# Introducing Parallelism in Privacy Preserving Data Mining Algorithms

[1,] Mrs.P.Cynthia Selvi, [2,] Dr.A.R.Mohamed Shanavas
[1]*Associate Professor, Dept. of Comp. Sci.*
*K.N.Govt.Arts College for Women, Thanjavur, Tamilnadu, India*
[2]*Associate Professor, Dept. of Comp. Sci.*
*Jamal Mohamed College Trichirapalli, Tamilnadu, India*

**ABSTRACT**

*The development of parallel and distributed data mining algorithms in various functionalities have been motivated by the huge size and wide distribution of the databases and also by the computational complexity of the data mining methods. Such algorithms make partitions of the huge database that is being used into segments that are processed in parallel. The results obtained from the processed segments of database are then merged; This reduces the computational complexity and improves the speed. This article aims at introducing parallelism in data sanitization technique in order to improve the performance and throughput.*

***KEYWORDS:*** *Cover, Parallelism, Privacy Preserving Data Mining, Restrictive Patterns, Sanitization, Sensitive Transactions. Transaction-based Maxcover Algorithm.*

## I. INTRODUCTION

Data mining is an emerging technology that enable the discovery of interesting patterns from large collections of data. As the amount of data being collected continues to increase very rapidly, scalable algorithms for data mining becomes essential; Moreover, it is a challenging task for data mining approaches to handle large amount of data effectively and efficiently. Scaling up the data mining algorithms to be run in high-performance parallel and distributed computing environments offers an alternative solution for effective data mining. Parallelization is a process that consist of breaking up a large single process into multiple smaller tasks which can run in parallel and the results of those tasks are combined to obtain an overall improvement in performance. With a lot of information accessible in electronic forms and available on the web, and with increasingly powerful data mining tools being developed and put into use, there are increasing concerns that data mining pose a threat to privacy and data security. This motivated the area of Privacy Preserving Data Mining(PPDM) and its main objective is to develop algorithms to transform the original data to protect the private data and knowledge without much utility loss. There are many approaches for preserving privacy in data mining; to name a few are perturbation, encryption, swapping, distortion, blocking, sanitization. The task of transforming the source database into a new database that hides some sensitive knowledge is called sanitization process[1].This article introduce the concept of parallelism in PPDM algorithms. Section-2 narrates the previous work on PPDM. Section-3 introduces the basic terminologies and the proposed algorithm is presented in section-4. Section-5 gives implementation details and the observed results.

## II. LITERATURE SURVEY

The idea behind data sanitization was introduced in [2], which considered the problem of modifying a given database so that the support of a given set of sensitive rules decreases below the minimum support value. The authors focused on the theoretical approach and showed that the optimal sanitization is an NP-hard problem. In [3], the authors investigated confidentiality issues of a broad category of association rules and proposed some algorithms to preserve privacy of such rules above a given privacy threshold.In the same direction, Saygin[4] introduced some algorithms to obscure a given set of sensitive rules by replacing known values with unknowns, while minimizing the side-effects on non-sensitive rules. Like the algorithms proposed in [3], these algorithms are CPU-sensitive and require various scans depending on the no. of association rules to be hidden. In [5,6], heuristic-based sanitization algorithms have been proposed. All these algorithms concentrated on data hiding principle to be implemented on the source database as a whole and parallelism is not dealt with. Hence this work makes an attempt to introduce parallelism in Transaction-based Maxcover Algorithm(TMA) proposed in [6] and improved performance is obtained from the observed results.

## III. BASIC CONCEPTS OF PPDM ALGORITHM

*Transactional Database :* A transactional database consists of a file where each record represents a transaction that typically includes a unique identity number (*trans_id*) and a list of items that make up the transaction. Let D be a source database which is a transactional database containing a set of transactions T, where each transaction *t* contain an itemset $X \in D$. Also, every $X \subseteq I$ has an associated set of transactions $T \subseteq D$, where $X \subseteq t$ and $t \in T$.

*Association Rule :* It is an expression of the form $X \Rightarrow Y$, where X and Y contain one or more itemsets(categorical values) without common elements ($X \cap Y = \phi$).

*Frequent Pattern* : An itemset or pattern that forms an association rule is said to be frequent if it satisfies a prespecified minimum support threshold(*min_sup*).

*Restrictive Patterns* : Let P be a set of significant patterns that can be mined from transactional source database D, and $R_H$ be a set of rules to be hidden according to some privacy policies. A set of all patterns $rp_i$ denoted by $R_P$ is said to be *restrictive*, if $R_P \subset P$ and if and only if $R_P$ would derive the set $R_H$. $\sim R_P$ is the set of *non-restrictive patterns* such that $\sim R_P \cup R_P = P$ [4].

*Sensitive Transactions* : A set of transactions is said to be *sensitive*, denoted by $S_T$, if every $t \in S_T$ contain atleast one restrictive pattern $rp_i$ . ie $S_T = \{ t \in T \mid \exists\ rp_i \in R_P, rp_i \subseteq t \}$.

*Cover* : The *Cover*[5] of an item $A_k$ can be defined as,

$C_{Ak} = \{ rp_i \mid A_k \in rp_i \subset R_P, 1 \leq i \leq |R_P| \}$

*i.e.,* set of all restrictive patterns which contain $A_k$. The item that is included in a maximum number of $rp_i$'s is the one with *maximal cover or maxCover;*

*i.e., maxCover* = *max*( $|C_{A1}|, |C_{A2}|, \ldots |C_{An}|$ ) such that $A_k \in rp_i \subset R_P$.

*Principle of Transaction-based Maxcover Algorithm(TMA)* [6]: Initially, identify the transaction-list of each $rp_i \subset R_P$. Starting with $rp_i$ having larger *supCount*, for every transaction *t* in *t-list*($rp_i$), find the *cover*($A_k$) within *t* such that $A_k \in rp_i \subset t$. Delete item $A_k$ with *maxCover* in *t*, and decrease the *supCount* of all $rp_i$'s which are included in *t.* Also mark this *t* as *victim transaction* in the *t-list* of the corresponding $rp_i$'s. Repeat this process until the *supCount* of all $rp_i$'s are reduced to 0.

## IV. ALGORITHM

The sanitization task is distributed among the Server (a server is an entity that has some resource that can be shared) and the Clients (a client is simply any other entity which wants to gain access to a particular server) and the task is implemented as two modules namely Server Module and Client Module.

*Procedure(Server module):*
*Input : Source transactional database(D)*
*Output : Sanitized database(D')*
*Start;*
  *Get Source Database(D);*
  *Get N; //N- no. of data segments;*
  *Horizontally partition the D into N segments;*
  *Initialize Lookup Tables;*
  *Allocate the segment to client; // one each*
  *Merge the sanitized segments received from N clients;*
  *Display Sanitized Database(D');*
*End*
*Procedure(Client module):*
*Start;*
  *Get data segment from Server;*
  *Access the Lookup Tables;*
  *Run TMA algorithm to sanitize the data segment;*
  *Return Sanitized segment to Server;*
*Stop.*

## V. IMPLEMENTATION

The proposed algorithm was tested on real databases *RETAIL & T10I4D100K*[7] with samples of transactions between 1000 and 10000. The restrictive patterns were chosen in a random manner with their support ranging between 0.6 and 5, confidence between 32.5 and 85.7 and length between 2 and 6. The test run was made on AMD Turion II N550 Dual core processor with 2.6 GHz speed and 2GB RAM operating on 32 bit OS; The implementation of the proposed algorithm was done with windows 7 - Netbeans 6.9.1 - SQL 2005. The coding part is done with JDK 1.7; because Java's clean and type-safe object-oriented programming model together with its support for parallel and distributed computing make it an attractive environment for writing reliable and parallel programs. The characteristics of Source and Sanitized databases are given in table-I & II respectively; the improved performance in terms of execution time is shown in Table-III & IV and the graphs.

**Table-I. Characteristics of source databases**

| Database Name | No. of Transactions | No. of Distinct Items | Min. Length | Max. Length | Min. no. of Sensitive Transactions | Max. no. of Sensitive Transactions | Database Size (MB) |
|---|---|---|---|---|---|---|---|
| Retail | 1K – 10K | 3176 – 8126 | 1 | 58 | 22 | 2706 | 90.2KB–928 KB |
| *T10I4D100K* | 1K – 10K | 795 - 862 | 1 | 26 | 7 | 78 | 94.8KB–786 KB |

**Table-II. Characteristics of sanitized databases**

| Database Name | No. of Transactions | No. of Distinct Items | Min. Length | Max. Length | Database Size(MB) |
|---|---|---|---|---|---|
| Retail | 1K – 10K | 3176 – 8126 | 1 | 58 | 90.0KB – 916KB |
| *T10I4D100K* | 1K – 10K | 795 - 862 | 1 | 26 | 94.0KB – 778KB |

### 5.1. Results

**Table-III. Results on the database *Retail***

| Size (No. of Transactions) | Execution Time (in seconds) | | | Merge Time (in seconds) | Turnaround Time(TT) in seconds TT=Max(Execution Time of Clients) + Merge Time |
|---|---|---|---|---|---|
| | Without Parallelism | With Parallelism | | | |
| | Server | Client-1 | Client-2 | | |
| 1000 | 50.90 | 0.297 | 0.218 | 0.009 | 0.306 |
| 5000 | 151.39 | 11.138 | 10.701 | 0.016 | 11.154 |
| 10000 | 110.03 | 23.08 | 20.65 | 0.02 | 23.10 |

**Table-IV. Results on the database *T10I4D100K***

| Size (No. of Transactions) | Execution Time(seconds) | | | Merge Time (seconds) | Turnaround Time(TT) in seconds TT=Max(Execution Time of Clients) + Merge Time |
|---|---|---|---|---|---|
| | Without Parallelism | With Parallelism | | | |
| | Server | Client-1 | Client-2 | | |
| 1000 | 8.80 | 0.015 | 0.018 | 0.015 | 0.033 |
| 5000 | 19.42 | 2.50 | 0.031 | 0.016 | 2.516 |
| 10000 | 29.00 | 1.747 | 0.973 | 0.031 | 1.778 |

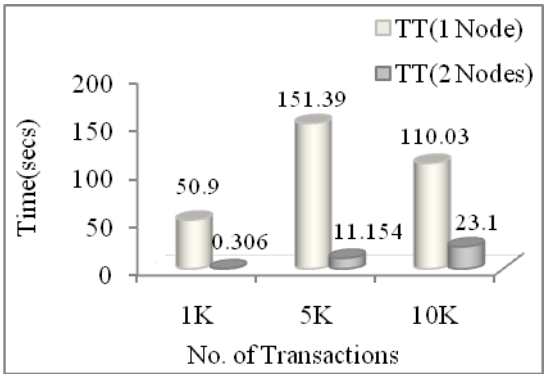


**Fig.1. Execution time(Retail Database)**

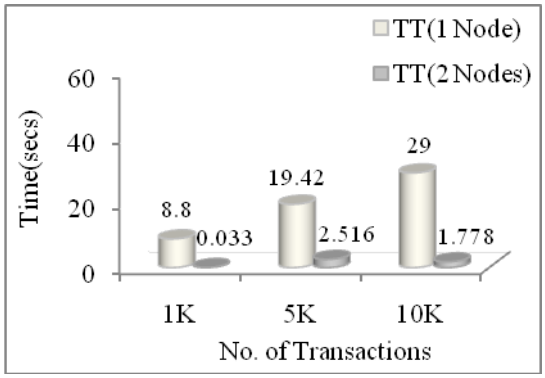


**Fig.2. Execution time(*T10I4D100K*)**

## VI. CONCLUSION

The main objective of this study is to improve the performance of the Privacy Preserving Data Mining Algorithm, TMA which was proposed and proved its performance in the earlier work[6]. In this article we implemented a parallel processing on TMA and we have observed very good improvement in the processing speed. Hence this empirical study showed that improved performance can be achieved but performance would degrade as the number of processors increased.

## REFERENCES

[1] A.Evfimievski, R.Srikant, R.Agarwal, Gehrke. Privacy Preserving mining of association rules. Proceedings of 8th ACM SIGKDD international conference on Knowledge discovery and data mining, Alberta, Canada. p.217-28. 2002.

[2] M.Atallah, E.Bertino, A.Elamagarmid, M.Ibrahim and V.Verykios. Disclosure Limitation of Sensitive Rules. In Proc. of IEEE knowledge and Data Engg.workshop. p.45-52. Chicogo, Illinois. Nov.1999.

[3] E.Dessani, V.S.Verykios, A.K.Elamagarmid and E.Bertino. Hiding association rules by using confidence and support. In 4th information hiding workshop. p.369-383. Pitsburg, PA. Apr 2001.

[4] Y.Saygin, V.S.Verikios and C.Clifton. Using unknowns to Prevent Discovery of Association Rules. SIGMOD Record. 30(4):45-54. Dec,2001.

[5] P.Cynthia Selvi, A.R.Mohamed Shanavas. An Improved Item-based Maxcover Algorithm to Protect Sensitive Patterns in Large Databases. IOSR-Journal on Computer Engineering. Volume 14, Issue 4 (Sep-Oct, 2013). PP 01-05. DOI. 10.9790/0661-1440105.

[6] P.Cynthia Selvi, A.R.Mohamed Shanavas. Towards Information Privacy using Transaction-based Maxcover Algorithm. Presented on International Conference on Data Mining and SoftComputing Techniques. SASTRA University, India. 25th & 26th Oct'2013.

[7] The Dataset used in this work for experimental analysis was generated using the generator from IBM Almaden Quest research group and is publicly available from http://fimi.ua.ac.be/data/