Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# CATD-LPT-CFPM- Cluster Aware Top-Down Linear Prefix Tree for Closed Frequent Pattern Mining

**M Sinthuja**

sinthuja.m@vit.ac.in

Vellore Institute of Technology

**P Saranya**

Vellore Institute of Technology

**M. Diviya**

Vellore Institute of Technology

**Article**

**Keywords:** Association rule mining, Closed frequent patterns, Closedness Pruning, Clustering, Frequent patterns, Top-Down Closedness Pruning

**Posted Date:** July 30th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-10346251/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# CATD-LPT-CFPM- Cluster Aware Top-Down Linear Prefix Tree for Closed Frequent Pattern Mining

Sinthuja M[1], P Saranya[2], M. Diviya[3]

[1,2,3]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India
*Corresponding author(s). E-mail(s): sinthuja.m@vit.ac.in;

Contributing authors: saranya.p@vit.ac.in; diviya.m@vit.ac.in

***Abstract***— In line with mining association rule, frequent pattern mining is an essential process. It discovers frequently used patterns in the transaction dataset and extracts all associated rules. Numerous duplicate patterns are found when mining frequently occurring itemsets, leading to an increase in meaningless itemsets. Maximum frequent pattern and closed frequent pattern are the two techniques used in protracted frequent pattern mining. The maximal frequent pattern, regrettably, suggests that every one of its subsets is frequent and that the minimal support criterion is not crossed by their support. The disadvantage of maximal frequent pattern mining is that the precise value of the support is unknown. In closed frequent pattern mining single search is applied for the entire transaction database resulting in higher search space, consumption of memory, time and computational overhead specifically for the dense datasets. To address this limitation, a novel technique of Cluster-Aware Top-Down Linear Prefix Tree for Closed Frequent Pattern Mining (CATD-LPT-CFPM ) framework is proposed. Similar transactions from the ordered transaction database are grouped into clusters to divide the search space and to reduce the redundant processing. For each cluster separate linear prefix tree is constructed for more focused pattern research. The proposed framework adopts a multi-level pruning technique that includes cluster elimination pruning, top-down support pruning, and a novel Top-Down Closedness Pruning (TDCP) mechanism. TDCP recognize the non-closed patterns during traversal by comparing the parent and the child node support values thereby minimizing the storing and analysis of duplicate itemsets. Explored local closed patterns from each cluster are merged and verified to produce the final global closed pattern. Experimental evaluation on benchmark dataset shows that the proposed CATD-LPT-CFPM technique minimizes the number of examined nodes and organization of search space is enhanced by clustering and top-down tree traversal. While comparing the proposed CATD-LPT-CFPM with the traditional closed frequent patterns the proposed shows efficient memory storage and execution time is relatively higher due to additional clustering and prefix tree construction overhead based in datasets characteristics.

***Keywords:*** Association rule mining, Closed frequent patterns, Closedness Pruning, Clustering, Frequent patterns, Top-Down Closedness Pruning

## 1. Introduction:

In Association rule mining, frequent itemsets mining (FIM) is a critical activity for identifying all frequent itemsets among a group of items in D. FIM's exponential complexity motivates academics to create new methods to enhance it. Traditional algorithms like Apriori [1], ECLAT [2], FP-growth [2], Quick-Apriori [3] are time-consuming and memory-intensive for mining large datasets. Apriori-based techniques have I/O overhead from multiple dataset scans, along with computational cost and and memory costs associated with creating a large number of candidates. FP tree-based systems include extensive procedures for modifying trees [4, 5, 6, 7]. Tree-based approaches [8, 9, 10, 11] of the FP-growth algorithm consume more time in building the tree. Frequent itemsets are just the itemsets that occur frequently in the transactional database. Mining frequent itemsets is improved by a cutting-edge technique, and an active data structure is also established. Historically, many approaches for frequent itemset exploration were investigated. Generating itemsets with support higher than or equal to a minimal support threshold value is the aim of frequent pattern mining. While mining frequent itemsets, there are many duplicate itemsets, resulting in more meaningless itemsets. Maximum frequent itemset and closed frequent itemset are the two methods used in prolonged frequent itemset mining. Unfortunately, the maximal frequent itemset suggests that every subset inside it is frequent and that the value of its support does not drop below the minimum support criterion. However, the actual value of the support is unknown. Current research focuses on the full closed frequent itemset, which may minimize the amount of itemsets while retaining all data. When the total support of a frequent itemset X's supersets is less than X itself, the frequent itemset X is considered closed. This indicates that itemset X is frequently closed. A significant number of itemsets will be produced by mining the entire collection of often occurring itemsets. Fortunately, this issue has been reduced to the mining of closed frequent itemsets. This closed frequent itemset contains all data for its relevant frequent itemsets. It promotes efficiency and efficacy while providing a concise representation.

Due to the rapid growth of transactional databases, the level of difficulty of closed frequent pattern mining has been significantly increased that created a significant challenge in the efficient exploration of meaningful patterns

with reduced computational cost. When it comes to dense and big datasets the existing closed frequent pattern mining frequently processes the entire transaction database with single search structure, resulting in more search space, execution time, and consumption of memory. As a result, an efficient mining approach that can minimize the unnecessary exploration, reducing the storage needs and increasing minimum speed is essential.

To get beyond these limitations, this research paper proposes a novel Cluster-Aware Top-Down Linear Prefix Tree algorithm. The proposed algorithm combines transaction clustering with multi-level pruning approach to explore frequent closed patterns. The following is the summary of proposed algorithm:

i. Similar transactions in the database are clustered together. Cluster that cannot satisfy the minimum support threshold are pruned by cluster elimination pruning that reduces the initial search space.
ii. Instead of building a single global search structure, Linear Prefix Tree is built for all clusters allowing for focused pattern investigation.
iii. Each clustered LPTs are traversed by top-down approach. Nodes which are less than minimum support are pruned. To generate local closed pattern, top down closedness pruning is adopted to compare the value of parent and the child nodes.
iv. Explored local frequent patterns are merged through global support calculation and closedness checking to generate final closed frequent patterns.

## 2. Related works:

This study uses the closed candidates-based incremental frequent itemset mining approach (CC-IFIM) [12][55] to produce frequently recurring global itemsets with a goal of reducing the number of candidates created and improving their precision. Before doing further calculations, the suggested method eliminates many candidates generated in previous rounds subsequently. Instead of utilizing all candidates, simply the local closed candidates were used to assess the similarity across partitions to improve the precision of the support count computation.

The MAAP algorithm makes use of Apriori's traits to improve performance overall. High-level itemsets introduce low-level itemsets, and the Apriori technique blends both types of itemsets. It generates candidate itemset that consumes more consumption of memory and execution time. MAAP generates new association rules by comparing frequent patterns to fresh transactions. MAAP can enhance FUP performance [13], although it still requires re-scanning the dataset. Zaki et al. [14] introduced Eclat, it generates candidate itemset where it considers the transaction id's, a DFS method that organizes data using a linked list. The NSFI approach for mining frequent itemsets was developed by Vo et al. [15]. This technique makes use of the N-list structure and subsume ideas to improve both execution time and memory consumption. The NegFIN technique [16] was introduced by Aryabarzana et al. to mine frequent items with a novel, efficient data structure called NegNodeset, which bitmap-encodes the nodes in a prefix tree. Researchers have studied the extending concerns of itemset mining, in addition to its widespread application. To generate closed frequent itemset vertical database format is taken and employed into the mining process [17].

Linear prefix tree is used in this paper where they adopt top-down mining approach to discover frequent patterns. Performance is assessed using benchmark datasets. In terms of memory and execution time, it is comparable to the current method [18]. Exploring frequent closed itemsets (FCIs) [19,20], maximum frequent itemsets [21,22,29] the generated frequent item by the classic algorithm is given as input to the maximal frequent itemset mining algorithm. In maximal frequent itemset mining it checks for the superset and identifies the maximal itemset the drawback with it is it does not consider the count of the item. New algorithm of parallel approach called parallel dynamic bit vector for closed sequential frequent pattern is introduced (pDBV-FCSP) [23]. It reduces the search space, and closed checking is performed to reduce the runtime for sequential pattern mining. Surve, research ideas, terminologies and relationships to other patterns are discussed [24].

In this study, the parallel approach of MCM-SPADE is introduced to accelerate the earlier methods of SPADE and CM-SPADE. As a multi-threading technique for SPM with an enormous database, it is a multi-core processing system. The proposed method uses the vertical data format and stores co-occurrence data in a data structure named CMAP (Co-occurrence MAP) [25]. In order to get top-k frequent patterns, prefix partitioning is introduced in this study. Partitioning is done through vertical mining. In order to efficiently mine top-k frequent itemsets, this work presents a PTF algorithm based on prefix partitioning, where each prefix-based partition preserves the transactions having the same prefix item. Vertical mining is performed that gives better performance [26]. Utility mining is performed for mining closed frequent itemset [27]. The problem of mining closed HUIs (CHUIs), a scalable, succinct, and concise version of HUIs, in dynamical unit profit databases is defined in this work. An innovative

algorithm named iEFIM-Closed is presented, based on a recently proposed utility measure. Based on a specialized data structure with an innovative efficient file representation method for compressed representation and an optimized file scan technique that minimised both database scan cost as well as the overhead involved in database processing during mining, a new utility measure was developed. By adopting the structure of linear prefix tree maximal frequent patterns are explored and the evaluations are determined in the aspects of execution time and consumption of memory. They have used the technique of maximal checking to explore maximal frequent patterns [29].

## 3. Definitions and Problem statements

Consider the following: $T$ is a set of items in transaction $T$ such that $T \subseteq I$ in a transactional database $D$. Let $I = \{it1, it2 \ldots itm\}$ be a group of m unique items. In $D$, the total of all transactions is expressed as $|D|$. Every transaction is assigned to a different identification known as *Tid.* Itemset is referred as $X \subseteq I$. The total amount of $D$ transactions containing $X$ is known as the occurrence of $X$, and the percentage of $D$ transactions that contain $X$ is known as *support(X). count(X) = support(X)* * $|D|$ is the relationship between *count(X) and support(X).*

**Definition 1: Clustering**
The transaction database is spited into k clusters $C=\{c1,c2 \ldots ck\}$ where each cluster has transactions with identical characteristics. The process of clustering minimizes the overall search space by allowing limited pattern mining within each cluster.

**Definition 2 : Cluster Elimination Pruning**

For every cluster Ci, the support of the cluster is denoted by the total number of transactions contained in that cluster, it is represented by |Ci|. Cluster elimination pruning, which does not produce any frequent itemsets, is used to prune clusters based on the minimum support threshold. Thus, reducing the number of linear tree construction and reducing the memory consumption.

$$If\ |Ci| < MinSup, \quad Ci \rightarrow Pruned$$

**Definition 3: Frequent Itemset**

An itemset X is known as frequent if it satisfies the minimum support threshold s

$$Support(X) >= s$$

**Definition 4: Closed Frequent Itemset**

If there is no proper superset X' then a frequent itemset X is known as closed frequent itemset. If an itemset ***X'*** fails to exist such that ***X⊂X'*** and ***support (X') = support(X)*****,** then a frequent itemset ***X*** is considered closed. ***X*** is then recognized as a closed frequent itemset.

Property 1: The item in L1 is the basis for every frequent closed itemset, and the collection of all frequent items in L1 is ordered in descending order.
Property 2: Every subset of a frequent itemset is also frequent.
Property 3: If not frequent items, supersets are all infrequent.

**Definition 5: Linear Prefix Tree**

Transactions within the cluster are inserted into the linear prefix tree. Transaction that has similar prefixes are merged into the same path that reduces the requirement of storage and enabling frequent and closed pattern generation [28].

## 4. Proposed Cluster-Aware Top-Down Linear Prefix Tree for Mining Closed Frequent Patterns (CATD-LPT-CFPM)

The proposed Cluster-Aware Top-Down Linear Prefix Tree (CATD-LPT-CFPM) is presented in this section for exploring closed frequent patterns. When working with the large scale and dense database the classical algorithms suffer from more consumption of memory, time, high computational complexity, redundant scanning of database.

To address these limitations, the proposed CATD-LPT-CFPM adopts clustering of transactions, cluster elimination pruning, construction of compact linear prefix tree, top-down support-based pruning, and closedness verification to discover efficient global set of closed frequent patterns.

The proposed CATD-LPT-CFPM consists of several stages: (1) Transaction Preprocessing (2) Transaction Clustering (3) Cluster Eliminate Pruning (CEP) (4) construction of cluster-wise Linear Prefix Tree (5) Top-Down support- based frequent pattern generation (6) Top-Down Closedness Pruning (7) Global support and Closedness Verification. Flow diagram of the proposed CATD-LPT-CFPM is shown in figure 1.

### 4.1 Transaction Preprocessing and ordering of items

Table 1 depicts the transaction database that represent the transaction id and the items acquired for each transaction. Compute the support count of each item and based on the minimum support threshold the item that is lower than the minimum support is pruned. The rest of the frequent patterns are arranged in chronological order of the support as shown in the table as ordered items. This process enhances the compactness of the CATD-LPT-CFPM by improving similar prefix sharing among the transactions.

**Table. 1** Table for Transactional dataset

| Tx id | Items Acquired | Ordered Items |
|---|---|---|
| 1 | Cable, Driver, HD, USB, SSD, Mouse | USB, HD, Cable, SSD |
| 2 | Cable, HD, Driver | HD, Cable, Driver |
| 3 | USB, Choc | USB |
| 4 | Cable, HD, RAM, USB, Driver | USB, HD, Cable, Driver, RAM |
| 5 | USB, HD, Cable, Driver, Wire | USB, HD, Cable, Driver |
| 6 | USB, CPU | USB |
| 7 | Cable, Fan, HD, USB, SSD, EarPods | USB, HD, Cable, SSD |
| 8 | HD, Cable, RAM | HD, Cable, RAM |
| 9 | Cable, HD, USB, Driver, Headphone | USB, HD, Cable, Driver |
| 10 | USB , HD, Cable, Driver, Pen drive | USB, HD, Cable, Driver |

### 4.2 Transaction Clustering

The procedure of putting together related transactions into distinct clusters according to shared prefixes is known as transaction clustering. The proposed CATD-LPT-CFPM partition the database into multiple and more homogeneous clusters that avoids the search space of mining entire transaction database.

For instance, in an ordered transaction database with a minimum support threshold of 2, transactions that begin with USB is the most dominant prefix. Thereby grouping the prefix USB related transaction in Cluster C1 and prefix HD are sorted into Cluster C2. Eight transactions are grouped in C1. Transactions that share the similar prefix path can be stored in CATD-LPT-CFPM as they have similar starting item. Therefore, the support count of C1 is 8. Transaction grouped in C1 are T1, T3, T4, T5, T6, T7, T9, T10 as shown in table 2. Two transactions are grouped in C2. They share the common prefix path HD. Transactions grouped in C2 are T2, T8 as shown in table 3.

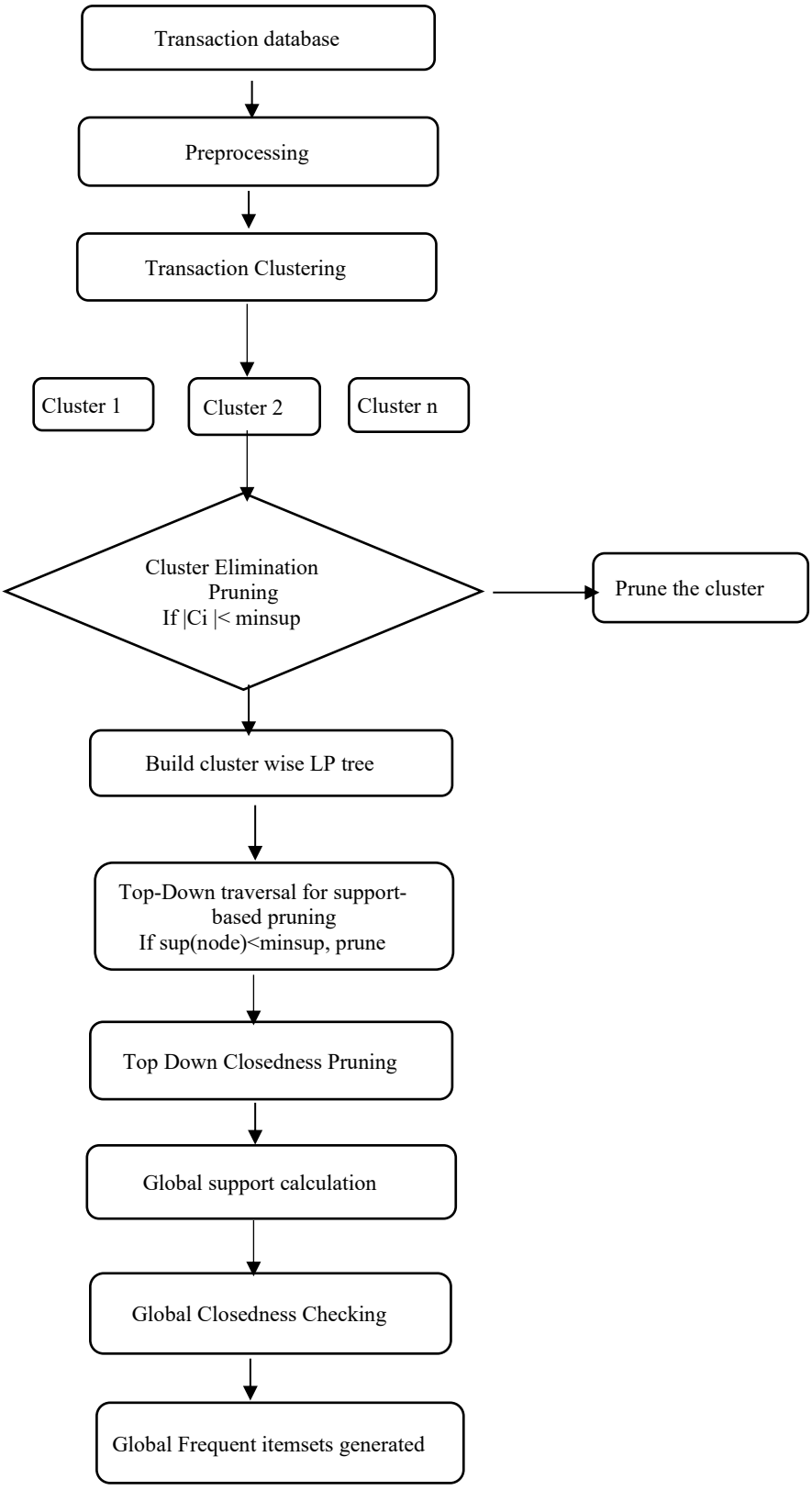



Fig 1. Flow diagram of the proposed CATD-LPT-CFPM

**Table 2: Cluster 1**

| Tid | Items |
|---|---|
| T1 | USB, HD, Cable, SSD |
| T3 | USB |
| T4 | USB, HD, Cable, Driver, RAM |
| T5 | USB, HD, Cable, Driver |
| T6 | USB |
| T7 | USB, HD, Cable, SSD |
| T9 | USB, HD, Cable, Driver |
| T10 | USB, HD, Cable, Driver |

**Table 3: Cluster 2**

| Tid | Items |
|---|---|
| T2 | HD, Cable, Driver |
| T8 | HD, Cable, RAM |

### 4.3 Cluster Elimination Pruning (CEP)

Next to the phase of transaction clustering, the proposed CATD-LPT-CFPM adopts cluster elimination pruning to identify the cluster that is not promising for the construction of prefix trees. Reducing search space, memory, time and ignoring the unwanted construction of CATD-LPT-CFPM is a key objective of CEP. The generated cluster is verified based on its capacity to provide adequate support for the generation of frequent patterns. The clusters that have the capacity to generate frequent patterns by satisfying the minimum support threshold are kept for future processes, else pruned to prevent the unwanted mining process. From table 1, it is observed that C1 has 8 transactions which are higher than the minimum support threshold (8>=2). Thus, cluster 1 is retained for the construction of CATD-LPT-CFPM. From table 2, it is noticed that C2 has 2 maximum possible support of patterns 8 which shows superior to the minimum support threshold (2>=2). Thus, cluster 2 is retained for the construction of CATD-LPT-CFPM.

***Cluster 1*** $|C1| = 8 >= 2$ → ***retain***

***Cluster 2*** $|C2| = 2 >= 2$ → ***retain***

### 4.4 Cluster-wise LP-tree Construction

Following the cluster elimination pruning, the retained clusters are further processed for the construction of linear prefix tree. The aim of the cluster-wise LP-tree is to express transactions in a portable hierarchical structure that shares similar prefixes among the transactions. Thus, it ignores the saving of the separate sequence for redundant patterns and lowers the need for frequent pattern mining. In the proposed CATD-LPT-CFPM, every transaction in the cluster is added into the LP-tree as node. Every node has the details of item name, support count, node link and flag value. At the time of insertion if any of the transactions have a similar prefix then a common path is generated and the support count of each node is increased. If the transaction has new sequence a new branch is created from the existing node. Thus, a compact LP-tree is created for all transactions within a single cluster along with their support details. In the same way, LP-tree is created for all clusters. Algorithm 1 outlines the cluster formation and LP-Tree Construction

From table 1 each transaction is inserted into the LP-tree. In C1 the first transaction {USB, HD, Cable, Fan, SSD} is inserted into the LP-tree with count 1, node link is null as no link for further node and set flag as false as no branch nodes are generated as shown in figure 2. From Second transaction {USB} is inserted into the LP-tree. As it shares the similar item with T1 only the count is upgraded to 2 without creating the new branch that reduces the storing space. Branch node list (BNL) is created for the LP-tree.

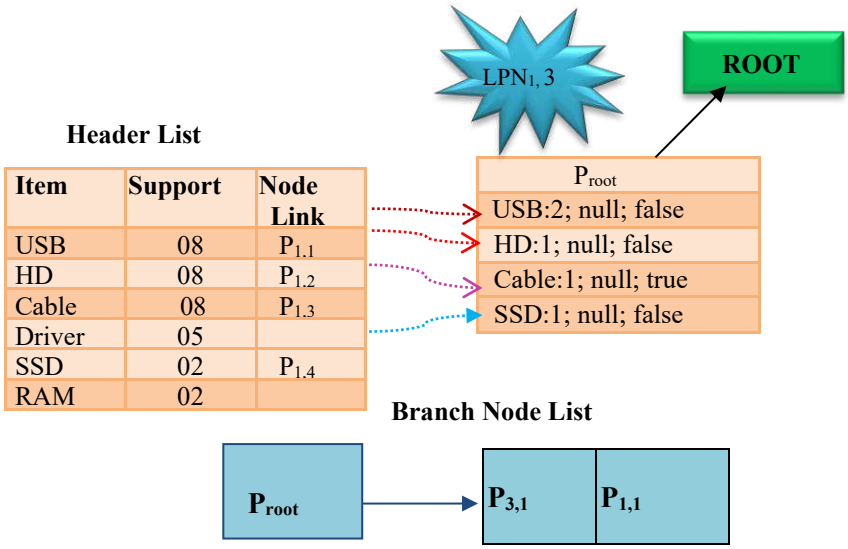



**Fig 2. Insertion of cluster 1: first and third transaction**

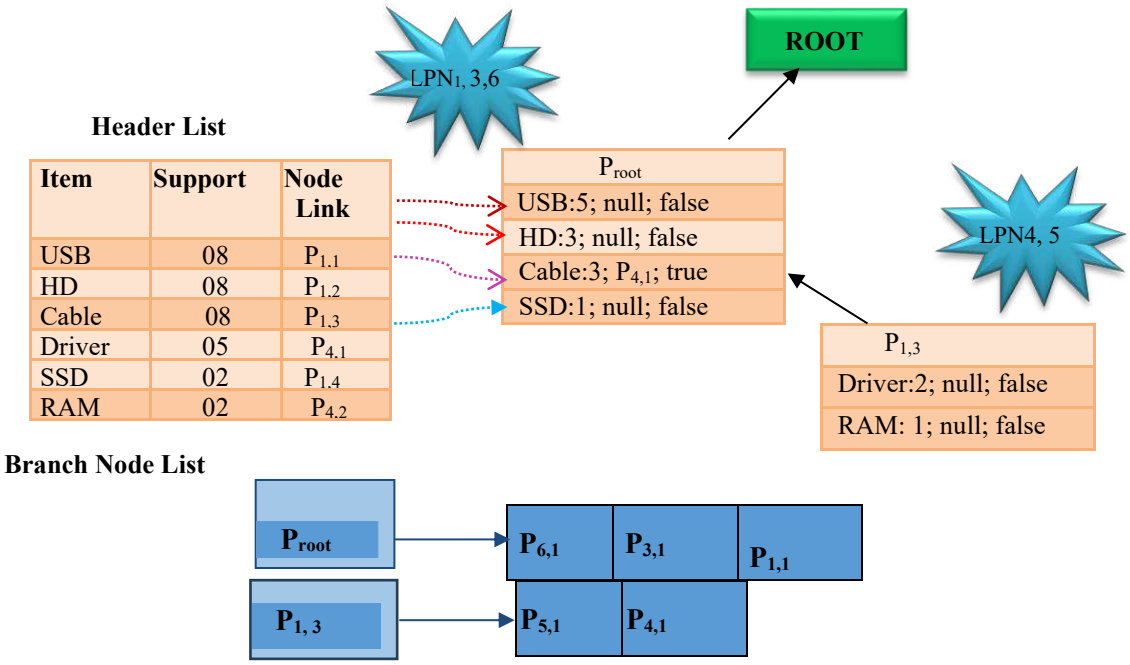



**Fig 3. Insertion of cluster 1: fourth, fifth and sixth transaction**

As the fourth transaction {USB, HD, Cable, Driver, RAM} shares its prefix path with the already existing one the count of USB is incremented to 3, HD to 2 and Cable to 2 but Driver is a new item so creating a node from Driver and generating two nodes Driver and Ram with count 1, as no node link is present it is marked as null and flag is set as false as no branch is created. In cable, the node link is changed to P4,1 as it has link to the next node and flag is changed to true as it has branch node. For the rest of the items node link and flag is set to null and false. Similarly, the transactions fifth {USB, HD, Cable, Driver} and sixth {USB} are added into the tree as shown in figure 3.

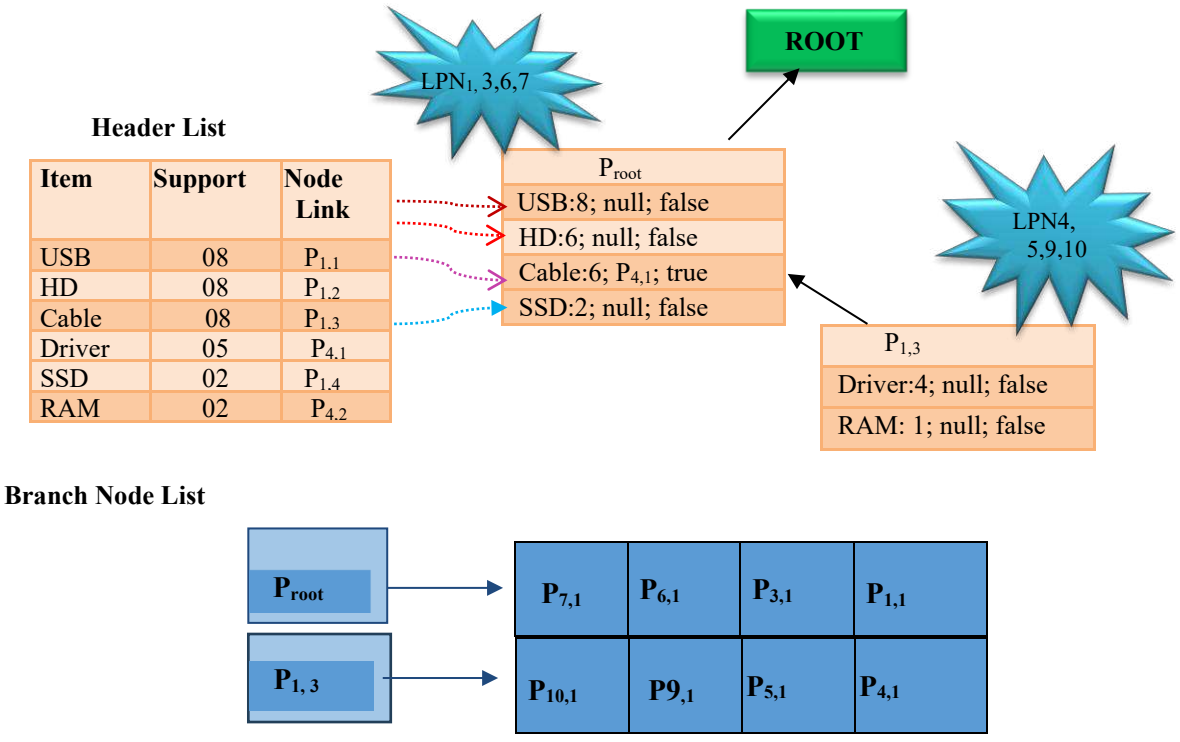



**Fig 4. Insertion of cluster 1: seventh, nineth and tenth transaction**

As the seventh transaction USB, HD, Cable, SSD shares its prefix, the count of USB is incremented to 6 , HD incremented to 4, cable incremented to 4 and SDD incremented to 2. Thus, there is no change in node link and flag. For the ninth and tenth, transaction is USB, HD, Cable, Driver as it shares with prefix only the count is updated as shown in figure 4.

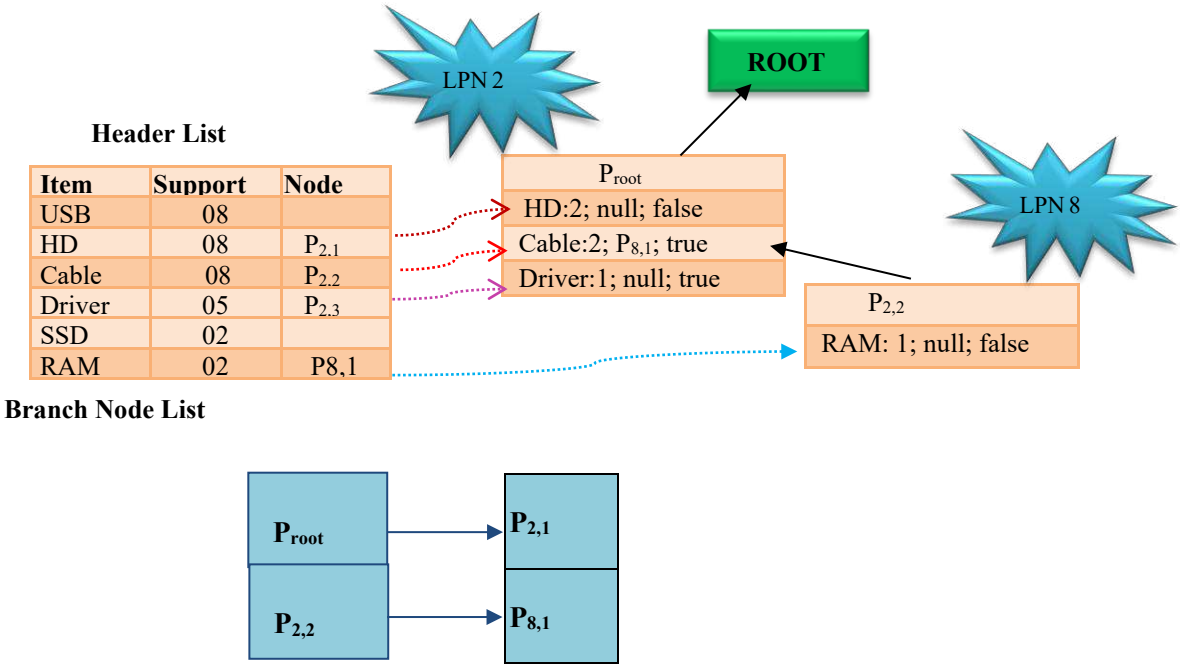



**Fig 5. Insertion of cluster 2: second and eighth transaction**

With reference to table 2, cluster 2 transactions are inserted into the separate LP-tree in figure 5. Second transaction contains the patterns of HD, Cable, Driver. It is added into the LP tree with support count of 1 for all. Node link to null and flag false as no branch exists. For the eighth transaction HD, Cable and RAM, HD and Cable

share with the existing tree so count of HD and cable are incremented to 2 and a fresh node is created from cable and Ram is added into the tree with count 1, node link null and flag as false. As new node is created from cable the node link is changed from null to $P_{8,1}$ and flag is updated to true.

### 4.5 Top-Down traversal for support-based pruning

After the construction of LP-tree top-down approach is applied to traverse the tree from root node to the leaf node. At the time traversal, each items support is compared with the minimum support threshold. Items that meet the minimum support threshold are recognized as frequent itemsets where the lower support nodes lower than minimum support threshold are eliminated immediately. As the pruning is performed in early stages it reduces the volume of candidate patterns and increases the efficiency of mining.

*If support(node) < MinSup* → *Prune*

Top-down traversal is applied to the cluster 1 as shown in figure 6. First node of the tree is {USB} with support count 8. Since 8>=2, {USB: 8} is frequent. By walking to the child node {USB, HD} with support count 6. Since 6>=2 {USB, HD: 6} is frequent. Next is {USD, HD, Cable} with support count 6. Since 6>=2 {USB, HD, Cable: 6} is frequent. Proceed with the next child node {USB, HD, Cable, SDD} with support count 2. As 2<=2 {USB, HD, Cable, SDD:2} is frequent. Explore to another child node {USB, HD, Cable, Driver} with support count 4. As 4<=2 {USB, HD, Cable, Driver: 4} is frequent. The next child node {USB, HD, Cable, Driver, RAM} with support count 1. As 1 <= 2 {USB, HD, Cable, Driver, RAM :1} is not frequent as shown in Fig 6. The generated frequent patterns are {USB: 8}, {USB, HD: 6}, {USD, HD, Cable: 6}, {USB, HD, Cable, SDD:2}, {USB, HD, Cable, Driver: 4}.

From the fig 7, top-down traversal is applied to the cluster 2. First node of the tree is {HD} with support count 2. Since 2>=2, {HD: 2} is frequent. Moving to the child node {HD, Cable} with support count 2. Since 2>=2, {HD, Cable:2} is frequent. The next child node {HD, Cable, Driver} with support count 1. Since 1is not greater than minimum support it is not frequent as shown in fig.5. The child node {HD, Cable, RAM} with support count 1. Since 1 is not greater than minimum support it is not frequent. The generated frequent patterns of cluster 2 are {HD: 2}and {HD, Cable:2}. Algorithm 2 outlines the frequent pattern mining using top-down LP-tree traversal.

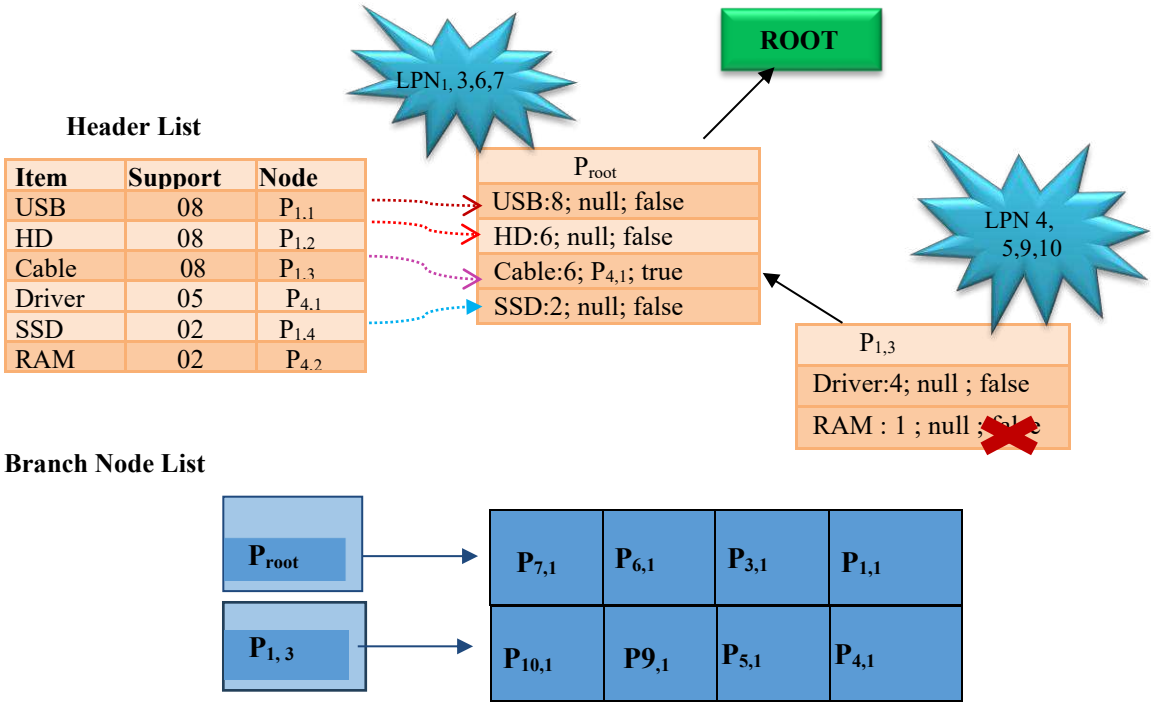



**Fig 6. Top-Down traversal for support-based pruning for cluster 1**

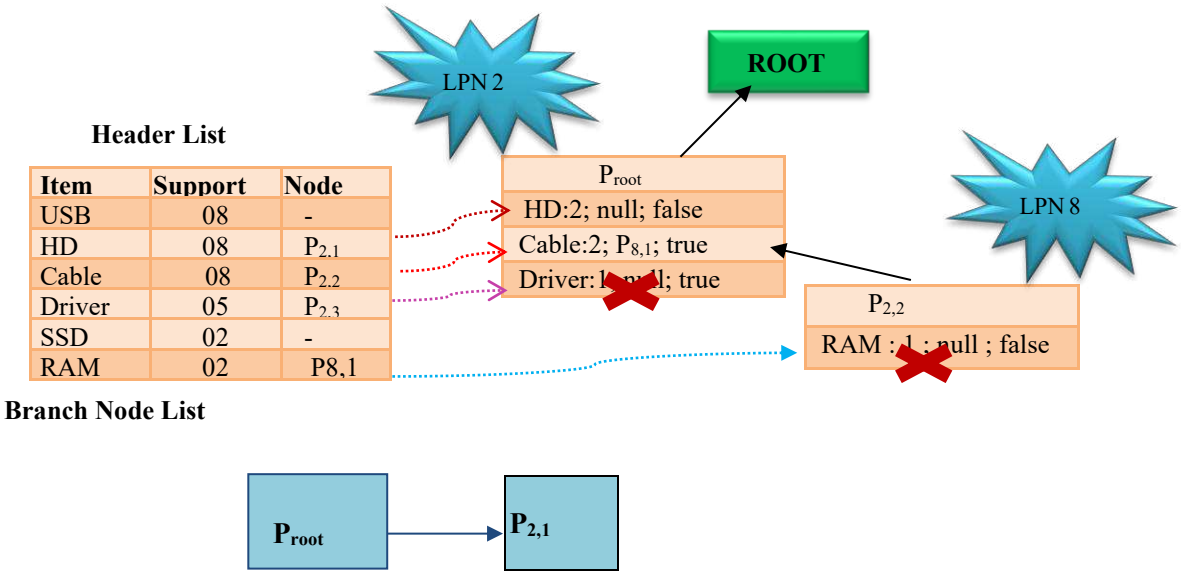



**Fig 7. Top-Down traversal for support-based pruning for cluster 2**

### 4.6 Top-Down Closedness Pruning

After performing support-based pruning still some redundant information may exist. To remove the redundancy the proposed CATD-LPT-CFPM adopts a top down closedness pruning. This approach is essential for pruning the frequent patterns where the superset has the same support values. A simplified but comprehensive representation of the frequent pattern set is obtained by retaining the closed frequent itemsets. In this process, all frequent patterns are compared with their supersets in a top-down approach. If the superset has the same support as the subset, then the subset is removed as non-closed. Local closed frequent patterns are retained in this process that reduces the search space of patterns and memory overhead while maintaining the entire frequency information.

*If Support (Subset) = Support (Superset)* → *non-closed*
*If Support (Subset) > Support (Superset)* → *closed frequent pattern*

From cluster 1 the generated frequent patterns {USB:8) is compared with its superset {USB, HD:6}. If the support count of subset {USB:8} is not equal to superset {USB, HD:6} then {USB:8} is closed.

***{USB:8} > {USB, HD:6}*** → ***Closed frequent pattern***

The pattern {USB, HD:6} is compared with its superset {USB, HD, Cable:6}.
***{USB, HD:6} = {USB, HD, Cable:6}*** → ***non-closed***

The pattern {USB, HD, Cable:6} is compared with its superset {USB, HD, Cable, SDD: 2} and {USB, HD, Cable, Driver: 4}. No superset with support 6.

***{USB, HD, Cable:6} > {USB, HD, Cable, SDD: 2}, {USB, HD, Cable, Driver: 4}*** → ***Closed frequent pattern***

The pattern {USB, HD, Cable, SDD:2} has no superset with the same support so it is closed.
***{USB, HD, Cable, SDD:2}*** → ***Closed frequent pattern***

The pattern {USB, HD, Cable, Driver: 4} has no superset with the same support so it is closed.
***{USB, HD, Cable, Driver: 4}*** → ***Closed frequent pattern***

From cluster 2, the frequent pattern {HD} support count 2 is compared with superset {HD, Cable} with support count 2.

{HD: 2} = {HD, Cable: 2} both have same support so it is not closed.

***{HD, Cable: 2}*** → ***Closed frequent pattern***

The child pattern {HD, Cable: 2} has no superset so it is closed.

### 4.7 Global Support Calculation

Following the top-down closedness pruning local closed frequent patterns for each cluster is generated. These local support values, however, only show the frequency of patterns within certain clusters; they do not show how frequently these patterns occur throughout the entire transaction database. Thus, the support contributions of the same itemsets from all clusters are combined in the proposed CATD-LPT-CFPM to perform Global Support Calculation. Algorithm 3 outlines the global support computation and closed frequent pattern mining.

Each local closed frequent itemset support is calculated again by scanning the entire transaction database and calculating the actual count.

*Global support for {USB: 8}* → *T1, T3, T4, T5, T6, T7, T9, T10*
*Global support for {USB, HD, Cable: 6}* → *T1, T4, T5, T7, T9, T10*
*Global support for {USB, HD, Cable, SDD: 2}* →*T1, T7*
*Global support for {USB, HD, Cable, Driver: 4} T4, T5, T9, T10*
*Global support for {HD, Cable: 2}*→*T1, T2, T4, T5, T7, T8, T9, T10*
Since the minimum support is 2 all the patterns above satisfy global support >= minimum support.

### 4.8 Global Closedness Checking

Global closedness checking is the final stage of pruning of the proposed CATD-LPT-CFPM. Globally generated frequent patterns {USB}: 8, HD, Cable: 8, {USB, HD, Cable}: 6, {USB, HD, Cable, Driver: 4, and {USB, HD, Cable, Fan, SSD: 2 from global support calculation are taken as input for global closedness checking. Here, globally generated frequent patterns are compared with respective superset using global support values. If the superset and subset have the same support, the subset is pruned as a redundant non-closed pattern. This process guarantees that final patterns contain only compact and complete global closed frequent patterns while preserving all frequency information.

*If* ***Support (Subset) = Support (Superset)*** → *Subset is redundant and removed.*
*If* ***Support (Subset) > Support (Superset)*** → *Subset contains additional occurrence information and is retained as a closed frequent itemset.*

The pattern {USB: 8} is compared with the superset that has USB along with other items like { USB, HD, Cable} with global support 6, {USB, HD, Cable, Driver} with support 4, {USB, HD, Cable, SSD}with support 2, {USB, HD} with support count 6. All the supersets of {USB} have lower support values and not same so the pattern {USB} retains as global closed frequent pattern. The pattern {USB, HD, Cable: 6} compared with the superset that has {USB, HD, Cable} along with other items like {USB, HD, Cable, SSD}with support 2 and {USB, HD, Cable, Driver} with support 4. No superset has the same support hence {USB, HD, Cable: 6} is retained as global closed frequent pattern.

The pattern {USB, HD, Cable, SSD:2} is compared with the superset that has {USB, HD, Cable, SSD} with other items. Unfortunately, no superset exists so the pattern {USB, HD, Cable, SSD:2}is retained as global closed frequent pattern. The pattern {USB, HD, Cable, Driver: 4}is compared with the superset that has {USB, HD, Cable, Driver} along with other items like {USB, HD, Cable, Driver, RAM} with support count 1. No superset has same support as the pattern {USB, HD, Cable, Driver: 4} retained as global closed frequent pattern. The pattern {HD, Cable:2} is is compared with the superset {HD, Cable, Driver} with global support 1, and {HD, Cable, RAM} with global Support 1. No supersets support is same, so it retains as global closed frequent pattern.

Final global frequent itemset generated are {USB: 8}, {USB, HD, Cable: 6},{USB, HD, Cable, SDD: 2}, {USB, HD, Cable, Driver: 4}, {HD, Cable: 2}.

**Algorithm 1** Cluster Formation and LP-Tree Construction

**Require:** Transaction Database $D$, Minimum Support Threshold $MinSup$
**Ensure:** Set of Cluster-wise LP Trees $LP = \{LP_1, LP_2, \ldots, LP_k\}$

**Terminology:**
$C = \{C_1, C_2, \ldots, C_k\}$: Set of clusters
$L_1$: Frequent 1-itemsets
$LP_i$: Linear Prefix Tree of cluster $C_i$
**Step 1:**
Partition transaction database $D$ into $k$ clusters.
$C = \{C_1, C_2, \ldots, C_k\}$
**Step 2:**
**for** each cluster $C_i$ **do**
    **if** $|C_i| < MinSup$ **then**
        prune cluster $C_i$
    **else**
        retain cluster $C_i$
    **end if**
**end for**
**Step 3:**
**for** each retained cluster $C_i$ **do**
    scan transactions
    generate frequent 1-itemsets $L_1$
    remove infrequent items
    sort frequent items in descending support order
    construct Cluster-wise Linear Prefix Tree $LP_i$
**end for**
**Step 4:**
**return** $LP = \{LP_1, LP_2, \ldots, LP_k\}$

**Algorithm 2** Frequent Pattern Mining Using Top-Down LP-Tree Traversal

**Require:** Cluster-wise LP Trees $LP$, Minimum Support Threshold $MinSup$
**Ensure:** Global Frequent Pattern Set $FP$

**Terminology:**
$FP_i$: Frequent patterns generated from cluster $C_i$
$FP$: Global frequent pattern set
**Step 1:**
Initialize $FP \leftarrow \varnothing$
**Step 2:**
**for** each LP Tree $LP_i$ **do**
    traverse $LP_i$ using Top-Down Mining
    generate candidate itemsets
    **for** each candidate itemset $X$ **do**
        **if** $Support(X) \geq MinSup$ **then**
            insert $X$ into $FP_i$
        **end if**
    **end for**
**end for**
**Step 3:**
Merge all cluster-wise frequent patterns:
$FP = FP_1 \cup FP_2 \cup \cdots \cup FP_k$
**Step 4:**
**return** $FP$

**Algorithm 3** Global Support Computation and Closed Frequent Pattern Mining

**Require:** Frequent Pattern Set $FP$
**Ensure:** Closed Frequent Pattern Set $CFP$

**Terminology:**
$GS(X)$: Global Support of pattern $X$
$CFP$: Closed Frequent Pattern Set
**Step 1:**
**for** each frequent pattern $X \in FP$ **do**
    Compute global support:
    $GS(X) = \sum Support_{C_i}(X)$
**end for**
**Step 2:**
Initialize $CFP \leftarrow \varnothing$
**Step 3:**
**for** each frequent pattern $X \in FP$ **do**
    **if** $\nexists X'$ such that $X \subset X'$ and $GS(X) = GS(X')$ **then**
        insert $X$ into $CFP$
    **end if**
**end for**
**Step 4:**
**return** $CFP$

## 5. Results and Discussions

This section presents the study results for the proposed CATD-LPT-CFPM algorithm. The outcomes of the proposed CATD-LPT-CFPM algorithm are compared to the existing FP-close, DFI-growth and DFI list method for mining closed frequent itemsets. The experimental analysis uses different databases with discrete transactions. The criteria are measured using various minimal support values. The experiment's result shows that the proposed CATD-LPT-CFPM algorithm mines the closed frequent patterns effectively. A web application is utilized to implement the proposed CATD-LPT-CFPM algorithm.

### 5.1 Experimental setup

Experimental analysis is used for three different benchmark databases with discrete transactions to compare with the proposed frequent pattern mining algorithms. The public dataset and private data were used for the implementation. Two of the datasets, Chess and Mushroom are obtained from FIMI repository at the University of Antwerp y (http://fimi.uantwerpen.be/data) [28]. The author has generated third dataset Online Shopping which is also available on request.

The chess dataset has four attributes; length, size, category, Number of items, and number of transactions. Product ID, location, latitude, longitude, order date, order ID and Customer ID are the attributes for Online Shopping data. The dataset has 4316 unique items and 724 transactions with average transaction length is 14.00 item. The datasets were pre-processed before implementation, by removing the empty data and standardized the input type and file formats were converted into transactional form suitable for the mining method. 8,124 transactions of the Mushroom dataset are divided into 22 attributes and converted into transactional format into 119 binary items.
Product ID, location, latitude, longitude, order date, order ID and Customer ID are the attributes for Online Shopping data. These data sets have been considered for this implementation since these datasets are frequently used for the research on pattern mining.

The Python language was used to carry out the implementation. The trials were conducted over datasets of various scales with an equal scaling technique based on the common ratio 1:2:3: 4:5. We examined five experimental trials per dataset by using one-fifth, two-fifth, three-fifth, four-fifth, and full transaction set respectively.
This technique has been employed to ensure consistent performance evaluations by comparing across datasets of varying size in an unbiased fashion. We compare the CATD-LPT-CFPM with existing DFI-growth, DFI-list, and FP-close algorithms for mining closed frequent itemset. We measured using varying values for the minimal support criterion.

The results indicate that CATD-LPT-CFPM is efficient and effective at mining closed frequent patterns, while offering competitive performance. Despite a longer execution time compared to existing techniques, the proposed method benefits from a cluster-aware mining approach that streamlines the search process and improves performance. The proposed CATD-LPT-CFPM algorithm was implemented through a web application. Table 3 depicts the comparison of execution time complexity for DFI-growth, DFI-list, FP-close and the CATD-LPT-CFPM algorithm. Table 4 presents the summary statistics for the Online Shopping Dataset.

Table 3: Time complexity

| **Algorithm** | **Time Complexity** |
|---|---|
| CATD-LPT-CFPM | $O(n \times m + C \times P)$ average, $O(2^m)$ worst case |
| FP-Close | $O(n \times m + F)$ average, $O(2^m)$ worst case |
| DFI-Growth | $O(\|FCI\| \times L + FI)$ (average), $O(2^m)$ (worst case) |
| DFI-List | $O(\|FCI\| \times L + FI)$ (average case), $O(2^m)$ (worst case) |

Table 4: Summary statistics of Online Shopping Dataset

| Attribute | Mean | Median | Standard Deviation | Minimum | Maximum |
|---|---|---|---|---|---|
| Sales | 2361.83 | 2063.58 | 1458.13 | 553.44 | 17499.95 |
| Shipping Cost | 25126.27 | 24454.50 | 12715.89 | 58.00 | 51284.00 |

### 5.2 Memory Consumption Analysis

Three benchmark datasets were utilized to compare the new CATD-LPT-CFPM algorithms to the existing FP-close, DFI-growth and DFI list techniques. With the different minimal support values represented on the x-axis and runtime in seconds represented on the y-axis.

A distinct minimum support threshold is defined for every database to examine the utilization of memory for a minimal closed frequent itemset. The memory used in MB for the proposed CATD-LPT-CFPM technique compared to the traditional FP-close, DFI-graph and DFI-list algorithm is shown on the perpendicular axis. The variance in memory utilization across various minimum support threshold values is displayed on the horizontal axis. As the database's minimum support grows, the proposed CATD-LPT-CFPM method uses a substantial amount of memory, as seen in the graph Fig. 8 to 10.

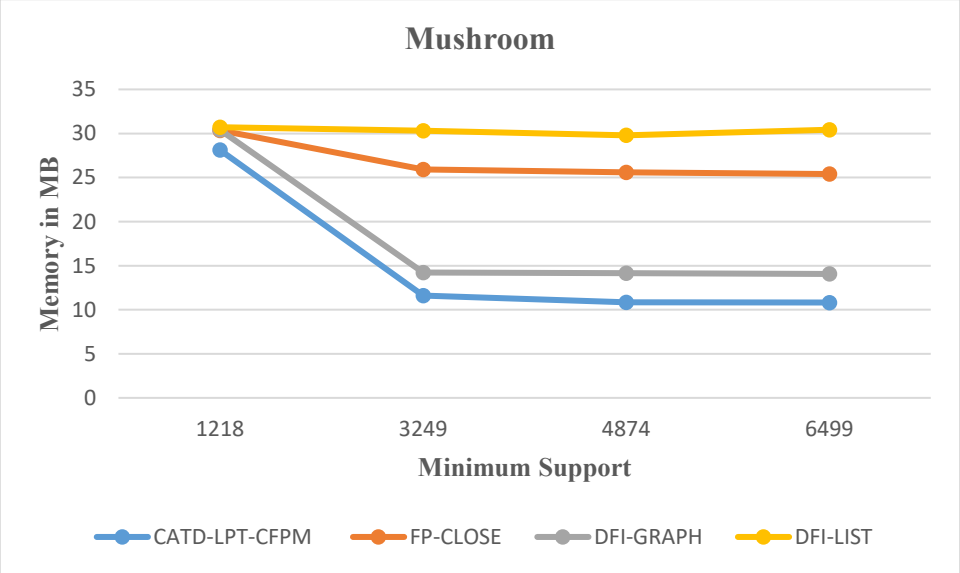



Fig.8 Memory usage comparison between the proposed CATD-LPT-CFPM and existing algorithms for Mushroom dataset

Table 5: Memory usage (MB): mean ± standard deviation of five separate runs for the Mushroom database

| MinSup | CATD-LPT (MB) | FP-CLOSE (MB) | DFI-GRAPH (MB) | DFI-LIST (MB) |
|---|---|---|---|---|
| 1218 | 28.12 ± 0.21 | 30.36 ± 0.24 | 30.41 ± 0.23 | 30.71 ± 0.25 |
| 3249 | 11.62 ± 0.10 | 25.91 ± 0.20 | 14.24 ± 0.12 | 30.31 ± 0.24 |
| 4874 | 10.86 ± 0.09 | 25.58 ± 0.19 | 14.15 ± 0.11 | 29.79 ± 0.23 |
| 6499 | 10.82 ± 0.09 | 25.40 ± 0.18 | 14.06 ± 0.11 | 30.44 ± 0.24 |

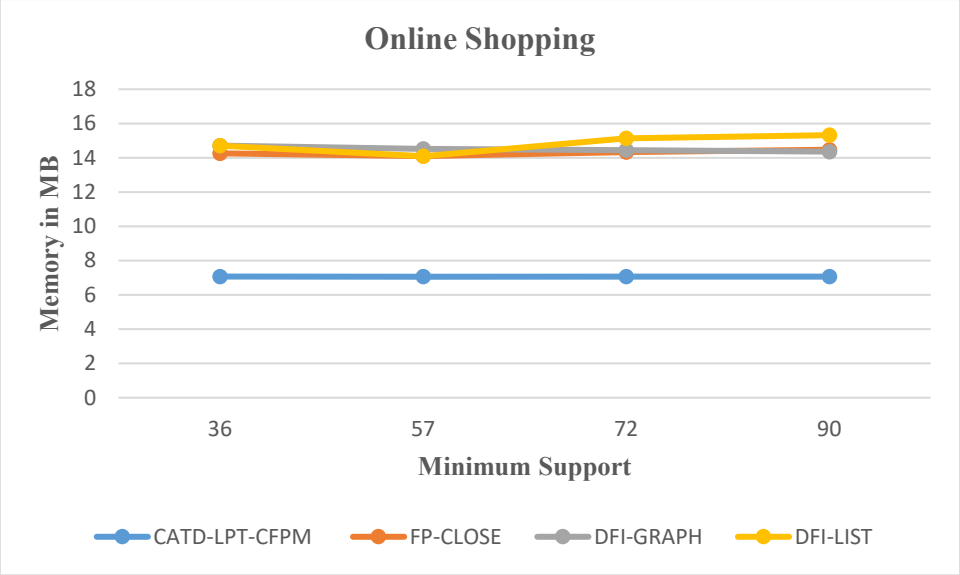



Fig.9 Memory usage comparison between the proposed CATD-LPT-CFPM and existing algorithms for Online shopping dataset

Table 6: Memory usage (MB): mean ± standard deviation of five separate runs for the online shopping dataset

| MinSup | CATD-LPT (MB) | FP-CLOSE (MB) | DFI-GRAPH (MB) | DFI-LIST (MB) |
|---|---|---|---|---|
| 36 | 7.07 ± 0.06 | 14.26 ± 0.11 | 14.71 ± 0.12 | 14.70 ± 0.11 |
| 57 | 7.06 ± 0.05 | 14.10 ± 0.10 | 14.51 ± 0.11 | 14.09 ± 0.10 |
| 72 | 7.06 ± 0.05 | 14.34 ± 0.12 | 14.45 ± 0.11 | 15.14 ± 0.13 |
| 90 | 7.06 ± 0.05 | 14.47 ± 0.12 | 14.36 ± 0.11 | 15.32 ± 0.14 |

Figure 8 graphically compares the efficacy of the proposed CATD-LPT-CFPM with the existing approaches. The proposed CATD-LPT-CFPM algorithm consumes 28.12 MB, FP-close consumes 30.36 MB, DFI-graph consumes 30.41MB and DFI-lists consumes 30.71for Mushroom dataset with a minimum support level of 1218. When improving the minimal support threshold value 3249, the proposed CATD-LPT-CFPM algorithm consumes 11.62 MB, FP-close consumes 25.91MB, DFI-graph consumes 14.24MB and DFI-list consumes 30.31MB to identify closed frequent itemsets, demonstrating that the existing technique is not as effective as the CATD-LPT-CFPM algorithm. The average memory usage and its standard deviation are shown in table 5.

The graph in figure 9 illustrates that the proposed CATD-LPT-CFPM algorithm consumes 7.05 MB of memory, whereas FP-close consumes 14.10MB, DFI-graph consumes 14.51MB and DFI-list consumes 14.09MB of memory for the threshold value of 57 for the dataset online shopping. The graph unequivocally demonstrates that the proposed CATD-LPT-CFPM method outperforms the existing algorithms in most circumstances. Furthermore, as compared to other algorithms, CATD-LPT-CFPM algorithm consumes memory in a highly constant and steady manner independent of threshold. The average memory usage and its standard deviation are shown in table 6.

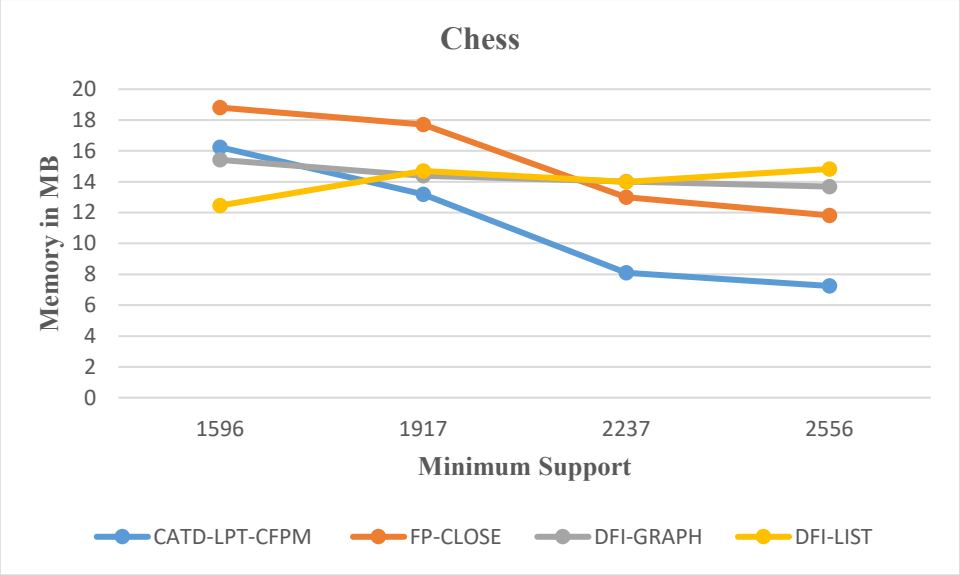



Fig. 10 Memory usage comparison between the proposed CATD-LPT-CFPM and existing algorithms for chess dataset

Table 7: Memory usage (MB): mean ± standard deviation of five separate runs for the chess dataset

| MinSup | CATD-LPT (MB) | FP-CLOSE (MB) | DFI-GRAPH (MB) | DFI-LIST (MB) |
|---|---|---|---|---|
| 1596 | 16.24 ± 0.13 | 18.81 ± 0.15 | 15.42 ± 0.12 | 12.46 ± 0.10 |
| 1917 | 13.18 ± 0.11 | 17.71 ± 0.14 | 14.39 ± 0.11 | 14.70 ± 0.12 |
| 2237 | 8.09 ± 0.08 | 13.00 ± 0.11 | 14.02 ± 0.11 | 13.99 ± 0.11 |
| 2556 | 7.25 ± 0.07 | 11.82 ± 0.10 | 13.69 ± 0.10 | 14.83 ± 0.12 |

Figure 10 shows the comparative memory usage results for the Chess database. According to the figure, the memory utilization of CATD-LPT-CFPM algorithm, FP-close, DFI-graph and DFI-list is 8.09MB and 13MB, 14.02MB and 13.99MB respectively, with a minimum level of support required to 2237. When the minimum level of support lowers, the existing method uses virtually the whole data size to perform its job. When the minimal level of support is increased to 2556, the amount of memory required to store closed frequent itemsets decreases. Thus, the existing FP-close, DFI-graph and DFI-list consumes 11.82MB, 13.69MB and 14.83MB of memory, whereas the proposed CATD-LPT-CFPM algorithm takes up 7.24MB. The average memory usage and its standard deviation are shown in table 7.

### 5.3 Execution Time Analysis

Three benchmark datasets were utilized to compare the proposed CATD-LPT-CFPM algorithms to the existing FP-close, DFI-graph and DFI-list techniques. With the different minimal support values represented on the x-axis and runtime in seconds represented on the y-axis. Based on the empirical data displayed in Figure 11, the existing DFI-graph technique takes 0.28 seconds to execute for a minimal support criterion value of 6499. Following it FP-close and DFI-list also takes lower time to mine the closed patterns. The proposed CATD-LPT-CFPM takes 20.23 seconds to mine the patterns. It is due to the steps included in exploration of clusters, construction of top-down linear prefix tree and closedness checking. Thus, rests in computation overhead in the process of mining. However, they identify the closed frequent patterns effectively and conduct organized search space investigation. Table 8 shows the average time and its standard deviation of mushroom dataset.

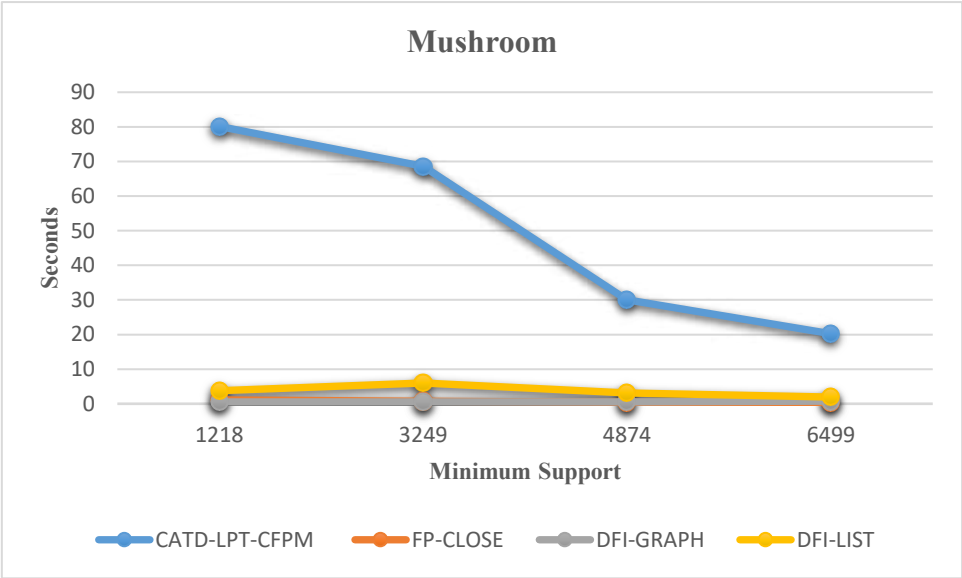



Fig.11 Execution time of proposed CATD-LPT-CFPM versus existing algorithms on mushroom dataset

Table 8: Execution time (seconds): mean ± standard deviation of five separate runs for the Mushroom database

| MinSup | CATD-LPT (s) | FP-CLOSE (s) | DFI-GRAPH (s) | DFI-LIST (s) |
|---|---|---|---|---|
| 1218 | 80.56 ± 0.73 | 1.13 ± 0.03 | 0.54 ± 0.02 | 3.79 ± 0.07 |
| 3249 | 68.41 ± 0.61 | 0.69 ± 0.02 | 0.57 ± 0.01 | 6.01 ± 0.09 |
| 4874 | 30.11 ± 0.39 | 0.47 ± 0.01 | 0.56 ± 0.01 | 3.16 ± 0.05 |
| 6499 | 20.28 ± 0.27 | 0.29 ± 0.01 | 0.76 ± 0.02 | 1.94 ± 0.04 |

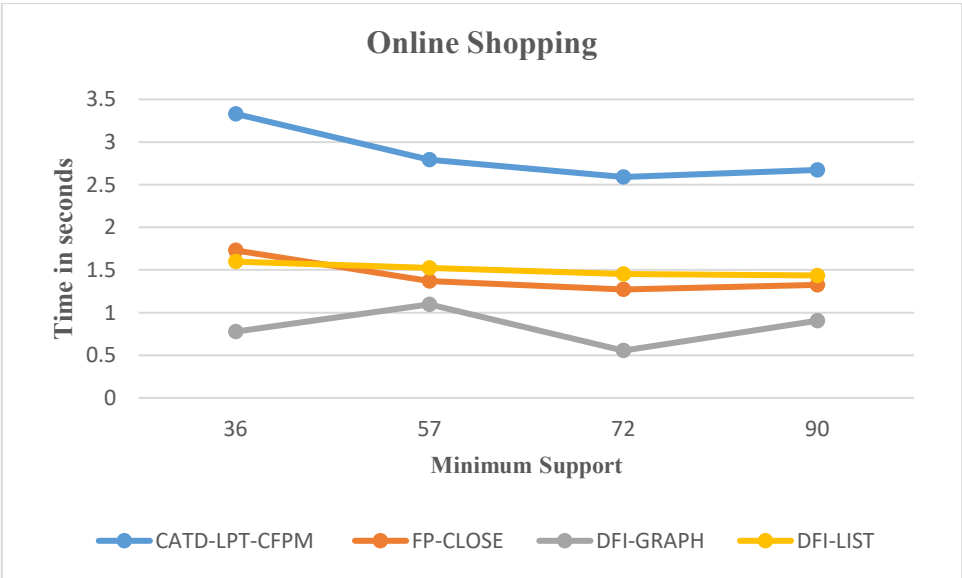



Fig.12 Comparison of execution time of implemented algorithms on online shopping dataset and CATD-LPT-CFPM.

From figure 12 it is shown that the proposed CATD-LPT-CFPM takes more time than the existing algorithms for online shopping dataset. It is due to the additional steps involved in the construction of clustering, pruning and closedness verification. Even though, the execution time is in acceptable range for the given dataset. Table 9 shows the average time and its standard deviation for online shopping dataset.

Table 9: Execution time (seconds): mean ± standard deviation of five separate runs for the online shopping dataset

| MinSup | CATD-LPT (s) | FP-CLOSE (s) | DFI-GRAPH (s) | DFI-LIST (s) |
|---|---|---|---|---|
| 36 | 3.33 ± 0.05 | 1.73 ± 0.03 | 0.776 ± 0.012 | 1.60 ± 0.03 |
| 57 | 2.79 ± 0.04 | 1.37 ± 0.02 | 1.097 ± 0.015 | 1.52 ± 0.02 |
| 72 | 2.59 ± 0.03 | 1.27 ± 0.02 | 0.555 ± 0.010 | 1.45 ± 0.02 |
| 90 | 2.67 ± 0.04 | 1.33 ± 0.02 | 0.904 ± 0.013 | 1.44 ± 0.02 |

Comparison of proposed CATD-LPT-CFPM and the existing algorithms are displayed in figure 13 for chess dataset. The proposed CATD-LPT-CFPM takes 73.72 seconds which is greater than the existing algorithms. This is because of the clustering phase, top-down prefix tree construction and closedness verification. Clustering needs to divide the transactional database into useful groups. Construction of top-down linear tree includes additional computational overhead. Closedness process involves support comparisons, subset and superset checking to validate the closed frequent itemset. Thus, it results in increased execution time. Table 10 shows the average time and its standard deviation for chess dataset.

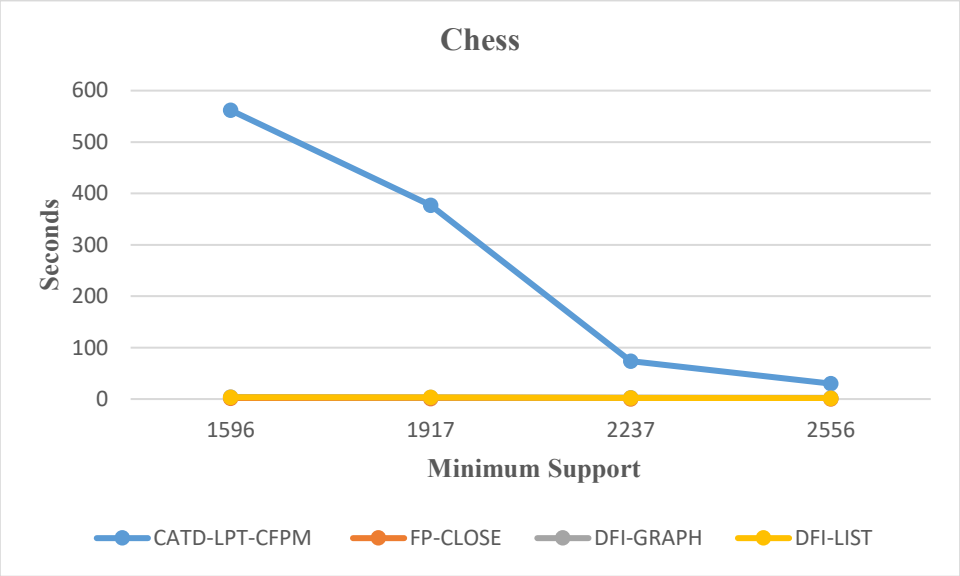



Fig.13 Execution time comparison of proposed CATD-LPT-CFPM and existing algorithms for chess dataset

Table 10: Execution time (seconds): mean ± standard deviation of five separate runs for the chess dataset

| MinSup | CATD-LPT (s) | FP-CLOSE (s) | DFI-GRAPH (s) | DFI-LIST (s) |
|---|---|---|---|---|
| 1596 | 561.79 ± 4.86 | 1.77 ± 0.03 | 4.08 ± 0.06 | 3.14 ± 0.05 |
| 1917 | 376.88 ± 3.24 | 1.01 ± 0.02 | 3.43 ± 0.05 | 3.18 ± 0.04 |
| 2237 | 73.72 ± 0.82 | 0.32 ± 0.01 | 2.85 ± 0.04 | 2.40 ± 0.03 |
| 2556 | 30.05 ± 0.39 | 0.28 ± 0.01 | 2.34 ± 0.03 | 1.76 ± 0.02 |

The following factors account for the proposed CATD-LPT-CFPM algorithm's enhanced performance:

i. The proposed CATD-LPT-CFPM adopts clustering for partitioning the transactional database into clusters, facilitating local mining and decreasing the needless pattern generation

ii. The proposed CATD-LPT-CFPM adopts linear prefix tree to organize the transactions efficiently, to speed up the tree traversal and to decrease the number of database scans.

iii. The approach minimizes the amount of candidate patterns that must be investigated since the proposed CATD-LPT-CFPM only considers the associated transaction clusters and uses pruning strategies.

iv. The proposed CATD-LPT-CFPM adopts global support checking to precisely detect the closed frequent patterns and prune redundant patterns while maintaining support information.

v. Exploration of closed frequent patterns provides a simplified representation of the frequent pattern space that enhances the pattern interpretability and efficiency in storage.

vi. The approach is ideal for big transactional datasets because of their clustering and tree-based structure, which effectively manage memory consumption.

### 5.4 Frequent Pattern Analysis

Creating frequent patterns that meet the minimum support criterion is the goal of frequent pattern mining. While a lower minimum support criterion leads to an increase in common patterns, a higher minimum support threshold efficiently prunes infrequent patterns. Count of frequent pattern generation is shown in figure 14 for mushroom dataset with different levels of minimum support. The Y-axis displays the quantity of often studied frequent patterns, while the X-axis displays the minimum support level. When the minimum support threshold is 1218 the number of patterns explored is 578 and while reducing to 6499, the number of patterns explored is reduced to 14 as only strong frequent patterns are generated.

The online shopping database has a minimum support of 36, with the most examined itemset being 28. By lowering the minimal support level significantly to 72, it results in 11 frequent itemsets. When the minimal support increases, the exploration of frequent itemsets is limited as shown the figure 15.

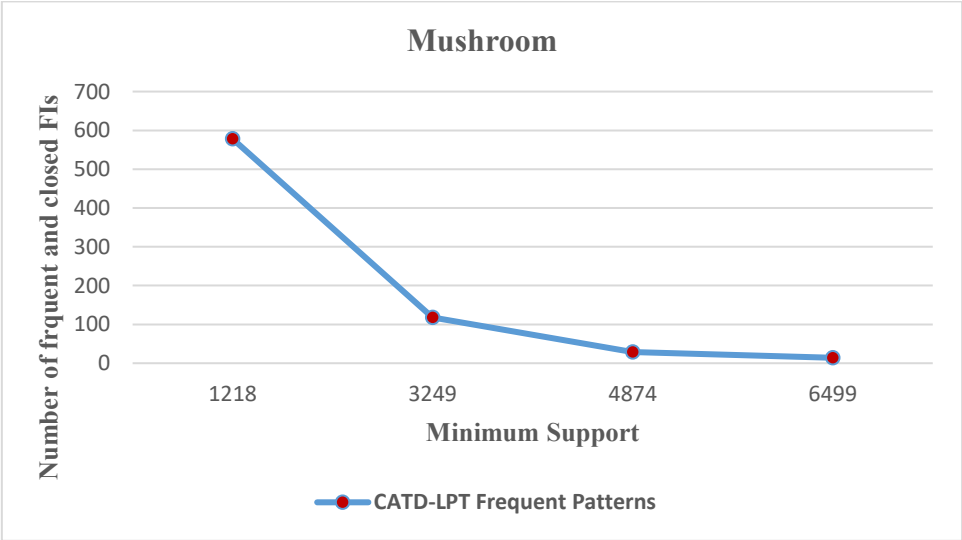



Fig.14 Number of frequent patterns generated for mushroom dataset

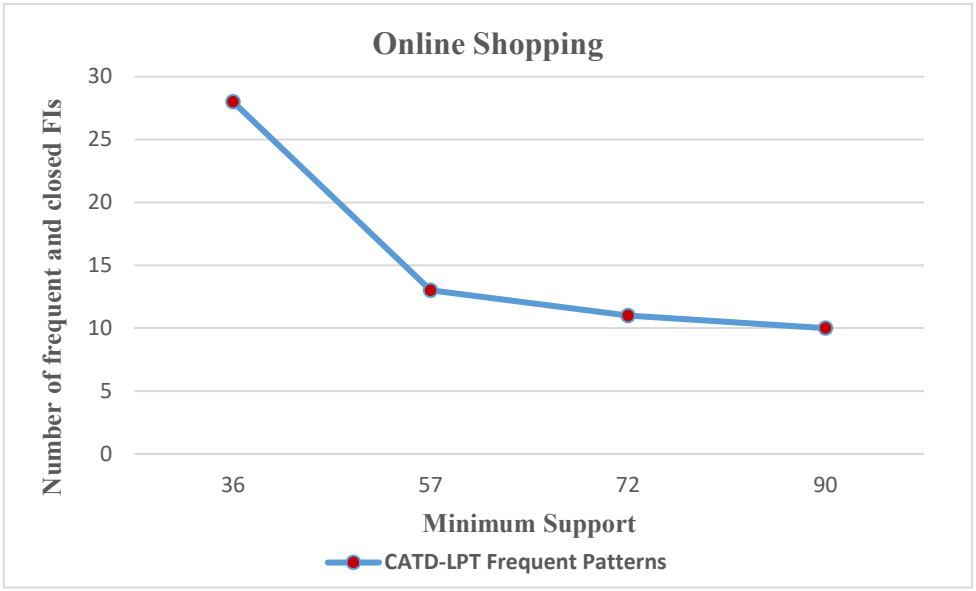



Fig.15 Number of frequent patterns generated for online shopping dataset

Figure 16 demonstrates the exploration of frequent patterns for chess datasets. Huge number of frequent patterns of 397 is generated for the minimum support threshold 1598 as chess is dense dataset in nature. Lowering the minimum support threshold to 2237 resulted in more search space that resulted in 152 frequent patterns by effective pruning. Overall, decreases in minimum support threshold result in more frequent pattern generation and increase in minimum support threshold result in lower frequent pattern generation.

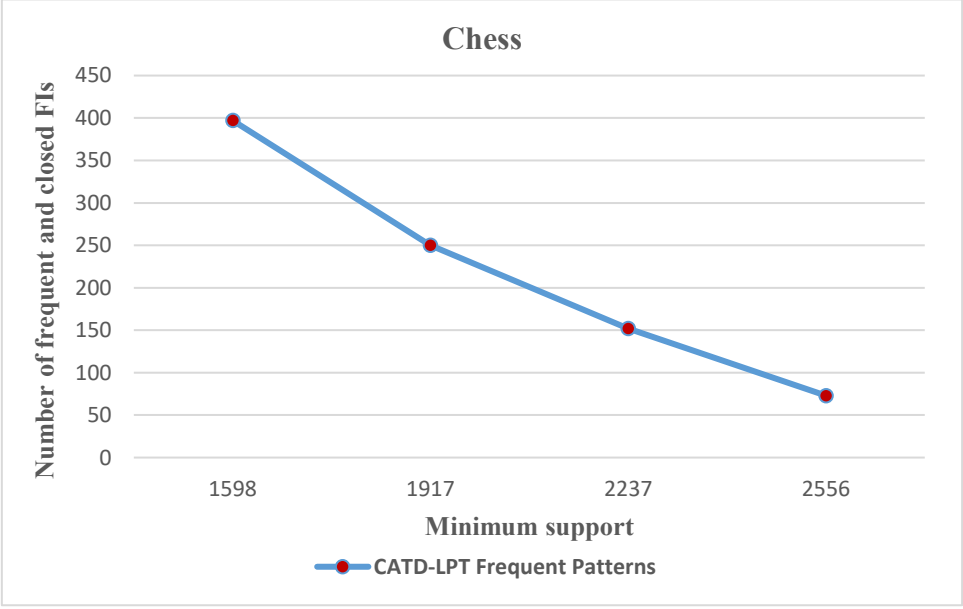



Fig.16 Number of frequent patterns generated for chess dataset

### 5.5 Closed Pattern Analysis

Closed frequent pattern identifies the non-redundant frequent patterns. A frequent pattern that has no proper superset with same support is known as closed frequent patterns. By pruning the redundant patterns, closed frequent pattern mining provides a compact representation of the frequent patterns space by removing redundant patterns.

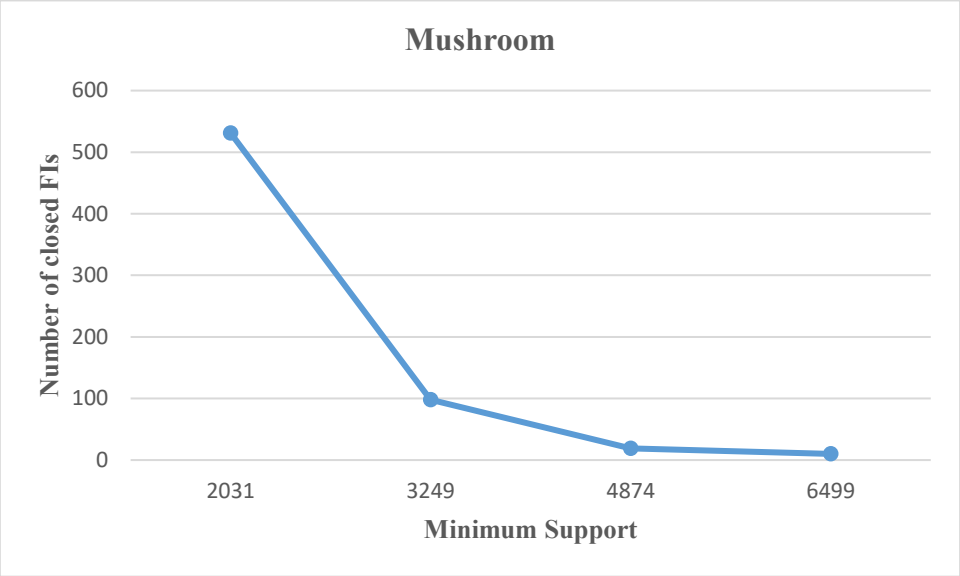



Fig.17 Number of closed frequent patterns generated for mushroom dataset

Figures 17–19 show the exploration of a closed frequent itemset using the CATD-LPT-CFPM method. The total quantity of closed frequent item sets examined is displayed on the vertical axis. The horizontal axis shows how the closed frequent itemset changes for different minimum support threshold values.

Figure 17 demonstrates the exploration of a closed frequent itemset for the dataset Mushroom. The closed pattern examined for the database Mushroom is 531. Despite lowering the minimal support level significantly to 4874 closed frequent itemset mines 19 patterns. Closed frequent itemsets are four times as tiny as frequent itemsets as proposed adopts the closedness checking.




Fig.18 Number of closed frequent patterns generated for online shopping dataset

The graph in Fig. 18 shows that when the minimal support increases, the exploration of closed frequent itemsets for the dataset online shopping decreases abruptly. The chart shows that the majority of closed frequent itemsets investigated are 28 for a minimum support threshold of 36.

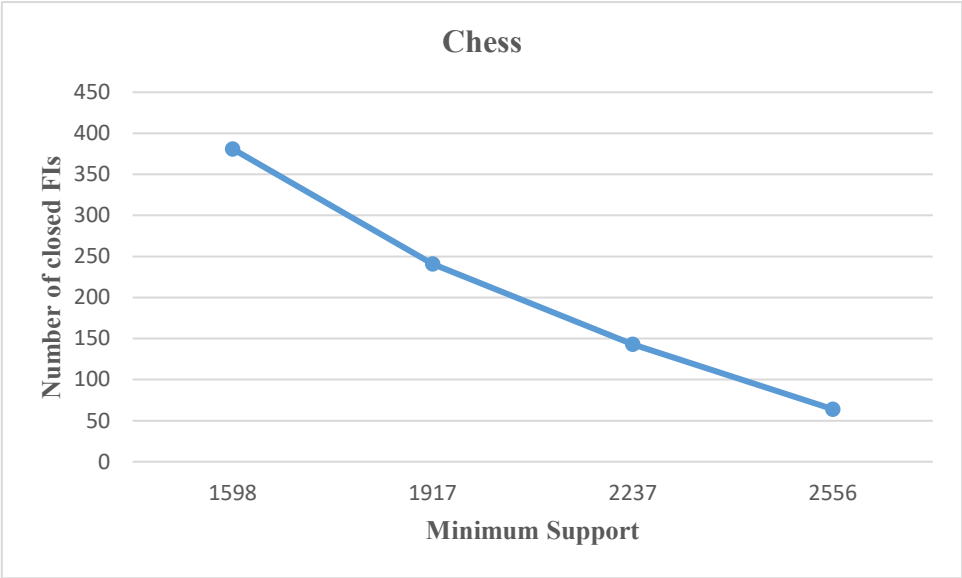


Fig.19 Number of closed frequent patterns generated for chess dataset

The graph in Fig. 19 illustrates that the volume of closed frequent item sets investigated for the dataset chess drops to 73 when the minimum support reaches 2556. The proposed CATD-LPT-CFPM approaches are used to remove unnecessary itemsets. There are 250 closed frequent itemsets that are examined with the lowest possible support of 1917. It has been found that the number of closed frequent item sets grows with declining support.

Overall, the closed frequent patterns reduce as minimum support threshold increases. Redundant frequent patterns are eliminated by adopting global checkness verification resulting in a structured and informative pattern set.

## 6. Conclusion and Future work

The closed frequent itemset mining technique is considered a viable solution for large datasets. In this research article, the proposed framework CATD-LPT-CFPM is presented. The proposed framework includes, clustering of transactions, Efficient construction of prefix tree and global closedness verification of explore closed frequent patterns that reduces the redundancy in the explored patterns. Experimental evaluations on three datasets of Mushroom, Online Shopping and Chess show the effectiveness of CATD-LPT-CFPM in creating structured and informative closed frequent patterns by using different minimum support thresholds. It is observed from the result that when the minimum support threshold increases, the number of frequent and closed frequent pattern decreases that reduces the search space and computational needs. Due to the additional processing steps involved in clustering, construction of tree and global closedness verification the proposed CATD-LPT-CFPM takes more time than the existing algorithms. Even though, it gives a compact mining framework to prune the redundant patterns effectively and improves the consumption of memory. From the experimentation it is observed that CATD-LPT-CFPM mines the closed frequent patterns effectively by managing the quality of pattern and compactness. As a result, CATD-LPT-CFPM is a promising method for closed frequent itemset mining, especially in situations where effective knowledge extraction, memory efficiency, and pattern reduction are more important than execution speed alone. For future research, the execution time can be improved by focusing on parallel processing and improved pruning strategies.

## Statements and Declarations

***Authors' contributions:*** Equal contribution from all authors

***Availability of data and materials****:* Both private and publicly accessible datasets were used in the implementation. The University of Antwerp's FIMI repository (https://fimi.uantwerpen.be/data/) provided two benchmark datasets, Chess and Mushroom. The author created a third dataset, Online Shopping, as part of their research, which is available upon reasonable request.

***Competing interests:*** The authors declare no financial or personal interests that could have influenced the work presented in this paper.

***Acknowledgements:*** not applicable

***Funding:*** This work is funded by Vellore Institute of Technology – Vellore.

**References**

1. Agrawal R, Imieliński T, Swami A: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
2. Zaki MJ: Scalable algorithms for association mining. IEEE Trans Knowl Data Eng 12(3):372–390 (2000)
3. Wael Z, Yasser K, Elham E, Fayed G: Mining association rule with reducing candidates generation. In: Proceedings of Fourth International Conference on Intelligent Computing and Information Systems (ICICIS2009). ACM (2009)
4. Xun Y, Cui X, Zhang J, Yin Q: Incremental frequent itemsets mining based on frequent pattern tree and multi-scale. Expert Syst Appl 163:113805 (2021)
5. Cheung DW, Han J, Ng VT, Wong C: Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 106–114 5. (1996)
6. Cheung DW, Lee SD, Kao B: A general incremental technique for maintaining discovered association rules. Database Syst Adva Appl 97:185–194 (1997)
7. Zhou Z, Ezeife C: A low-scan incremental association rule maintenance method based on the apriori property. In: Conference of the Canadian Society for Computational Studies of Intelligence, Springer, pp. 26–35 (2001)
8. Han J, Pei J, Yin Y, Mao R: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min Knowl Disc 8(1):53–87 10. (2004)
9. Hong TP, Lin CW, Wu YL: Incrementally fast updated frequent pattern trees. Expert Syst Appl 34(4):2424–2435 (2008)
10. Lv D, Fu B, Sun X, Qiu H, Liu X, Zhang Y: Efficient fast updated frequent pattern tree algorithm and its parallel implementation. In: 2017 2nd International Conference on Image, Vision and Computing (ICIVC), IEEE, pp. 970–974. (2017)
11. Sun J, Xun Y, Zhang J, Li J: Incremental frequent itemsets mining with FCFP tree. IEEE Access 7:136511–136524. (2019)
12. Magdy, M., Ghaleb, F.F.M., Mohamed, D.A.E.A. *et al:* CC-IFIM: an efficient approach for incremental frequent itemset mining based on closed candidates. *J Supercomput* 79, 7877–7899 (2023).
13. Cheung DW, Han J, Ng VT, Wong C: Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 106–114. (1996)
14. M.J. Zaki, K. Gouda: Fast vertical mining using diffsets, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 326–335. , 2003
15. B. Vo, T. Le, F. Coenen, T.P. Hong: Mining frequent itemsets using the N-list and subsume concepts, Int. J. Mach. Learn. Cybern. 7 (2) 253–265. (2016)
16. N. Aryabarzan, B. Minaei-Bidgoli, M: Teshnehlab, negFIN: An efficient algorithm for fast mining frequent itemsets, Expert Syst. Appl. 105 129–143. (2018)
17. N. Aryabarzan, B. Minaei-Bidgoli, NEclatClosed: A vertical algorithm for mining frequent closed itemsets, Expert Syst. Appl. 174 114738. (2021)
18. Sinthuja, M., Diviya, M. & Saranya, P. CLTD-LP: an optimized top-down clustering approach with linear prefix trees for scalable frequent pattern discovery in large datasets. Sci Rep 16, 9918 (2026). https://doi.org/10.1038/s41598-026-37338-9
19. Wu, C. W., Huang, J. T., Lin, Y. W., Chuang, C. Y., & Tseng, Y.-C. (2021). Efficient algorithms for deriving complete frequent itemsets from frequent closed itemsets. Applied Intelligence, 52(6), 7002-7023.
20. Advance online publication. https://doi.org/10.1007/s10489-020-02172-7B. Huynh, B. Vo, V: Snasel, An efficient parallel method for mining frequent closed sequential patterns, in: IEEE Access, Vol. 5, , pp. 17392–17402. 2017
21. K. Gouda, M.J. Zaki, GenMax: An efficient algorithm for mining maximal frequent itemsets, Data Min. Knowl. Discov. 11 (3) 223–242. (2005)
22. S. Pham, T. Le, Z. Deng, B. Vo: A novel approach for mining maximal frequent patterns, Expert Syst. Appl. 73 178–186. (2017)
23. B. Huynh, B. Vo, V. Snasel: An efficient parallel method for mining frequent closed sequential patterns, in: IEEE Access, Vol. 5, 20(2017).
24. P. Fournier-Viger, J.C. Lin, R.U. Kiran, Y.S. Koh, R. Thomas: A survey of sequential pattern mining, Data Sci. Pattern Recognit. (DSPR) 1 (1) 54–77. (2017)
25. B. Huynh, C. Trinh, H. Huynh, T. Van, B. Vo, V: Snasel, An efficient approach for mining sequential patterns using multiple threads on very large databases, Eng. Appl. Artif. Intell. 242–251 74 (2018)

26. Wan, X., Han, X: Efficient Top-k Frequent Itemset Mining on Massive Data. Data Sci. Eng. 9, 177–203 (2024). https://doi.org/10.1007/s41019-024-00241-2
27. T.D.D. Nguyen, L.T.T. Nguyen, L. Vu, B. Vo, W. Pedrycz: Efficient algorithms for mining closed high utility itemsets in dynamic profit databases, Expert Syst. Appl. 186 (2021) 115741.
28. Frequent Itemset Mining Dataset Repository. http://fimi.uantwerpen.be/data/
29. M. Sinthuja , S. Pravinthraja , B.K Dhanalakshmi , Gururaj H L, G Jyothish Lal, An efficient and resilience linear prefix approach for mining maximal frequent itemset using clustering, Journal of Safety Science and Resilience, Volume 6, Issue 1, March 2025, Pages 93-104.