Chapter 4

Distributed Compact Pattern Tree

The present marketing strategies are distributed by nature. The examples, malls like Big Bazaar, Super Markets etc., The transactional data compiled at any of the distributed nodes will be voluminous. Thus, distributed computing strategies are essential and further optimal representation strategies are needed. This chapter focuses on representing this type of transactional data through a Compact Pattern Tree which is a variant of Transactional Tree representation. In the following, Section 4.1 provides an adequate Introduction. Section 4.2 describes about compact tree structure and CPTree, overview with the illustrations. Section 4.3 describes about the proposed Distributed Compact Pattern tree with the illustrations. Section 4.4 describes about Rule Extraction. Results and Discussion of CPTree are placed in Section 4.5.

4.1 Introduction

The nature of business is distributed across the continents geographically and the transactions at different locations (zone) themselves sufficient. There is a need to extract location-wise patterns and consolidating those patterns at global level for minimizing data transmission load. This implicitly results in privacy preserving. This demands distributed computing strategies.

Extracting rules for hierachal layers in order to obtain multi-level information for decision making in various situations. Many business organizations need collaborative knowledge to make decisions.
Several algorithms have been developed for privacy preserving in distributed environment \cite{AS00,LP00,P02,SK07}. In addition to that some algorithms have been developed for discovering unknown patterns through distributed \cite{GRSF97}, \cite{SP98} and parallel \cite{SK96,ZOP96} environments.

Privacy preserving data mining also plays key role in data mining. In Year 2000 by Agrawal \cite{AS00} proposed a reconstruction procedure to estimate the distribution of original data. Lindell and Pinkas \cite{LP00,P02} showed that privacy-preserving data mining problems could be solved by using the technique called Secure Multiparty Computation (SMC) \cite{GMW87}. Josenildo et al. \cite{SK07} stated in terms of privacy preservation, descriptive data owned by a local is no more disclosed to other sites. Here descriptive data may be data about individual transactions, buying patterns etc., Distributed sites may not want to disclose data to other sites in order to protect individual privacy.

In this chapter we focused to mine local and global rules from the abstraction of the distributed data by preserving the privacies of distributed data sources and addressed the constraints of communication cost. To validate the proposed method, in this chapter experiments have been conducted on different benchmark datasets such as Mushroom dataset and Pima Indian diabetes dataset with various support thresholds in multiple processors.

### 4.2 Compact Pattern Tree (CPTree)

Tanbeer et al. proposed CPTree \cite{TAJL08}. The CPTree, on the contrary, builds an FPTree like compact frequency-descending tree structure with only a single scan of the transaction database like CanTree which was discussed in earlier chapter. The transactions are partitioned and each partition of the transactions is called as block. CPTree achieves a frequency-descending structure by capturing the block of data from
the database dynamically in order to construct the tree. The construction process consists of two phases such as Insertion and Restructuring phase. The following is the procedure for constructing and restructuring the tree.

- A block of transactions are inserted into the tree with the same items appearance order of each transaction. The ilist (item appearance order in each transactions) is developed as and when the itemset is added to the tree. This procedure continued till all the transactions in the block are completed.
- In order to obtain frequency-descending tree structure, the tree has to be restructured for a block of transactions.
- Restructuring has to be done with the help of sorted ilist in frequency descending order. ilist has to be sorted and maintained as isort (ilist is sorted in frequency descending order) as shown in Figure 4.1(b).
- Each branch of the existing tree (block of transactions) has to be restructured according to the isort. Tanbeer et al. [TAJL08A] proposed a new tree restructuring technique called Branch Sorting Method. BSM sorts each path in the branch according to the isort by removing it from the tree, sorting it into a temporary array and inserting back into the tree.

As mentioned earlier, in Insertion Phase, transactions are scanned once and inserts them into the tree by maintaining ilist and updates the frequency count of respective items in the ilist. In Restructuring phase, ilist has to be rearranged according to frequency-descending order of items and restructures each branch according to isort. When a new block of transactions are inserted into the existing tree, the tree may lose the property of compactness. In order to achieve compactness the two phases should execute alternatively for each block of transactions. Tanbeer et al. [TAJL08A] stated that the above two phases executes alternatively to construct a frequency-descending tree structure to reduce overall restructuring cost and improves the possibility of prefix
sharing among all the patterns with one scan i.e., more frequently occurring items are more likely to be shared and thus they are arranged closer to the root of the CPTree. But Vishnu Priya et al. [VVT10] mentioned that the prefix tree construction time for mining frequent patterns is much higher. In order to overcome this problem Vishnu Priya et al. [VVT10] constructed the tree for the entire database at once instead of the blocks of database. Then restructuring of the tree takes place by rearranging each branch of the tree according to isort. In this chapter CPTree is constructed and restructured for the entire database instead of blocks of transactions. Illustrations of both block wise and entire database of transactions are provided in the following sections. The following are some of the properties of CPTree:

- Property 1: The total frequency count of any node in the CPTree is greater than or equal to the sum of the total frequency counts of its children.
- Property 2: CPTree maintains a complete set of all item projections of each transaction for the transaction database only once.
- Property 3: Projected databases can be formed by traversing the paths upwards starting from the last frequent item in the ilist. Since items are dynamically and consistently arranged in frequency-descending order, one can guarantee the inclusion of all frequent items using just upward traversals.
4.2.1 Illustrations of CPTree Construction (Block-Wise)

In this Section, Construction and Restructuring mechanism of a Compact Pattern Tree takes place by considering few transactions as a block as shown in Figure 4.1 which is developed by Tanbeer et al. [TAJL08A]. Table 4.1 shows the sample transactions for illustration, TID is the transaction ID and ‘items’ denote the items purchased during a particular transaction.

Table 4.1 Sample Database to illustrate CPTree

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>{c, a, e}</td>
</tr>
<tr>
<td>t₂</td>
<td>{b, d, f}</td>
</tr>
<tr>
<td>t₃</td>
<td>{a, b, d}</td>
</tr>
<tr>
<td>t₄</td>
<td>{b, d, e}</td>
</tr>
<tr>
<td>t₅</td>
<td>{f}</td>
</tr>
<tr>
<td>t₆</td>
<td>{g}</td>
</tr>
</tbody>
</table>
Figure 4.1 Compact Pattern Tree (Block Wise)

(a) inserting the first block of transaction i.e., $t_1$ to $t_3$
(b) restructuring the first block of transaction i.e., $t_1$ to $t_3$
(c) inserting the second block of transaction i.e., $t_4$ to $t_6$
(d) restructuring the second block of transaction i.e., $t_4$ to $t_6
4.2.2 Illustrations of CPTree Construction (Entire Database)

In this chapter the construction and restructure of tree is followed by considering all the transactions in the database. In order to obtain CPTree, one need to construct and restructure the tree at one time. In this thesis, Insertion of the entire transactions into the tree is called as Transactional Tree (TT) and restructuring the transaction tree is called as Restructure Tree (RT). The entire transactions in Table 4.1 are considered for the illustration given in Figure 4.1.

The approach starts with constructing tree by maintaining ilist (item appearance order list (unsorted list)). Scan the entire transactions once and insert into the tree by maintaining ilist as shown in Figure 4.2 (Transactional Tree). Further, ilist is sorted in descending order and maintained as isort (item sorted list). Based on isort the restructuring of the tree takes place as shown in Figure 4.3 (Restructured Tree).

![Figure 4.2 After inserting TID t₁ to t₆ (Transactional Tree)](image1)

![Figure 4.3 After restructuring Figure 4.2 according to isort (Restructured Tree)](image2)
4.3. Distributed Compact Pattern Tree

In order to extract local and global rules it is required to have Local Compact Pattern Trees (LCPT). As discussed in the Section 4.1 now-a-days the nature of business is distributed across the locations. Hence there is need to extract local patterns and consolidate them to achieve global patterns. In this scenario, one need to adapt distributed computing paradigm. For example considering Table 4.2 as the sample database of three locations (need not to be the same number of transactions). The database size may vary for each location. Distributed Compact Pattern Tree is nothing but constructing CPTree in distributed locations and each location CPTree is termed as LCPT. Three LCPTs i.e., LCPT$_1$, LCPT$_2$, LCPT$_3$ are shown in Figures 4.5, 4.7 and 4.9 respectively.

**Table 4.2 Sample Database of various distributed locations**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>TID</th>
<th>Items</th>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t$_1$</td>
<td>{a,b,d,g,e,c}</td>
<td>t$_6$</td>
<td>{a,b,d}</td>
<td>t$_{11}$</td>
<td>{c,b,a,e}</td>
</tr>
<tr>
<td>t$_2$</td>
<td>{c,a,e}</td>
<td>t$_7$</td>
<td>{d,a,b}</td>
<td>t$_{12}$</td>
<td>{g}</td>
</tr>
<tr>
<td>t$_3$</td>
<td>{d,f,b,a,e}</td>
<td>t$_8$</td>
<td>{b,d,e}</td>
<td>t$_{13}$</td>
<td>{a,b,c}</td>
</tr>
<tr>
<td>t$_4$</td>
<td>{b,d,f}</td>
<td>t$_9$</td>
<td>{a,c,b}</td>
<td>t$_{14}$</td>
<td>{f,e}</td>
</tr>
<tr>
<td>t$_5$</td>
<td>{a}</td>
<td>t$_{10}$</td>
<td>{f}</td>
<td>t$_{15}$</td>
<td>{a,b,c}</td>
</tr>
</tbody>
</table>

The approach starts by constructing and restructuring of the tree by taking the transaction TIDs from 1$^{st}$ location. The TT$_1$ is constructed by maintaining TL$_1$(1$^{st}$ location local database item appearance order list) as shown in Fig 4.4. TL$_1$ sorted in frequency descending order and maintained as TS$_1$ (TL$_1$ sorted in frequency descending order). Based on TS$_1$ the restructuring of the tree takes place and shown in Fig 4.5. The same procedure is followed for constructing and restructuring of TIDs from second location and TIDs from third location. The figures corresponding to the second location
Figure 4.4 Transactional Tree ($TT_1$) of first location

Figure 4.5 Restructured Tree ($RT_1$) of first location - LCPT
Figure 4.6 Transactional Tree (TT_2) of second location

Figure 4.7 Restructured Tree (RT_2) of second location - LCPT_2

Figure 4.8 Transactional Tree (TT_3) of third location

Figure 4.9 Restructured Tree (RT_3) of third location - LCPT_3
are shown in Figure 4.6 and 4.7. Similarly, the figures corresponding to the third location are shown in Figure 4.8 and 4.9. Transactional Tree is constructed by applying the algorithm [VVT10] with few modifications described below.

**Algorithm 4.1 Transactional Tree** [VVT10]

**Input:** LDB: Local Database

N: Total Number of transactions

**Output:** TT: Transactional Tree

TS: Item sorted list

**Method:**

1. Read LDB, N and set j=1.
2. Create the root ‘r’ of the tree T and label it as “null”
3. while (j<=N){
4. Scan the transaction t_j once
5. Let the transaction be [p/P], where p is the first element and P is the remaining elements in the transaction, then call construct_tree ([p/P],r)
6. j=j+1}
7. Find occurrences of each item in the tree and arrange each item based on occurrence. The arranged items are sorted in TS.

Function **construct_tree**

Input: [p/P]: p current element of a transaction, P: Remaining portion of transaction

T: Current node of the Tree.

Method:

1. If T has a child C such that C.item_name=P.item_name
(2) then increment count by 1.

(3) If length of $P > 0$ and $P$ is $[q/Q]$

(4) then

   a. \textit{construct\_tree} ($[q/Q], C$)

(5) else

   a. Attach $[p/P]$ as the child branch of $T$.

---

**Algorithm 4.2 Restructure Tree**

**Input:** TT: Transactional Tree

TS: ISort

**Output:** LCPT: Local compact pattern tree

**Method:**

(1) for each branch $b_i$ in TT

(2) do

(3) remove the branch $b_i$ from TT

(4) sort the nodes according to TS

(5) insert sorted $b_i$ into LCPT

(6) done
4.3.1 Global Compact Pattern Tree

The proposed method for computing GCPT contains two phases: (1) Organizing and (2) Merging of Compact Pattern Trees. Assuming ‘n‘ is three (three different local nodes). LDB_1 contains TID’s from first location and LDB_2 contains TID’s from second location and LDB_3 contains TID’s from third location. The first phase is the distributed activity (reordering of LCPT’s using GS sent by central server) at each location based on trigger raised by the central server and sending back LOT’s (Local Organized Trees) to the central location. The second phase is to Merge LOT’s.

4.4 Rule Extraction

In this section we discuss about the proposed model to obtain Multilevel information by deriving local and global knowledge for mining frequent patterns in a distributed environment.

4.4.1 Extraction of Local and Global rules from Distributed Compact Pattern Tree

Let ‘n’ be the number of distributed sites. LDB_1, LDB_2,……,LDB_n are the Local DataBases. LCPT_1, LCPT_2,……,LCPT_n are the Local Compact Pattern Trees, TS_1, TS_2,……TS_n are the Local item Sorted list. GS be the Global item Sorted list. LOT_1, LOT_2,……,LOT_n are the Local Organized CP Trees. LAR_1, LAR_2,……,LAR_n are the Local Association Rules and GAR are the Global Association Rules. The following Table 4.3 shows the notation description of the notation used in algorithms and Table 4.2 is used for illustrations.
### Table 4.3 Notations Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCPT</td>
<td>Global Compact Pattern Tree</td>
</tr>
<tr>
<td>LCPT&lt;sub&gt;i&lt;/sub&gt;</td>
<td>i&lt;sup&gt;th&lt;/sup&gt; node Local Compact Pattern Tree</td>
</tr>
<tr>
<td>TL&lt;sub&gt;i&lt;/sub&gt;</td>
<td>i&lt;sup&gt;th&lt;/sup&gt; node local database item appearance order list</td>
</tr>
<tr>
<td>TS&lt;sub&gt;i&lt;/sub&gt;</td>
<td>TL&lt;sub&gt;i&lt;/sub&gt; sorted in frequency descending order</td>
</tr>
<tr>
<td>GS</td>
<td>Global items sorted in frequency descending order</td>
</tr>
<tr>
<td>LOT&lt;sub&gt;i&lt;/sub&gt;</td>
<td>i&lt;sup&gt;th&lt;/sup&gt; node Local Organized Tree</td>
</tr>
<tr>
<td>Branch b&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Branch of the tree from root to i&lt;sup&gt;th&lt;/sup&gt; leaf node</td>
</tr>
<tr>
<td>TT&lt;sub&gt;i&lt;/sub&gt;</td>
<td>i&lt;sup&gt;th&lt;/sup&gt; node Transactional Tree</td>
</tr>
<tr>
<td>RT&lt;sub&gt;i&lt;/sub&gt;</td>
<td>i&lt;sup&gt;th&lt;/sup&gt; node Restructured Tree</td>
</tr>
</tbody>
</table>

The Figure 4.10 depicts architecture for ‘n’ is 3, local node uses local database such as LDB<sub>1,2,3</sub> and builds LCPT<sub>1,2,3</sub> in respective nodes and communicates TS to Central Server. On receipt of the GS from central server local node reorganized LCPT as LOT and communicates to Central Server. Local node derives the LAR and plugs GAR from Central Server for business strategy building at that location. The Central Server compiles TS received from different local nodes, generates GS and distributes to all the local nodes. On the receipt of LOTs from the local nodes it merges to obtain GCPT. Then the mining process is carried out on GCPT by applying the FP-growth[HPY00] algorithm. Then the frequent patterns are generated by satisfying the predefined minimum support and the Global Association Rules (GAR) are extracted by satisfying predefined minimum confidence. The Central node facilitates GAR for each local node as well as global policy making system. By sending the abstraction of the distributed data to central location, this method achieves privacy preserving without any information loss while generating rules.
- TS—Local database item sorted in frequency descending order
- GS—Global items sorted in frequency descending order
- LOT—Local Organized Tree

Figure 4.10 The Local and Global Association Rules Architecture
4.4.1.1 Extracting Local Rules and Organizing the Local Compact Pattern Trees

This section deals with the organizational aspects of Local Compact Pattern Tree (LCPT) in order to achieve GCPT. From Figures 4.5, 4.7 and 4.9, it is possible to derive local knowledge by mining out on LCPT by applying the FP-growth [HPY00] algorithm. Then the local frequent patterns are generated by satisfying the predefined minimum support and the Local Association Rules (LAR) are extracted by satisfying predefined minimum confidence. In order to obtain GCPT, the approach begins by combining TS\textsubscript{1,2,3} and sorted in frequency descending order. We obtain GS as shown in Fig 4.11. The Central Server distributes GS to three local nodes. The first local node organizes the RT\textsubscript{1} (Figure 4.5) by using GS. Organized tree is nothing but the LOT\textsubscript{1} as shown in Fig 4.11. The same procedure is followed for the remaining two local nodes for obtaining the LOT\textsubscript{2}, LOT\textsubscript{3} are shown in Fig 4.12, 4.13 respectively.

4.4.1.2 Extracting Global Rules

In order to obtain global rules, one has to merge LOT\textsubscript{1}, LOT\textsubscript{2}, LOT\textsubscript{3} of LDB\textsubscript{1,2,3} , the resultant GCPT is shown in Fig 4.14. From Fig. 4.14 the global knowledge is derived. To obtain global knowledge mining process is carried out on GCPT by applying the FP-growth [HPY00] algorithm. Then the global frequent patterns are generated by satisfying the predefined minimum support and the Global Association Rules (GAR) are extracted by satisfying predefined minimum confidence.

In this chapter an algorithms for restructuring, organizing and merging were proposed and shown below. Algorithms for frequent pattern generation [HPY00] and extraction of association rules are given in chapter 3 as algorithms 3.4 and 3.5.
Figure 4.11 LCPT₁ of Fig 4.5 organized – LOT₁

Figure 4.12 LCPT₂ of Fig 4.7 organized - LOT₂

Figure 4.13 LCPT₃ of Fig 4.9 organized – LOT₃
Algorithm 4.3 Global Compact Pattern Tree

**Input:** TS<sub>i</sub>, LCPT<sub>i</sub> where i=1 to n

**Output:** GCPT

**Method:**

1. a) Combine TS<sub>1</sub>, TS<sub>2</sub>…..TS<sub>n</sub> as GS
   b) Sort the GS in frequency descending order
2. for i=1 to n
3. a) LOT<sub>i</sub>= OrganizeLCPT(LCPT<sub>i</sub>, GS)
4. GCPT=MergeLOT(LOT<sub>1</sub>, LOT<sub>2</sub>, LOT<sub>3</sub>… LOT<sub>n</sub>)
5. done

Algorithm 4.4 Organize Tree

**Input:** LCPT: Local Compact Pattern Tree

GS: Global Sorted IList

**Output:** LOT

**Method:**

1. LOT=Call Restructure Tree(LCPT, GS)
2. Return LOT
Algorithm 4.5 Merge Tree (MergeLOT)

Input: Local Organized Trees LOT₁ where i=1 to n

Output: GCPT

Method:

(1) if n=2
   a. GCPT=\textit{Merge}(LOT₁, LOT₂)

(2) else
   a. GCPT₁=\textit{MergeLOT}(LOT₁, LOT₂, LOT₃,..., LOT\_{\lceil n/2 \rceil})
   b. GCPT₂=\textit{MergeLOT}(LOT\_{\lceil n/2 \rceil+1}, ..., LOTₙ)
   c. GCPT=\textit{Merge}(GCPT₁, GCPT₂)
Function *Merge*

**Input:** Local Organized Trees LOT₁, LOT₂

**Output:** GCPT

**Method:**

1. Initialize GCPT = LOT₁
2. for each branch bᵢ in LOT₂
3. do
   a. find branch bⱼ in GCPT having common nodes at beginning with bᵢ
   b. If bⱼ exists then
      i. Let bᵢ = s.yᵢ, bⱼ = s.yⱼ where ‘s’ is maximal prefix containing the common beginning nodes of bᵢ, bⱼ
      ii. If ‘s’ is nonempty
         iii. then
            1. Update frequencies of ‘s’ in bᵢ path with the frequencies of ‘s’ in bⱼ path of LOT₂
            2. Attach yᵢ with its frequencies as a child node to last node of ‘s’ in bⱼ
      c. else
         i. Add bᵢ as a new branch to the root node of GCPT with its frequencies
4. done
4.5 Data Sets and Experimental Set up

In this chapter to extract Local and Global Association Rules. We carried out the experiments with two benchmark data sets Mushroom Data Set and Pima Indian Diabetes Data Set, both obtained from UCI Machine Learning Repositories and also with synthetic data set. The data is converted into transactional database suitable to the experiments. The Mushroom Data Set has 8124 instances, contains 22 attributes whereas in Pima Indian Diabetes Data Set having 768 instances. The following are the characteristics of dataset detailed description of the dataset is available in section 7.3 of chapter 7. The implementation of the existing and proposed algorithms is done using Java Programming Language.

4.5.1 Characteristics of MushRoom Data Set

- **Data Set Characteristics**: Multivariate
- **Number of Instances**: 8124
- **Attribute Characteristics**: Nominal
- **Number of Attributes**: 22
- **Date Donated**: 1987-04-27
- **Missing Values**: Yes
4.5.2 Experimentation with MushRoom Data Set

Considering 1000 transactions from the Mushroom Data Set, the transactions are divided into four parts into unequal distribution such as 200, 300, 400 and 100 transactions respectively.

4.5.2.1 Association Rules from GCPT for Mushroom Data Set

Assuming 'n' is four (four different local nodes) LDB₁ contains 1ˢᵗ part 200 transactions as first node, LDB₂ contains 2ⁿᵈ part 300 transactions as second node, LDB₃ contains 3ʳᵈ part 400 transactions as third node and LDB₄ contains last 100 transactions as fourth node. The distribution of data to four different nodes is sent to different processors and is done using threads. The tree construction is carried out using the OpenMP (Multi-processing tool). The synchronization between processors is achieved internally by these OpenMp. Table 4.4 shows the experimental results of execution time for extracting association rules from Global Compact Pattern Tree for various support thresholds. Table 4.5 is the summary of Table 4.4. Figure 4.15 shows the corresponding graph.
Table 4.4 Execution Time for extracting global association rules from GCPT of Mushroom Dataset for various support thresholds

<table>
<thead>
<tr>
<th>Min. support</th>
<th>min. confidenc.e</th>
<th>TT$_1$(*) 200</th>
<th>RT$_1$(LCPT$_1$) (*) 200</th>
<th>TT$_2$(*) 300</th>
<th>RT$_2$(LCPT$_2$) (*) 300</th>
<th>TT$_3$(*) 400</th>
<th>RT$_3$(LCPT$_3$) (*) 400</th>
<th>TT$_4$(*) 100</th>
<th>RT$_4$(LCPT$_4$) (*) 100</th>
<th>LOT$_1$</th>
<th>LOT$_2$</th>
<th>LOT$_3$</th>
<th>LOT$_4$</th>
<th>Merging (LOT$_1$,2,3)</th>
<th>Time for extracting frequent patter from GCPT</th>
<th>Time for extracting association rules from GCPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>60%</td>
<td>.005</td>
<td>.006</td>
<td>.007</td>
<td>.009</td>
<td>.010</td>
<td>.012</td>
<td>.004</td>
<td>.007</td>
<td>.004</td>
<td>.006</td>
<td>.008</td>
<td>.004</td>
<td>.023</td>
<td>.022</td>
<td>13.963</td>
</tr>
<tr>
<td>45%</td>
<td>65%</td>
<td>.004</td>
<td>.008</td>
<td>.006</td>
<td>.008</td>
<td>.009</td>
<td>.011</td>
<td>.003</td>
<td>.008</td>
<td>.005</td>
<td>.005</td>
<td>.007</td>
<td>.003</td>
<td>.020</td>
<td>.016</td>
<td>6.011</td>
</tr>
<tr>
<td>50%</td>
<td>70%</td>
<td>.005</td>
<td>.007</td>
<td>.007</td>
<td>.009</td>
<td>.008</td>
<td>.011</td>
<td>.004</td>
<td>.007</td>
<td>.003</td>
<td>.004</td>
<td>.006</td>
<td>.004</td>
<td>.018</td>
<td>.015</td>
<td>3.192</td>
</tr>
<tr>
<td>55%</td>
<td>75%</td>
<td>.004</td>
<td>.008</td>
<td>.006</td>
<td>.008</td>
<td>.009</td>
<td>.003</td>
<td>.007</td>
<td>.006</td>
<td>.003</td>
<td>.006</td>
<td>.006</td>
<td>.003</td>
<td>.015</td>
<td>.013</td>
<td>1.813</td>
</tr>
<tr>
<td>60%</td>
<td>80%</td>
<td>.003</td>
<td>.006</td>
<td>.006</td>
<td>.007</td>
<td>.008</td>
<td>.004</td>
<td>.006</td>
<td>.005</td>
<td>.002</td>
<td>.012</td>
<td>.007</td>
<td>.002</td>
<td>.012</td>
<td>.010</td>
<td>0.943</td>
</tr>
<tr>
<td>65%</td>
<td>85%</td>
<td>.003</td>
<td>.006</td>
<td>.007</td>
<td>.007</td>
<td>.008</td>
<td>.004</td>
<td>.005</td>
<td>.003</td>
<td>.005</td>
<td>.005</td>
<td>.005</td>
<td>.002</td>
<td>.009</td>
<td>.007</td>
<td>0.476</td>
</tr>
</tbody>
</table>

- (*) time in seconds
- TT$_1$ takes 200 transactions
- TT$_2$ takes 300 transactions
- TT$_3$ takes 400 transactions
- TT$_4$ takes 100 transactions
Table 4.5 Total time for extracting Global Rules in distributed environment of Mushroom Dataset

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th>Minimum Confidence</th>
<th>Total time for extracting association rules from GCPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>60%</td>
<td>14.038</td>
</tr>
<tr>
<td>45%</td>
<td>65%</td>
<td>6.074</td>
</tr>
<tr>
<td>50%</td>
<td>70%</td>
<td>3.25</td>
</tr>
<tr>
<td>55%</td>
<td>75%</td>
<td>1.865</td>
</tr>
<tr>
<td>60%</td>
<td>80%</td>
<td>0.9888</td>
</tr>
<tr>
<td>65%</td>
<td>85%</td>
<td>0.513</td>
</tr>
</tbody>
</table>

Figure 4.15 Execution time for extracting global association rules from Mushroom Dataset for various support thresholds of GCPT
4.5.2.2 Varying support threshold for Mushroom Dataset

The Figure 4.15 is in agreement with natural phenomenon that number of frequent patterns will monotonically decreasing as threshold increases. Subsequently search time for rule extraction with less number of frequent patterns will be further less. Hence the total rule extaction time is showing higher order monotonic decreasing nature.

4.5.2.3 Sequential and Parallel Implementation

The following is the sequential and parallel implementation of Compact Pattern Tree in order to derive association rules from GCPT. Different sizes of transactions from the Mushroom Dataset are considered and sequentially constructed the CPtree. From the CPTree the association rules are extracted. On the other hand, by considering same size as discussed earlier the experiments is performed in different processors (parallel). The following is the procedure to construct the CPTree in parallel environment. Considering four processors in the experiment, the transactions are divided into four parts. Each part of transactions are sent to each processor to construct the CPtree. Then the procedure which is given in Section 4.4.1.1 and 4.4.1.2 is followed for organizing the local tree and merging the trees in order to obtain GCPT. From GCPT, the association rules are generated. The following Table 4.6 shows the mining time with single and multiple processors. Different sizes of transactions are considered such as 400,800,1000 and 2000, divided among four processor in parallel environment. Figure 4.16 shows the corresponding graph.
Table 4.6
The following table shows the execution time for extracting rules from Mushroom Dataset for the given number of transactions with a support of 40% and a confidence of 60% in sequential and parallel implementation.

<table>
<thead>
<tr>
<th>Number of transactions</th>
<th>TT (*)</th>
<th>RT – CPTree (*)</th>
<th>LOT (*)</th>
<th>Merging Time (*)</th>
<th>Time for extracting frequent patterns from CPTree (*)</th>
<th>Time for extracting association rules from CPTree (*)</th>
<th>Total Time (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>P</td>
<td>S</td>
<td>P</td>
<td>S</td>
<td>P</td>
<td>S</td>
</tr>
<tr>
<td>400</td>
<td>.021</td>
<td>.013</td>
<td>.026</td>
<td>.014</td>
<td>.006</td>
<td>.002</td>
<td>.020</td>
</tr>
<tr>
<td>800</td>
<td>.024</td>
<td>.015</td>
<td>.041</td>
<td>.015</td>
<td>.007</td>
<td>.006</td>
<td>.022</td>
</tr>
<tr>
<td>1000</td>
<td>.026</td>
<td>.017</td>
<td>.047</td>
<td>.020</td>
<td>.008</td>
<td>.008</td>
<td>.026</td>
</tr>
<tr>
<td>2000</td>
<td>.038</td>
<td>.021</td>
<td>.063</td>
<td>.023</td>
<td>.009</td>
<td>.010</td>
<td>.029</td>
</tr>
</tbody>
</table>

S – Sequential  P - Parallel
4.5.2.4 Varying data size of Mushroom Dataset

It is obvious when the transactions are increasing, the elapsed time will also be increasing. It is observed in a single processor system the increases exponential whereas in four processor the rate of increase is gradual. It is also observed that expected reduction in elapsed time is not simply proportional to number of the processors. This is because of the merging and reordering instruction command to be performed at central node.

4.5.3 Characteristics of Pima Indian Diabetes Data Set

<table>
<thead>
<tr>
<th>Data Set Characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Instances</td>
<td>768</td>
</tr>
<tr>
<td>Area</td>
<td>Life</td>
</tr>
<tr>
<td>Attribute Characteristics</td>
<td>Integer, Real</td>
</tr>
<tr>
<td>Number of Attributes</td>
<td>8</td>
</tr>
<tr>
<td>Date Donated</td>
<td>1990-05-09</td>
</tr>
</tbody>
</table>
Missing Values : Yes

4.5.4 Experimentation with Pima Indian Diabetes Data Set

Considering 700 transaction from the Pima Indian Diabetess DataSet, the transactions are divided into four parts into unequal distribution such as 200, 300,150 and 50 transactions respectively.

4.5.4.1 Association Rules from GCPT for Pima Indian Diabetes Dataset

Assuming ’n‘ is four (four different local nodes) LDB\(_1\) contains first part with 200 transactions as first node, LDB\(_2\) contains second part with 300 transactions as second node, LDB\(_3\) contains third part with 150 transactions as third node and LDB\(_4\) contains last with 50 transactions as fourth node. The distribution of data to four different nodes is done using threads where each node is a processor. The tree construction is carried out using the OpenMP (Multi-processing tool). The synchronization between processors is achieved internally by these OpenMp. Table 4.7 shows the experimental results of execution time for extracting association rules from Global Compact Pattern Tree for various support thresholds. Table 4.8 is the summary of Table 4.7. Figure 4.17 shows the corresponding graph.
Table 4.7 Execution Time for extracting global association rules from GCPT of Pima Indian Diabetes Dataset for various support thresholds

<table>
<thead>
<tr>
<th>Min. supp ort</th>
<th>min. confi dence</th>
<th>TT₁ (LCPT₁) (*) 200</th>
<th>RT₁ (LCPT₁) (*) 200</th>
<th>TT₂ (LCPT₂) (*) 300</th>
<th>RT₂ (LCPT₂) (*) 300</th>
<th>TT₃ (LCPT₃) (*) 50</th>
<th>RT₃ (LCPT₃) (*) 50</th>
<th>TT₄ (LCPT₄) (*) 50</th>
<th>RT₄ (LCPT₄) (*) 50</th>
<th>LOT₁</th>
<th>LOT₂</th>
<th>LOT₃</th>
<th>LOT₄</th>
<th>Merging (LOT₁,₂,₃)</th>
<th>Time for extracting frequent patter from GCPT</th>
<th>Time for extracting association rules from GCPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>25%</td>
<td>.007</td>
<td>.009</td>
<td>.008</td>
<td>.012</td>
<td>.006</td>
<td>.007</td>
<td>.004</td>
<td>.003</td>
<td>.005</td>
<td>.006</td>
<td>.003</td>
<td>.002</td>
<td>.015</td>
<td>.017</td>
<td>.007</td>
</tr>
<tr>
<td>6%</td>
<td>26%</td>
<td>.006</td>
<td>.009</td>
<td>.008</td>
<td>.011</td>
<td>.006</td>
<td>.006</td>
<td>.004</td>
<td>.003</td>
<td>.006</td>
<td>.007</td>
<td>.003</td>
<td>.002</td>
<td>.015</td>
<td>.015</td>
<td>.006</td>
</tr>
<tr>
<td>7%</td>
<td>27%</td>
<td>.007</td>
<td>.008</td>
<td>.008</td>
<td>.011</td>
<td>.005</td>
<td>.005</td>
<td>.003</td>
<td>.002</td>
<td>.005</td>
<td>.007</td>
<td>.004</td>
<td>.003</td>
<td>.014</td>
<td>.013</td>
<td>.004</td>
</tr>
<tr>
<td>8%</td>
<td>28%</td>
<td>.005</td>
<td>.007</td>
<td>.006</td>
<td>.009</td>
<td>.005</td>
<td>.005</td>
<td>.004</td>
<td>.003</td>
<td>.005</td>
<td>.006</td>
<td>.004</td>
<td>.002</td>
<td>.013</td>
<td>.013</td>
<td>.003</td>
</tr>
<tr>
<td>9%</td>
<td>29%</td>
<td>.005</td>
<td>.007</td>
<td>.006</td>
<td>.008</td>
<td>.004</td>
<td>.005</td>
<td>.004</td>
<td>.002</td>
<td>.005</td>
<td>.006</td>
<td>.002</td>
<td>.001</td>
<td>.012</td>
<td>.012</td>
<td>.002</td>
</tr>
<tr>
<td>10%</td>
<td>30%</td>
<td>.003</td>
<td>.006</td>
<td>.006</td>
<td>.004</td>
<td>.002</td>
<td>.004</td>
<td>.002</td>
<td>.002</td>
<td>.004</td>
<td>.005</td>
<td>.002</td>
<td>.001</td>
<td>.011</td>
<td>.009</td>
<td>.002</td>
</tr>
</tbody>
</table>

- (*) time in seconds
  - TT₁ takes 200 transactions
  - TT₂ takes 300 transactions
  - TT₃ takes 150 transactions
  - TT₄ takes 50 transactions
Table 4.8 Total time for extracting Global Rules in distributed environment of Pima Indian Diabetes Dataset

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th>Minimum Confidence</th>
<th>Total time for extracting association rules from GCPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>25%</td>
<td>0.065</td>
</tr>
<tr>
<td>6%</td>
<td>26%</td>
<td>0.062</td>
</tr>
<tr>
<td>7%</td>
<td>27%</td>
<td>0.057</td>
</tr>
<tr>
<td>8%</td>
<td>28%</td>
<td>0.05</td>
</tr>
<tr>
<td>9%</td>
<td>29%</td>
<td>0.046</td>
</tr>
<tr>
<td>10%</td>
<td>30%</td>
<td>0.038</td>
</tr>
</tbody>
</table>

Figure 4.17 Execution time for extracting global association rules from GCPT from Pima Indian Diabetes Dataset for various support thresholds
4.5.4.2 Varying support threshold for Pima Indian Diabetes Dataset

From Figure 4.17 it is observed to be same as in section 4.5.2.2

4.5.4.3 Sequential and Parallel Implementation

The sequential and parallel implementation of Compact Pattern Tree in order to derive association rules from GCPT have been presented in this section. Different sizes of transactions from the Pima Indian Diabetes Dataset are considered and sequentially constructed the CPTree. From the CPTree the association rules are extracted. On the other hand, by considering same size of transactions, the transactions are performed parallely. The following is the procedure to construct the CPTree in parallel environment. Four processors are considered in the experiment and the transactions are divided into four parts. Each part of transactions are sent to each processor to construct the CPTree. Then the procedure which is given in Section 4.4.1.1 and 4.4.1.2 is followed for organizing the local tree and merging the trees in order to obtain GCPT. From GCPT, the association rules are generated. The following Table 4.9 shows the mining time with single and multiple processors. Different sizes of transactions are considered such as 400,800,1000 and 2000. These transactions are divided among the four processor in parallel environment.
Figure 4.18 Execution time for extracting global association rules from Pima Indian Diabetes Dataset by varying data size

4.5.4.4 Varying data set size of Pima Indian Diabetes Dataset

From Figure 4.18 it is observed to be same as in section 4.5.2.4
Table 4.9

The following table shows the execution time for extracting rules from pima Indian diabetes dataset for the given number of transactions with a support of 5% and a confidence of 25% in sequential and parallel implementation.

<table>
<thead>
<tr>
<th>Number of transactions</th>
<th>TT (*)</th>
<th>RT – CPTree (*)</th>
<th>LOT (*)</th>
<th>Merging Time (*)</th>
<th>Time for extracting frequent patterns from CPTree (*)</th>
<th>Time for extracting association rules from CPTree (*)</th>
<th>Total Time (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>.017</td>
<td>.012</td>
<td>.023</td>
<td>.004</td>
<td>.010</td>
<td>.008</td>
<td>.019</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>.029</td>
<td>.013</td>
<td>.026</td>
<td>.006</td>
<td>.011</td>
<td>.002</td>
<td>.013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>.032</td>
<td>.014</td>
<td>.032</td>
<td>.016</td>
<td>.015</td>
<td>.003</td>
<td>.018</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>.036</td>
<td>.020</td>
<td>.041</td>
<td>.021</td>
<td>.022</td>
<td>.003</td>
<td>.025</td>
</tr>
</tbody>
</table>

4