ISSN:0975-9646

Compact Structure of Felonious Crime Sets Using FP-Tree Comparable Algorithms Analysis

Apexa Joshi[#1], Dr. Suresh M. B.[*2]

[#1] *Research Scholar, School of Science, R K University, Rajkot, India*
Assistant Professor, JVIMS-MCA-GTU, Jamnagar, India
[*2] *Professor & Head, Dept. of ISE, East West Institute of Technology, Viswandeedam, Post Bangalore*

***Abstract*— Discovering related offense case subsets is a crucial task for intelligence analysts in criminality investigation. It can not only deliver numerous evidences to resolve crimes but also increase efficiency to catch the lawbreakers. Mining recurrent crime sets from the large felonious database is very critical and essential duty of law enforcement agency. FP – tree similar algorithms are considered as very effective algorithms for effectively mine recurrent crime sets, in comparison to the techniques proposed in the past. These techniques measured as effectual because of their squeezed construction and also for less generation of candidates crime sets compare to Apriori and Apriori similar algorithms. So this paper aims to offerings basic concepts, their capabilities and comparisons of the FP – Tree similar algorithms for mining recurrent crime sets to diminish the criminality from the society.**

***Keywords*— Crime, Data mining, FP-Tree based Algorithm, Frequent Crime sets.**

I. INTRODUCTION

Massive crime data have been accumulated in law enforcement organizations. Millions of cases have been stored in felonious central database of department of public security and this number increases every year. Thus how to effectively utilize the potential information from the crime data is a huge challenge. "Data mining holds the promise of making it easy, convenient, and practical to explore huge databases for societies and users" [2] and a general framework for crime data mining has been proposed in [1].

Frequent Crime set mining is one of the most essential issues of research for association rule mining in data mining research domain. As an association rule mining is defined as the relation between various crime sets. Association rule mining is a part of pattern discovery techniques in knowledge discovery and data mining (KDD). As performance of association rule mining is depends upon the frequent crime sets mining, thus is necessary to mine frequent crime set efficiently.

Finding subsets of similar cases from large crime data is an important task for intelligence analysts in law-enforcement organizations. If such subsets are found and provided to crime investigators, multiple clues can be obtained from different cases. For example, the offender in case A stole bicycles by cracking the lock and in case B the offender stole bicycles around shopping malls. If case A and case B were confirmed to be similar cases, investigators can conclude that the offender often stole bicycles around shopping malls by cracking the lock. We find that when intelligence analysts query databases, they often query some specific attributes, such as location, victim, tools used etc. These attributes play a critical role in finding the similar case subsets because offenders tend to act similarly as they did before, like choosing the same location or using the same tools. In addition, different behavioural attributes play the main part in different cases categories.

For instance, according to trained analysts' experience, burglary offenders tend to focus on ways to break in a house while fraud offenders tend to focus on choosing their targets, either a person or a company. Therefore these different focuses must been well utilized in the process of finding similar cases subsets to reflect the nature of each case category.

The process of extracting association rule mining is divided into two parts:

1. Mine all frequent crime sets pattern each of these patterns should satisfy the minimum support threshold. Once these entire frequent patterns are mined, then
2. Association rules are produced from these frequent crime sets. These association rules essentially fulfil the least support and least confidence. This minimum support and confidence should be designed by the user. Association rule depends upon the generation of the frequent crime.

More database scans, large number of candidate generation, low processing, huge search space and large I/O required, and high memory dependency these are the major problems with apriori similar algorithms which is proposed by many researchers. So in this paper we explore the FP-Growth similar algorithms to decrease I/O dependencies, to reduce memory requirements and for efficiently generation of frequent crime sets. In this paper confine only to the working, properties and comparative performance of FP-Growth similar algorithms.

The paper is separated as follows. In Section 2, we briefly define the problem statement for finding the frequent crime sets from felonious database. Section 3 describes the classic FP-Growth algorithm. A section 4 elaborates the prevailing techniques constructed upon the FP- tree data structure. Section 5 outlines Comparison of algorithms. And finally conclude the paper in section 6.

II. PROBLEM STATEMENT

The problem of mining association rules over criminal investigation analysis finding association between the crimes that are present in the criminal's information from the database.

The database may be from any CBI, FBI, and police department or from any other federal agencies. As defined in [3] the problem is stated as follows: Let C = c1, c2, c3….n is a set of literals, called crimes and n is considered the length of the problem. Let D be a set of FIRs / IR (First Information/investigation Report), where each investigation report F is a set of crimes such that F ∈ C. A unique identifier FID is given to each investigation report. An investigation report F is said to contain X, a set of crimes in C. X ∈ F An association rule is an implication of the form "X⇒ Y", where X ∈ C, Y ∈ C, and X ∩ Y = Ø. A crime set X is said to be large or frequent if its support s is greater or equal than a given minimum support threshold ∂. A crime set X satisfies a constraint CS if and only if CS(X) is true. The rule X ⇒ Y has a support Sp in the investigation report set D if Sp% of the investigation report set in D contain X ⇒ Y. In other words, the support of the rule is the probability that X and Y hold together among all the possible presented cases. It is said that the rule X ⇒ Y holds in the investigation report set D with confidence CF if CF% of investigation report in D that contain X also contain Y .

In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedent X. The problem of discovering all association rules from a set of transactions D consists of generating the rules that have a support and confidence greater than a given threshold. These rules are called Strong Rules. This association-mining task can be broken into two steps:

1. Finding the frequent k-crime set from the large database.
2. Generate the association rule from these frequent crime sets.

In this paper, we focus exclusively on the first step: generating frequent crime sets.

III. FP – GROWTH ALGORITHM

In past numerous algorithms proposed similar to Apriori algorithms. This algorithm is work on anti-monotone property. There are two loop holes to working with these algorithms i.e. repeated database scan and high computational cost, there is need of compact data structure for mining frequent crime sets, which moderates the multi scan problem and improve the candidate crime set generation. To represent node in a frequent pattern using lexicographic tree we utilized an efficient algorithm know as tree projection [2].

FP-Growth algorithm [2] is an efficient algorithm for producing the frequent crime sets without generation of candidate crime sets. This algorithm is work like merge sort techniques is working on divide and conquers approach. It needs a 2 database scan for finding all frequent crime sets. This approach compresses the database of frequent crime sets into frequent pattern tree recursively in the identical order of scale as the figures of frequent patterns, then in next step divide the compressed database into set of conditional databases.

A. Construction of Frequent Pattern Tree

First scan the database and manage the crimes appearing in the investigation report set. Then all the crimes whose support is less than the minimum support which is user defined are considered as infrequent are deleted from consideration. All other remaining crimes are considered as frequent crimes and arrange in the sorted order of their frequency. In header table sorted list is store. All the respective support of the crimes is stored using pointers in the frequent pattern tree. To achieve compact structure we build the frequent pattern tree. The sorted crimes according to frequency in header table are used to build the FP-tree. To build FP-tree requires a whole database scan. when the crime insert in the tree checks if it exist earlier in tree as in same order then increment the counter of support by one which is mentioned along with each crime in the tree separated by comma, otherwise add new node with 1 as a support counter[2,3]. A link is maintained using pointers which same crime and its entry in header table. In header table, pointer points to the first occurrence of each crime.

It uses the tree data structure which stores all frequent elements in a compact form as comparative with Apriori which uses array type structure. Let sample database in table 1, 2 and corresponding the support count in table 3 is:

TABLE 1
TWO CASES WITH CATEGORICAL ATTRIBUTE

Case No.	**000001**	**000002**
Case Category	Burglary	Fraud
Time	2006-3-28	2006-4-3
Location Category	Dwelling house	Hotel
Modus Operandi	attack from window	Temptation of money
Victim/Target	Old woman	Middle-aged man
Motivation	For money	For money
Characteristic	Individual	Gang

TABLE 2
SAMPLE DATABASE

FID	**Crime**
F1	C1, C2, C3, C4, C5
F2	C5, C4, C6, C7, C3
F3	C4, C3, C7, C1, C8
F4	C4, C7, C9, C1, C10
F5	C1, C5, C10, C11, C12
F6	C1, C4, C13, C14, C2
F7	C1, C4, C6, C15, C2
F8	C16, C7, C9, C17, C5
F9	C1, C9, C8, C10, C11
F10	C4, C9, C12, C2, C14
F11	C1, C3, C5, C6, C15
F12	C3, C7, C5, C17, C16
F13	C8, C3, C4, C2, C11
F14	C4, C9, C13, C12, C18
F15	C5, C3, C7, C9, C15
F16	C18, C7, C5, C1, C3
F17	C1, C17, C7, C9, C4
F18	C4, C3, C16, C5, C1

TABLE 3
SUPPORT COUNT FOR EACH CRIME

Crimes	Support	Crimes	Support
C1	11	C10	3
C2	4	C11	3
C3	9	C12	3
C4	10	C13	2
C5	10	C14	2
C6	3	C15	3
C7	8	C16	3
C8	3	C17	3
C9	7	C18	3

Suppose minimum support is 5. Thus delete all infrequent crimes whose support is less them 5.After all the remaining investigation reports arranged in descending order of their frequency. Create a FP - tree. For Each investigation report create a node of an crimes whose support is greater than minimum support, as same node encounter just increment the support count by 1.

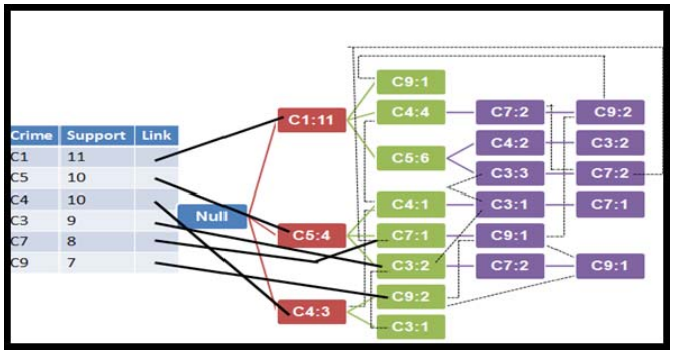

Fig. 1: FP-Tree Constructed For Sample Database

In this way after constructing the FP-Tree one can easily mine the frequent crime sets by constructing the conditional pattern base.

IV. FP – GROWTH VARIATIONS

The above approach is efficient then Apriori algorithm but as the database become large it makes the processing slow, due to large database the FP- tree construction is very large and sometimes does not fit into the main memory. To overcome these limitations there are many optimized techniques listed in [10] on this basic FP-Growth algorithm [8, 9, 11, 12, 15]. In this paper investigate the details of some of the variations of FP-growth namely COFI-tree mining [8], CT-PRO Algorithm [12] and FP growth [2] (as discussed above). Our goal is to take the overview details of each algorithm and discuss the main optimization ideas of each algorithm.

A. COFI-Tree Algorithm

COFI tree generation is depends upon the FP-tree however the only difference is that in COFI tree the links in FP-tree is bidirectional that allow bottom up scanning as well [7,8]. The relatively small tree for each frequent crime in the header table of FP-tree is built known as COFI trees [8]. Then after pruning mine the each small tree independently which minimize the candidacy generation and no need to build he conditional sub-trees recursively. At any time only one COFI tree is present in the main memory thus in this way it overcome the limitations of classic FP-tree which cannot fit into main memory and has memory problem.

COFI tree is based upon the new anti-monotone property called global frequent/local non frequent property [8]. It states that all the nonempty subsets of frequent patterns with respect to the crime X of an X-COFI tree must also be frequent with respect to crime X. In this approach trying to find the entire frequent crime set with respect to the one frequent crime sets. If the Crime set participate in making the COFI tree then it means that crime set is globally frequent but this doesn't mean that crime set is locally frequent with respect to the particular crime.

1) *Algorithm: Create aCOFI-Tree*

1. Take FP-tree as an input with bidirectional link and threshold value.
2. Consider the least frequent crime from the header table let it be X.
3. Compute the frequency that share the path of crime X and remove all non-frequent crimes for the frequent list of crime X.
4. Create COFI tree for X known as X-COFI tree with support-count and participation=0
5. If crimes on Y which is locally frequent with respect to X form a new prefix path of X-COFI tree
6. DO, Set support count= support of X and participation count =0 for all nodes in a path.
7. Else adjust the frequency count and pointers of header list until all the nodes are not visited.
8. Repeat step 2 goes on until all frequent crimes not found.
9. Mine the X-COFI tree.

Support count and the participation count are used to find candidate frequent pattern and stored in the temporary list, which will be more clear after example.

Let the above FP-tree in figure 1 is the input for making COFI tree. Consider the links between the nodes are bidirectional. Then according to algorithm the COFI tree forms are first consider the crime set C9 as is least frequent crime set, when scan the FP-tree the first branch is (C9,C1) has frequency 1, therefore frequency of the branch is frequency of the test crime, which is C9. Now count the frequency of each crime in a path with respect to C9. It is found that (C7, C3, C4, C5, C1) occur 4, 2, 4, 2, 3 times respectively thus according to anti-monotone property C9 will never appear in any frequent pattern expect itself. In the same way find the global frequent crimes for (C7, C3, C4,C5) which are also locally frequent, like for C7 two crimes C3 and C5 globally frequent crime are also found locally frequent with frequency 5and 6 respectively thus a branch is created for each such crimes with parent C7. If multiple crimes share same prefix they are merged into one branch and counter is adjusted. COFI trees for crimes are follows:

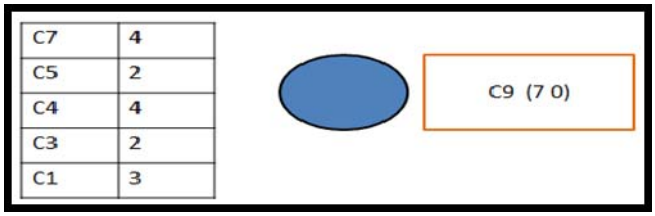

Fig. 2: C9 COFI – Tree

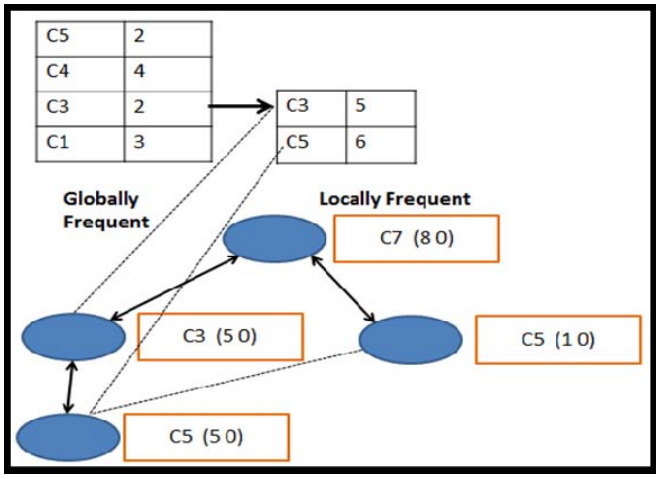

Fig. 3: C7 COFI – Tree

Similarly COFI tree is built for C3, C4, and C5. First after the globally frequent, find the crime sets which are also locally frequent , than find the support counter and make participation counter always equal to '0'. We are representing only for C7 in above example.

Once the COFI trees have built then we have to mine the frequent crimes from these COFI trees with the help of following procedure:

2) Algorithm: MINE X-COFI Tree

1. For node X select the crime from the most frequent to least frequent with its chain
2. Until there are no node left, select all nodes from node X to root save in D list and in list F save the frequency count and participation count.
3. Generate all non-discarded patterns Z from crimes D.
4. Add the list with frequency =F whose patterns not exist in X candidate list, else increment the frequency by F.
5. Increment the participation value by F for all crimes in D.
6. Select the next node and repeat step 2 until there is no node left.
7. Remove all non-frequent crimes from X-COFI tree.

The COFI trees of all frequent crimes are mined independently one by one [8], first tree is discarded before the next COFI tree is come into picture for mining. Based on the support count and participation count frequent patterns are identified and non-frequent crimes are discarded in the end of processing.

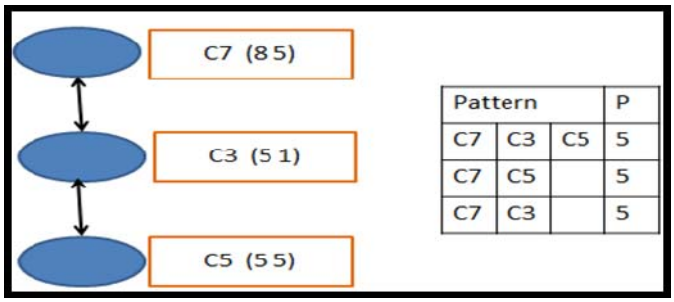

Figure 4: Mining E COFI tree for branch (C7, C3, CI5)

Let's take an example for C7 COFI-Tree as mining process start from the most local frequent crime C5 and as from the figure 3, I5 exist in two branches C5, C3, C7 and C5, C7. Therefore

As the frequency for each branch is equal to frequency of first crime minus participation count of that node, Thus C5 has frequency 5 and participation count is 0 therefore first frequent set is found i.e. (C7,C3, C5:5).

Participation value is incremented by 1 for branch (C7, C3, C5) increment by 5 same as the frequency of C5. the pattern C7,C5 generates a pattern 1 which is already exist therefore increment the previous participation for C7, C5 by 1.therefore for C7 is become 6 and for C5 it become 1 for branch C7,C5.

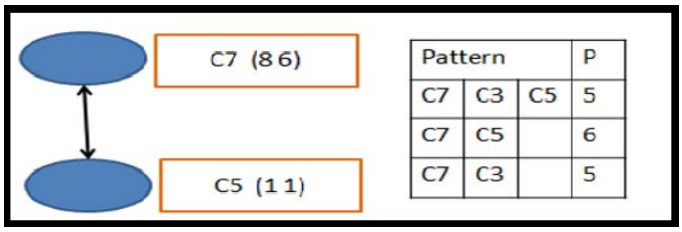

Figure 5: Mining C7 COFI tree for branch (C7, C5)

Similarly mine for sub branch (C7, C3) and it is found that frequent patterns (C7, C3: 5), (C7, C5: 6), (C7, C3, C5: for COFI tree C7. Similarly main the frequent pattern for C3, C4, and C5 crime sets. In this way COFI tree mine the frequent crime sets very easily then the FP-growth algorithm with the help of FP-tree[8]. It saves to memory space as well as time in comparison to the FP-growth algorithm. It mines the large transactional database with minimal usage of memory. It does not produce any conditional pattern base. Only simple traversal is needed in the mining process to find all the frequent crime sets. It is based upon the locally and globally frequent crime sets thus easily remove the frequent crime sets in the early stages and don't allow any locally non frequent elements to takes part in next stage.

B. CT-PRO Algorithm

CT-PRO is also the variation of classic FP-tree algorithm [12]. It is based upon the compact tree structure [9, 11, 14]. It traverses the tree in bottom up fashion. It is based upon the non-recursive based technique [12]. Compress tree structure is also the prefix tree in which all the crimes are

stored in the descending order of the frequency with the field index, frequency, pointer, crime-id [11]. In this all the crimes if the databases after finding the frequency of crimes and crimes whose frequency is greater than minimum support are mapped into the index forms according to the occurrence of crimes in the transaction. Root of the tree is always at index '0' with maximum frequency elements. The CT-PRO uses the compact data structure known as CFP-tree i e. compact frequent pattern tree so that all the crimes of the transactions can be represented in the main memory [6, 12, 13].

The CT-PRO algorithm consists following basic steps [12]:

1. In the first step all the elements from the transaction are found whose frequency is greater than the minimum user defined support.
2. Mapping the elements according to the index value.
3. Construct the CFP-tree which is known as globally CFP-tree.
4. Mine the Global CFP-tree by making local CFP-tree for each particular index.

In this way by following the above steps can easily find the frequent crime sets. The frequency of crime sets greater then minimum support which defined as '5' for the sample database present in Table 1. After the mapping the Global Tree formed from the transactions are follows:

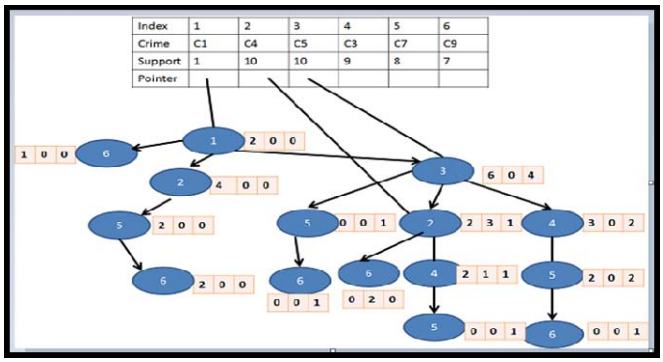

Figure 6: CFP- Tree for table 1

Note: No transaction starts from Crime C3, C7, and C9 therefore no pointer is there.

1) Steps for Making Global CFP - Tree

1. Take Database and threshold value as input.
2. Find the frequent crimes from database and sort in descending order in new list.
3. Then map the frequent crimes of the investigation report in the index form, and sort in ascending order of their FIR Id (Fid).
4. Make maximum frequency crime is the root node of the tree and makes it for index 1; insert all sub children in the tree.
5. For new index starting crimes, adjust pointer and build sub trees and give incremented index value or level as in above figure 6.
6. CFP-Tree has been built. Mine the CFP-tree index wise as projections.

After the global CFP-tree, the local CFP-Tree is build for each index crime separately. It starts from the least frequency element.

The local CFP- Tree for the index 5 i.e. for the crime C7 is found by first counting the other indexes that occur with the index 5. The other indexes are: 1, 3, and 4 which occur 4, 6 and 5 time respectively. As minimum support is 5 thus index 1 is pruned from local tree. The corresponding crimes are C1, C5 and C3 Crime 1 is not locally frequent thus eliminated. Now construct the local CFP-tree projection for crime C7 is as follows:

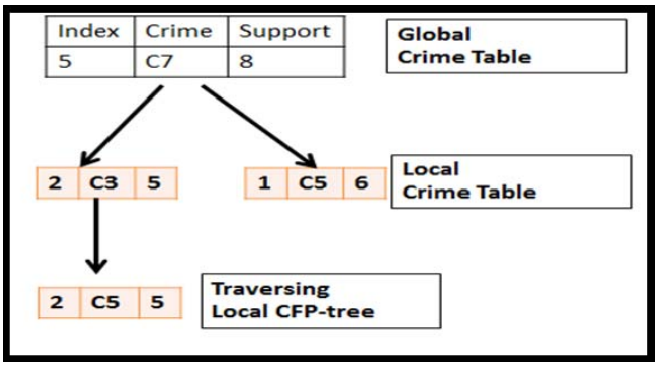

Figure 7: Frequent crime sets in Projection 5

The frequent crimes are that can be easily found by the above projection for index 5 is as follows: (C7, C3,C5:5), (C7, C5: 6), (C7, C3: 5) Similarly the frequent patterns are easily found for other indexes.

2) Algorithm: Mine Global CFP-Tree

1. Take Global CFP-tree as input.
2. Start from least frequent crime index, check all the indexes come together in the global tree with the desired index.
3. Count support of all the indexes find on above step.
4. Prune all those indexes whose support is less then minimum support means those are not locally frequent.
5. Construct the local CFP-tree by those remaining indexes by again mapping along with the support.
6. Join the links between the crimes as same the linkage in the global CFP-tree in between only that crime's index i.e. parent should be parent and child will be child of particular existing projection.
7. The new supports of nodes are the support of frequent crimes of that particular projection.
8. Repeat the process until no index is left.

In this way CFP-tree Provide facility to easily mine the frequent crimes with the help of projections which prune the not frequent crimes locally and utilize the memory space efficiently by mining projections one by one. For large database the crimes can also easily fit into main memory.

V. COMPARISON OF ALGORITHMS

As from reviewing the various techniques i.e. FP-Growth [2], COFI-Tree [8], CT-Pro [12] and many more [6, 9, 11, 13, 14, and 15] we can differentiate them by the following considerations:

TABLE 4
COMPARISON OF FP-GROWTH, COFI-TREE, CT-PRO ALGORITHMS

Algorithm / Parameters	FP-Growth	COFI – Tree	CT-Pro
Structure	Simple tree based structure.	Use Bidirectional FP-Tree structure.	Use compressed FP – Tree data structure.
Approach	Recursive	Non – Recursive	Non – Recursive
Technique	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support.	It constructs bidirectional FP – Tree and builds the COFI – Trees for each crime then mines the COFI – Tree locally for each crime.	It constructs the compact FP-Tree through mapping into index and then mine frequent crime sets according to projections index separately.
Memory Utilization	Low as for large database complete tree structure cannot fit into main memory.	Better, Fit into main memory due to mining locally in parts for the complete tree, Thus every part represent in main memory.	Best, as compress FP – Tree structure used and mine according to projections separately thus easily fit into main memory.
Databases	Good for dense databases.	Good for dense as well as sparse databases. But with low support in sparse databases performance degrades.	Good for dense as well as for sparse databases.

VI. CONCLUSIONS

FP-Growth is the first successful tree base algorithm for mining the frequent crime sets. As for large databases its structure does not fit into main memory therefore new techniques come into pictures which are the variations of the classic FP-Tree. FP-Growth recursively mine the frequent crime sets but some variations COFI -Tree and CT-PRO based upon non recursive. Pruning method consists of removing all the locally non frequent crimes and also COFI-Tree and CT-PRO need less memory space and comparatively fast in execution then the FP-Tree because of their compact and different mining techniques.

ACKNOWLEDGMENT

This research paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings.

REFERENCES

[1] R.Agrawal, R.Srikant, "Fast algorithms for mining association rules", Proceedings of the 20th Very Large DataBases Conference (VLDB'94), Santiago de Chile, Chile, 1994, pp. 487-499.
[2] J.Han, J.Pei and Y.Yin., "Mining frequent patterns without candidate Generation", in: Proceeding of ACM SIGMOD International Conference Management of Data, 2000, pp.1-12
[3] Jiawei Han, M.Kamber, "Data Mining-Concepts and Techniques", Morgan Kanufmann Publishers, Sam Francisco, 2009.
[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of crimes in large databases. In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data, pages 207–216, Washington, D.C., May 1993.
[5] M.-L. Antonie and O. R. Za¨ıane. Text document categorization by term association. In IEEE International Conference on Data Mining, pages 19–26, December 2002.
[6] J. Hipp, U. Guntzer, and G. Nakaeizadeh. Algorithms for association rule mining - a general survey and comparison. ACM SIGKDD Explorations, 2(1):58–64, June 2000.
[7] M. El-Hajj and O. R. Za¨ıane. Inverted matrix: Efficient discovery of frequent crimes in large datasets in the context of interactive mining. In Proc. 2003 Int'l Conf. on Data Mining and Knowledge Discovery (ACM SIGKDD), August 2003.
[8] M. El-Hajj and O. R. Za¨ıane: COFI-tree Mining:A New Approach to Pattern Growth with Reduced Candidacy Generation. Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA, CEUR Workshop Proceedings, vol. 90 (2003)
[9] R. P. Gopalan and Y. G. Sucahyo, "High Performance Frequent Pattern Extraction using Compressed FPTrees", Proceedings of SIAM International Workshop on High Performance and Distributed Mining (HPDM),Orlando, USA, 2004.
[10] FIMI, "FIMI Repository", 2004, http://fimi.cs.helsinki.fi, last accessed at 20/04/2011.
[11] Y. G. Sucahyo and R. P. Gopalan, "CT-ITL: Efficient Frequent Item Set Mining Using a Compressed Prefix Tree with Pattern Growth", Proceedings of the 14th Australasian Database Conference, Adelaide, Australia, 2003.
[12] Y. G. Sucahyo and R. P. Gopalan, "CT-PRO: A Bottom Up Non Recursive Frequent Itemset Mining Algorithm Using Compressed FP-Tre Data Structure". In proc Paper presented at the IEEE ICDM Workshop on Frequent Itemset Mining Implementation (FIMI), Brighton UK, 2004.
[13] A.M.Said , P.P.Dominic, A.B. Abdullah; " A Comparative Study of FP-Growth Variations", In Proc. International Journal of Computer Science and Network Security, VOL.9 No.5 may 2009.
[14] http://www.enggjournals.com/ijcse/doc/IJCSE11-03-07-092.pdf
[15] S.P Latha, DR. N.Ramaraj, "Agorithm for Efficient Data Mining", Proceeding on IEEE International Computational Intelligence and Multimedia Aplications 2007, pp. 66-70.
[16] Q.Lan, D.Zhang, B.Wu, "A New Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree", Proceeding on IEEE International Conference on Global Congress on Intelligent System, 2009, pp.360-364.