Introducing Parallelism in Privacy Preserving Data Mining Algorithms

Mrs. P. Cynthia Selvi, 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

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 consists 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 introduces the concept of 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.

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 ∈ D. Also, every X ⊆ T has an associated set of transactions T ⊆ D, where X ⊆ t and t ∈ T.

**Association Rule**: It is an expression of the form X ⇒ Y, where X and Y contain one or more items sets(categorical values) without common elements (X ∩ Y = φ).

**Frequent Patterns**: 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_P be a set of rules to be hidden according to some privacy policies. A set of all patterns P_P, denoted by R_P is said to be restrictive, if R_P ⊂ P and if and only if R_P would derive the set R_H. ~R_P is the set of non-restrictive patterns such that ~R_P ∪ R_P = P [4].

**Sensitive Transactions**: A set of transactions is said to be sensitive, denoted by S_T, if every t ∈ S_T contain at least one restrictive pattern R_P, i.e S_T = { t ∈ T | ∃ R_P ∈ R_P, R_P ⊆ t }.

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

\[ C_{A_i} = { | R_P | A_i \in R_P, 1 \leq i \leq |R_P| } \]

i.e., set of all restrictive patterns which contain A_i. The item that is included in a maximum number of R_P,’s is the one with maximal cover or maxCover;

i.e., \( \maxCover = \max\{ C_{A_1}, C_{A_2}, \ldots, C_{A_n} \} \) such that A_i ∈ R_P ⊆ R_P.

**Principle of Transaction-based Maxcover Algorithm(TMA)** [6]: Initially, identify the transaction-list of each R_P ⊆ R_P. Starting with R_P, having larger supCount, for every transaction t in t-list(R_P), find the cover(A_i) within t such that A_i ∈ R_P ⊆ t. Delete item A_i with maxCover in t, and decrease the supCount of all R_P,’s which are included in t. Also mark this t as victim transaction in the t-list of the corresponding R_P,’s. Repeat this process until the supCount of all R_P,’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>
<thead>
<tr>
<th>Database Name</th>
<th>No. of Transactions</th>
<th>No. of Distinct Items</th>
<th>Min. Length</th>
<th>Max. Length</th>
<th>Min. no. of Sensitive Transactions</th>
<th>Max. no. of Sensitive Transactions</th>
<th>Database Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retail</td>
<td>1K – 10K</td>
<td>3176 – 8126</td>
<td>1</td>
<td>58</td>
<td>22</td>
<td>2706</td>
<td>90.0KB – 928 KB</td>
</tr>
<tr>
<td>T10I4D100K</td>
<td>1K – 10K</td>
<td>795 - 862</td>
<td>1</td>
<td>26</td>
<td>7</td>
<td>78</td>
<td>94.8KB – 786 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database Name</th>
<th>No. of Transactions</th>
<th>No. of Distinct Items</th>
<th>Min. Length</th>
<th>Max. Length</th>
<th>Database Size(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retail</td>
<td>1K – 10K</td>
<td>3176 – 8126</td>
<td>1</td>
<td>58</td>
<td>90.0KB – 916KB</td>
</tr>
<tr>
<td>T10I4D100K</td>
<td>1K – 10K</td>
<td>795 - 862</td>
<td>1</td>
<td>26</td>
<td>94.0KB – 778KB</td>
</tr>
</tbody>
</table>

5.1. Results

<table>
<thead>
<tr>
<th>Size (No. of Transactions)</th>
<th>Execution Time (in seconds)</th>
<th>Merge Time (in seconds)</th>
<th>Turnaround Time(TT) in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without Parallelism</td>
<td>With Parallelism</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>Client-1</td>
<td>Client-2</td>
</tr>
<tr>
<td>1000</td>
<td>50.90</td>
<td>0.297</td>
<td>0.218</td>
</tr>
<tr>
<td>5000</td>
<td>151.39</td>
<td>11.138</td>
<td>10.701</td>
</tr>
<tr>
<td>10000</td>
<td>110.03</td>
<td>23.08</td>
<td>20.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size (No. of Transactions)</th>
<th>Execution Time (seconds)</th>
<th>Merge Time (seconds)</th>
<th>Turnaround Time(TT) in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without Parallelism</td>
<td>With Parallelism</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>Client-1</td>
<td>Client-2</td>
</tr>
<tr>
<td>1000</td>
<td>8.80</td>
<td>0.015</td>
<td>0.018</td>
</tr>
<tr>
<td>5000</td>
<td>19.42</td>
<td>2.50</td>
<td>0.031</td>
</tr>
<tr>
<td>10000</td>
<td>29.00</td>
<td>1.747</td>
<td>0.973</td>
</tr>
</tbody>
</table>
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

[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/