DATA MINING:FREQUENT TREE MINING ANALYSIS

MR. DHARMESH N. MEHTA

M.E.[Computer Science and Engineering] Student, Department Of Computer ,Kalol Institute Of Technology and Research, Kalol, Ahmedabad, Gujrat.

dnm13ster@gmail.com

***ABSTRACT*:** Frequent pattern mining has been a focused theme in datamining research for over a decade.Abundant literature has been dedicated to this research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining, associative classification, and frequent pattern-based clustering, as well as their broad applications. In this article, I provide a brief analysis on the current status of frequent pattern mining and discuss a few promising and novel algorithms. Frequent pattern mining research has substantially broadened the scope of data analysis and will have deep impact on data mining methodologies and applications in the long run. However, there are still some challenging research issues that need to be solved before frequent pattern mining can claim a cornerstone approach in data mining applications.

Keywords—Frequent pattern mining

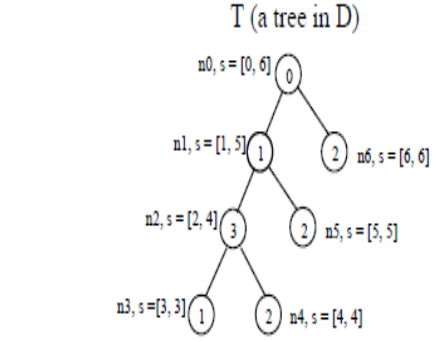
I: INTRODUCTION

Frequent patterns are itemsets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set, is a *frequent itemset*. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (*frequent*) *sequential pattern*. A *substructure* can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently in a graph database, it is called a (*frequent*) *structural pattern*. Finding frequent patterns plays an essential role inmining associations, correlations, and many other interesting relationships among data. Moreover, it helps in data indexing, classification, clustering, and other data mining tasks as well. Thus, frequent pattern mining has become an important data mining task and a focused theme in data mining research.

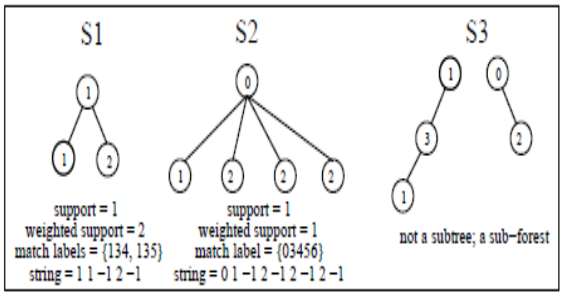
II: VARIOUS NOVEL ALGORITHMS

1.TREE MINER

TREE MINER(D,minsup):
F1={frequent 1-subtrees};
F2={classes $[P]_1$ of frequent 2-subtrees};
For all $[P]_1 \in E$ do Enumerate-Frequent-Subtrees($[P]_1$);
ENUMERATE-FREQUENT-SUBTREES([P]):
For each element (x,i) Є [P] **do**
$[P_x]=\Phi$;
For each element (y,j) Є [P] **do**
R={(x,i) EX-OR (y,j)};
L(R) = { L(x) ∩ L(y)};
If for any R Є R, R is frequent **then**
$[P_X] = [P_x] \cup \{R\}$;
Enumerate-Frequent-Subtrees($[P_x]$);

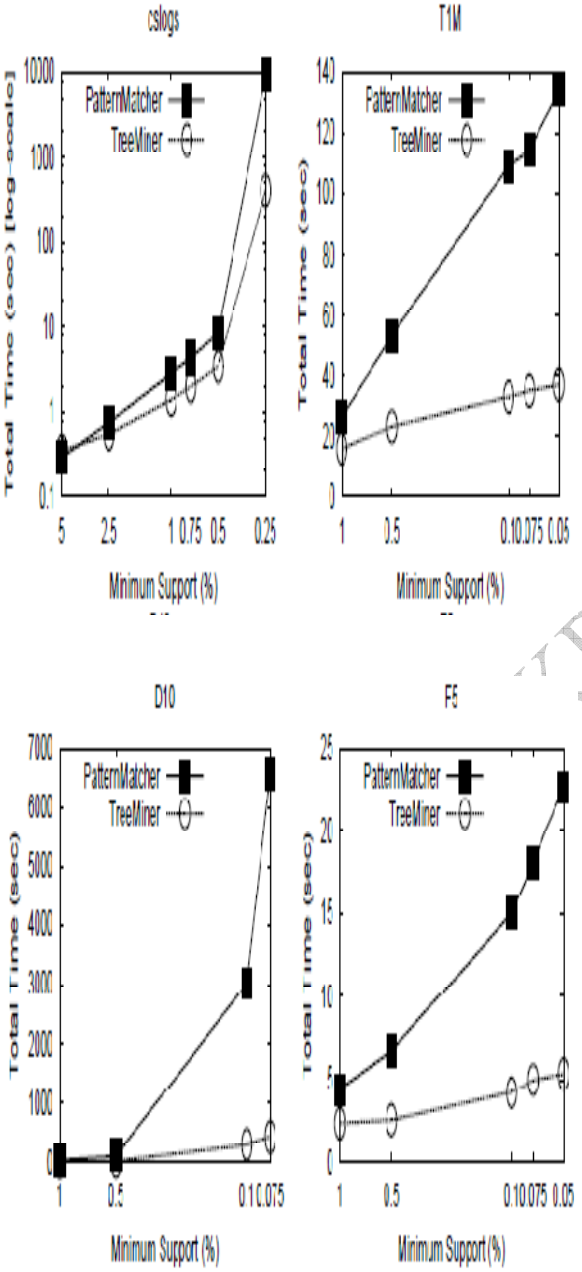


T's String Encoding: 0 1 3 1 −1 2 −1 −1 2 −1 −1 2 −1



An Example Tree with Subtrees

Looking upon performance of TREE MINER, it uses depth-first search, it also uses novel scope-list vertical representation of trees to quickly compute candidate tree frequencies via scope-list joins based on interval algebra.
Experiments on real and synthetic data confirmend that TREE MINER out performs PATTERN MATCHER from factor of 4 to 20 and scales linearly in the number of trees in the forest.



Performance Comparison

2.FREQUENT PATTERN TREE(MINING FREQUENT PATTERN WITHOUT CANDIDATE GENERATION)

Frequent pattern mining plays an essential role in mining associations,correlations,causality , sequential patterns , episodes , multi- dimensional patterns,max-patterns, par- tial periodicity , emerging patterns, and many other important data mining tasks.

Algorithm (FP-tree construction)
Input: A transaction database DB and a minimum support threshold _.
Output: Its frequent pattern tree, FP-tree
Method: The FP-tree is constructed in the following steps.
1. Scan the transaction database DB once. Collect the set of frequent items F and their supports. Sort F in support descending order as L, the list of frequent items.
2. Create the root of an FP-tree, T, and label it as \null". For each transaction Trans in DB do the following.
Select and sort the frequent items in Trans according to the order of L. Let the sorted frequent item list in Trans be [pjP], where p is the element and P is the remaining list. Call insert tree([pjP]; T). The function insert tree([pjP]; T) is performed as follows. If T has a child N such that N.item-name = p.item-name, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link be linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree(P;N) recursively.

Analysis. From the FP-tree construction process, we can see that one needs exactly two scans of the transaction database, DB: the _rst collects the set of frequent items, and the second constructs the FP-tree. The cost of inserting a transaction Trans into the FP-tree is O(jTransj), where jTransj is the number of frequent items in Trans. We will show that the FP-tree contains the complete information for frequent pattern mining.

Algorithm 2 (FP-growth: Mining frequent patterns with FP-tree by pattern fragment growth)

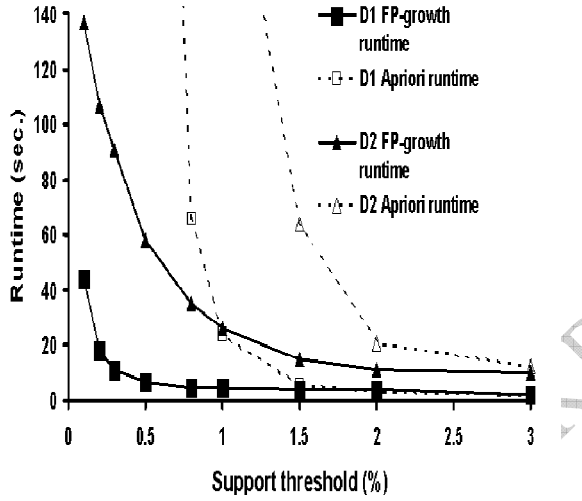
Input: FP-tree constructed based on Algorithm 1, using DB and a minimum support threshold.
Output: The complete set of frequent patterns.
Method: Call FP-growth (FP-tree ; null).

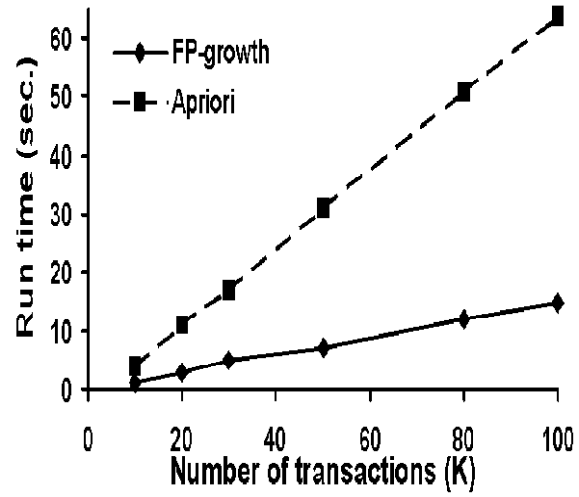
Procedure FP-growth (Tree,α)

(1) **if** Tree contains a single path P
(2) **then for each** combination (denoted as β) of the nodes in the path P **do**
(3) generate pattern β U α with support = minimum support of nodes in β;
(4) else for each ai in the header of Tree do f
(5) generate pattern $\beta = a_i \cup \alpha$ with support = ai support;
(6) construct β's conditional pattern base and

then β's conditional FP-tree $Tree_\beta$;
(7) if $Tree_\beta = \Phi$;
(8) then call FP-growth ($Tree_\beta$,β) }
}
Analysis. With the properties and lemmas in Sections 2 and 3, we show that the algorithm correctly Finds the complete set of frequent itemsets in transaction database DB.

FP-growth scales much better than Apriori. This is because as the support threshold goes down, the number as well as the length of frequent itemsets increase dramatically. The candidate sets that Apriori must handle becomes extremely large, and thepattern matching with a lot of candidates by searching through the transactions becomes very expensive.





3.TRIPS AND TIDES ALGORITHM

TRIPS stands for TRee mIning algorithms using Prufer Squences .TIDES stands for Tree mIning algorithm using DEpth first order Sequences. TRIPS and TIDES di_er only in their candidate generation step. The support counting step is the same for both the algorithms. Method of candidate generation in both the algorithms is quite similar in its spirit except for a few di_erences. While TRIPS restricts the growth points to the Left Most Path of the pattern, TIDES restricts them to the Right Most Path (RMP).

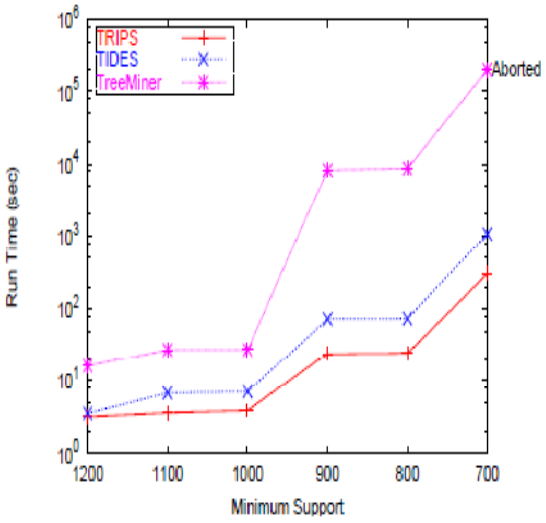
Two algorithms are given here.

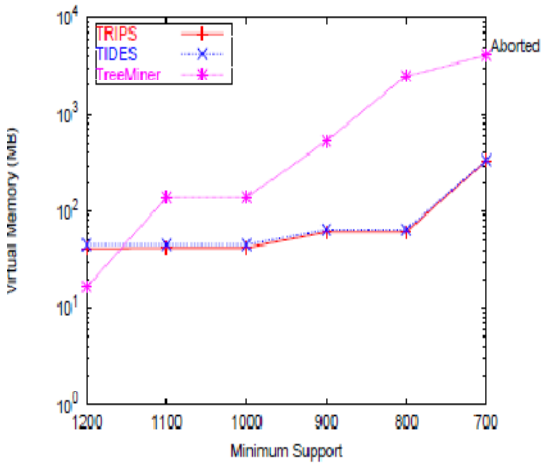
Algorithm 1 Subtree Mining Algorithm
Require: D = fT1; T2; : : : ; TNg; minsup
1: F1 = readTrees (D)
2: for all v in F1 do
3: mineTrees (NULL, (v, □1), D)
4: end for
Algorithm 2 Mining a given pattern
mineTrees (pat, (l, pos), tidlist)
1: newpat = extend (pat, l, pos)
2: newtidlist = NULL
3: for all T in tidlist do
4: if (l, pos) is an extension point for pat in T then
5: update embedding list of T
6: add T to newtidlist
7: end if
8: end for
9: H = NULL
10: for all T in newtidlist do
11: Scan T for extension points of newpat
12: Add generated extension points to H
13: end for
14: for all h in H do
15: if h:support _ minsup then
16: mineTrees (newpat, (h:l, h:pos), newtidlist)
17: end if
18: end for

TRIPS is clearly scalable and continues to perform better as the dataset size increases. It achieves a speedup of 11:6 on DS1- 40K and a speedup of 8:6 on DS4-100K.

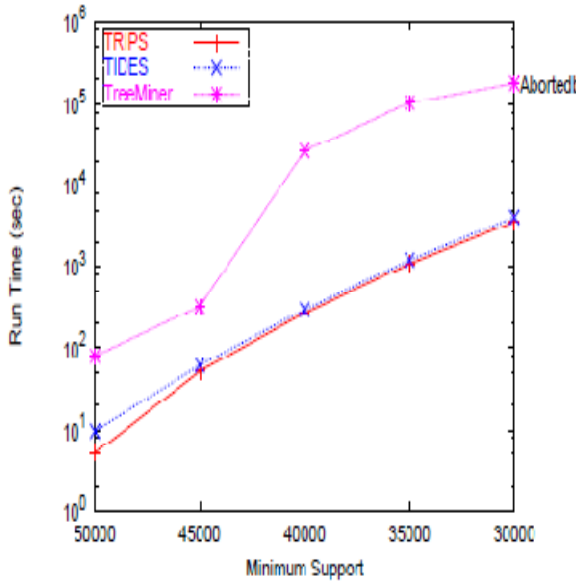
The performance of TIDES is similar to that of TRIPS on all synthetic datasets.

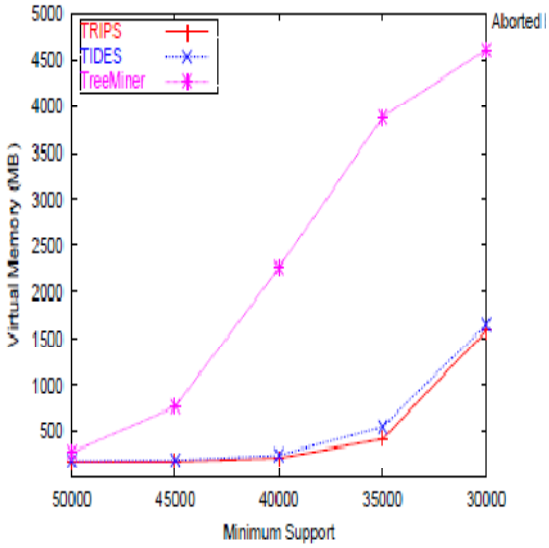
With TREEBANK dataset at minsup = 45000. Both TRIPS and TIDES exhibit excellent cache performance with (L1 hit rate, L2 hit rate, CPI) of (99:6, 99:94, 0:72) and (99:7, 99:96, 0:7), respectively. Whereas, TreeMiner produced hit rates and CPI of (96:59, 99:96, 1:2).





Performance on CSLOGS dataset (a) Run Time (b) Virtual





Performance on TREEBANK dataset

VI: CONCLUSION

In this article, we present a brief overview of the current status and future directions of frequent pattern mining.With over a decade of extensive research, there have been hundreds of research publications and tremendous research, development and application activities in this domain.

It is impossible for us to give a complete coverage on this topic with limited space and our limited knowledge. Hopefully, this short overview may provide a rough outline of the recent work and give people a general view of the field.

In general, we feel that as a young research field in datamining, frequent pattern mining has achieved tremendous progress and claimed a good set of applications However, in-depth research is still needed on several critical issues so that the field may have its long lasting and deep impact in data mining applications.

REFRENCES

[1] Mohammed J. Zaki," **Efficiently Mining Frequent Trees in a Forest",** Computer Science Department, Rensselaer Polytechnic Institute, Troy NY,2002.

[2]Jiawei Han, Jian Pei, and Yiwen Yin," **Mining Frequent Patterns without Candidate Generation",** School of Computing Science ,Simon Fraser University,2005.

[3] Shirish Tatikonda, Srinivasan Parthasarathy, and Tahsin Kurc," **TRIPS and TIDES: New Algorithms for Tree Mining",** *CIKM'06,* November 5–11, 2006

[4] Jiawei Han · Hong Cheng · Dong Xin · Xifeng Yan," **Frequent pattern mining: current status and future directions",** Received: 22 June 2006 / Accepted: 8 November 2006 / Published online: 27 January 2007 Springer Science+Business Media, LLC 2007