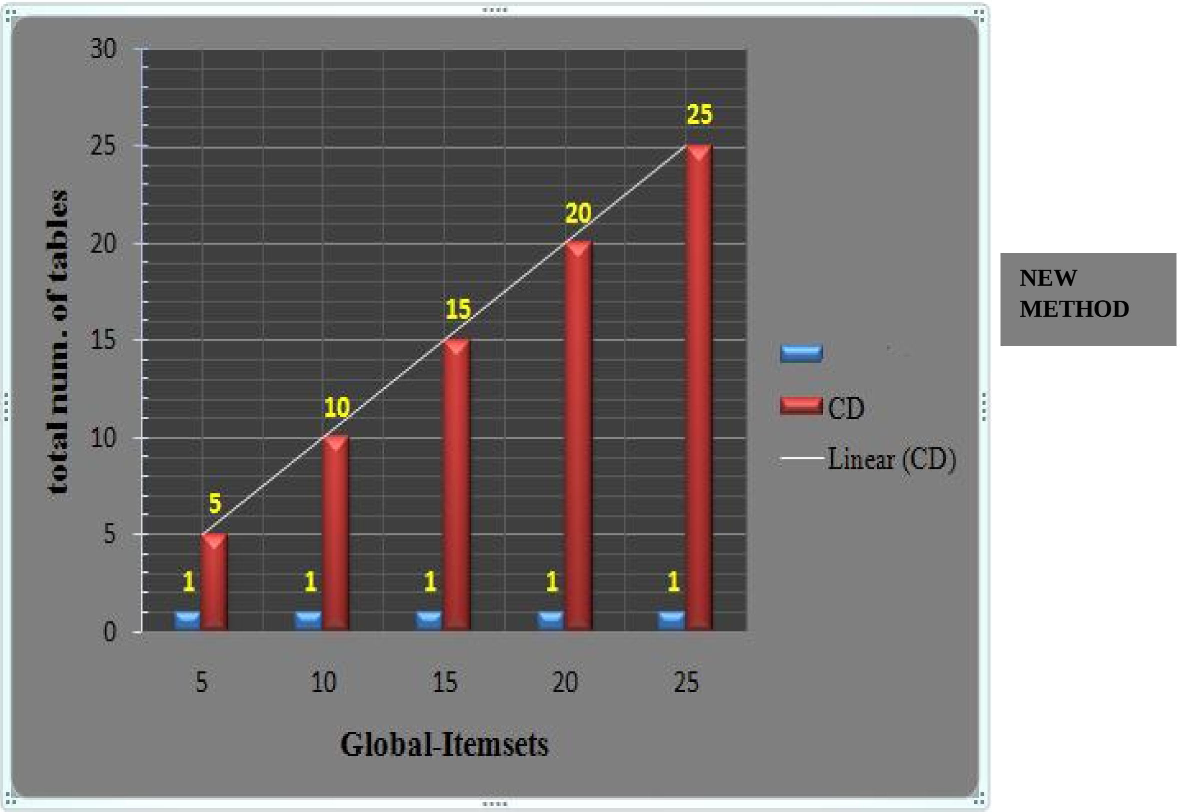


Figure (3-3) total number of tables required by the proposed method and Count Distribution Algorithm.