A Parallel Approach to ModifiedRPset

R.Prabamanieswari, Research Scholar [1], **D.S.Mahendran,** [2]Principal,**T.C. Raja Kumar[3]**, [3]Associate Professor

Research Scholar [1] (Reg No. 11991), Manonmaniam Sundaranar University, Abishekapatti, Tirunelveli -12, prabacs_2006@yahoo.com

(Resarch Center: St. Xavier's College, Palayamkottai)

[2]Principal, Aditanar College of Arts & Science, Tiruchendur, India,dsmahe65@yahoo.com

[3]Associate Professor, Department of Computer Science, St. Xavier's College, Palayamkottai, India,grajazion@gmail.com

ABSTRACT

Recently, most of the applications will not need precise support information for frequent pattern. They approximate the frequency of every frequent pattern with a guaranteed maximal error bound. But, these algorithms generally have some problems and challenges while processing large-scale data. Therefore, the parallel approach has emerged as an important research topic. This paper implements the algorithm ModifiedRPset in a parallel manner while creating NCFP-tree. It follows the concepts such as considering closed patterns, finding C(X)s for closed patterns and applying greedy set cover algorithm which are same as in the previous algorithms MinRPset and ModifiedRPset. The experiment results show that the proposed algorithm gives better execution time for finding a small number of representative patterns to best approximate all other frequent patterns comparing to the algorithm ModifiedRPset.

Keywords: Frequent itemset, threaded NCFP-tree, greedy method, representative pattern sets.

1.INTRODUCTION

Efficient algorithms for mining frequent itemsets are crucial for mining association rules as well as for many other data mining tasks. Different methods introduced by different researchers generated the frequent itemsets by using the candidate generation [2] as well as without candidate generation [3]. Many algorithms and techniques based on tree [3] are posed for enumerating itemsets from transactional databases. Let the transactional database D= $\{t_1, t_2, \ldots, t_n\}$, where t_j is a transaction containing a set of items, $j \in [1,n]$. Let $I = I_1, I_2 \ldots\ldots, I_m$ be a set of m distinct attributes and t be transaction that contains a set of items such that $T \subseteq I$. Each subset of I is called an itemset. If an itemset contains k items, then the itemset is called a k-itemset. The support of itemset X in database D is defined as the percentage of

transactions in D containing X, that is, support t_D (X) = {t | t∈D and X ⊆ t}/D. If the support of a pattern X is larger than a user specified threshold min-sup (min-sup ∈(0, 1]), then X is called a frequent pattern. Given a transaction database D and a minimum support threshold min-sup, the task of frequent pattern mining is to find all the frequent patterns in D with respect to min-sup.

Nowadays, most of the applications will not need precise support information for frequent pattern. They approximate the frequency of every frequent pattern with a guaranteed maximal error bound. Recently, several approaches have been proposed to tackle the concise representation of frequent itemsets such as top-k frequent patterns [6], top-k redundancy-aware patterns [8], and error-tolerant patterns [5]. When considering the availability of massive volume of data, analyzing and decision making is still a major issue. To address this issue, it is necessary to study parallel implementations of data mining algorithms. In order to create parallel algorithms it is important to have an efficient way to tell the operating system to create either threads or processes. Many parallel problems are solved through the decomposition of data by creating threads to work on the data in parallel. In this paper, we propose a parallel formulation of the NCFP-Tree algorithm using multiple threads.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the parallel construction of NCFP-tree and generation of representative pattern sets with example. The experimental results are shown in section 4. Finally, section 5 concludes the paper.

2. RELATED WORK

In many applications, knowing the approximate support of frequent patterns is sufficient. Several approaches are proposed to make a trade-off between pattern set size and the precision of pattern support. Xin et al. [7] proposed two algorithms, RPglobal and RPlocal. The algorithm RPglobal is very time-consuming and space-consuming. RPlocal is very efficient, but it produces more representative patterns than RPglobal. G.Liu et al [11] analysed the bottlenecks of RPglobal and RPlocal and developed two algorithms, MinRPset and FlexRPset, to solve the problem. The algorithm MinRPset is similar to RPglobal, but it utilizes several techniques to reduce running time and memory usage. In particular, MinRPset uses a tree structure called CFP-tree [9] to store frequent patterns compactly. The algorithm FlexRPset provides one extra parameter K, which allows users to make a trade-off between efficiency and the number of representative patterns selected. In [13], the algorithm NCFPGEN for creating a NCFP-tree is proposed. The algorithm ModifiedRPset which is proposed in [15] uses NCFP-tree instead of CFP-tree and MinRPset to generate minimum representative pattern sets. Although CFP-tree and NCFP-tree algorithms

are very efficient, they still take a lot of time to find minimum representative pattern sets. To reduce the execution time, it is necessary to study parallel implementation of such algorithms.

There are many parallel association rule mining algorithms within the literature.

Most such parallel algorithms are Apriori based. A parallel tree projection based algorithm, called MLFPT, based on FP-Tree algorithm is presented in [4] for a shared memory environment. In [10] a massively parallel FP-Growth algorithm is presented, the proposed algorithm allows to eliminate virtually communication among computers. The algorithm is expressed with the MapReduce framework. Arpan H. Shah et al., proposed Optimum pattern Tree [12] on incremental database with same threshold value. This algorithm is implemented on Hadoop to reduce the computation cost. Swathika.K and Helen W.R proposed a data partition [14] based Frequent Itemset Mining model over Hadoop that could be used to classify the type of tumour based on the prediction. In this paper, the proposed work creates parallel construction of sub trees of NCFP-tree [13] for each frequent 1-item in the header table.

3. PARALLEL REPRESENTATIVE PATTERN SETS MINING

Our proposed parallel representative Pattern Sets mining algorithm ParallelRPset consists of parallel construction of NCFP-tree using a thread concept and mining of the tree structure NCFP-tree for generating representative Pattern Sets**.** A detailed description of each major step is described in subsequent sections.

3.1 Parallel NCFP-tree Construction

The parallel pre-processing is performed in order to construct the final data structure that is going to be used for the mining. Given transactional database D and a minimum support threshold, finding of all frequent items and removing infrequent items in D are performed. Then, a conditional database D′and an extended conditional database D′′are constructed. We use a header table to maintain the set of frequent items u_i in a database. The frequent items are sorted into ascending frequency order in the header table. The mining is performed on D′′ to mine all the itemsets starting with u_i. Now, a thread concept is applied to process unique item u_i in the header table. This is accomplished by letting each thread to create its own NCFP -tree, from its own node corresponds to header table. The process in NCFP-tree [13] such as creating single node and multiple nodes is done simultaneously by using multiple threads. The sub trees are constructed in parallel manner instead of processing each unique item u_i separately for creating sub trees. Thus, the same NCFP-tree is created in parallel manner.

3.2 GENERATION OF REPRESENTATIVE PATTERN SETS

Now, we use NCFP-tree to generate C(X)—the set of frequent patterns that X covers and to find minimum number of representative pattern sets. Here, we use ModifiedRPset [15] MinRpset [11] and the algorithm greedy [1] to generate the minimum number of representative pattern sets.

Algorithm: ParallelRPset

Input:

D is the database

min-supp is the minimum support threshold

Output:

Minimum number of Representative Pattern Sets

Description:

1. Find frequent items, remove infrequent items and sort frequent items
2. Create conditional database D′ and extended conditional database D′′
3. Create a header for sorted frequent 1-itemsets (unique items u_i)
4. For each unique item $u_i \in$ header

   Create NCFP-tree sub tree using thread
5. Apply ModifiedRPset to find minimum number of Representative Pattern Sets

EXAMPLE

Consider the Transactional Database given in Table I. It has seven transactions, that is |D|=7. Assume minsup= 40%. The set of frequent itemsets are determined and are given in Table II. The construction of NCFP-tree in parallel manner is done by using six threads such as Thread 1, Thread 2..... and Thread 6. It is represented in programming aspect as:

Thread 1	Thread 2	Thread 3
c	d	p
c-->|-|m|-|a		p-->|-|f
		p-->-->m
		p-->-->-->a
		p-->|-|f-->m
		p-->|-|f-->-->a
		p-->|-|f-->m-->|-|a
Thread 4	Thread 5	Thread 6
f	m	a
f-->m	m-->|-|a	
f-->-->|-|a		
f-->m-->|-|a		

The following figure shows the parallel construction of NCFP-tree.

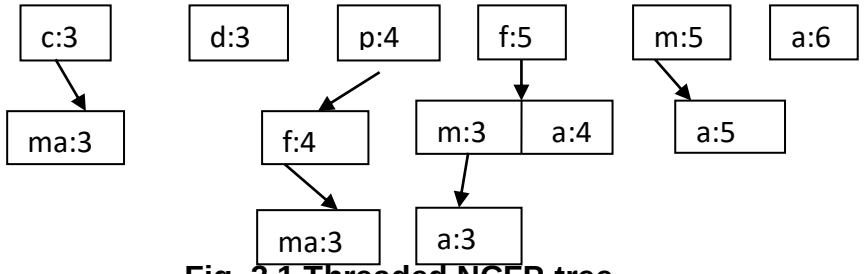

Fig. 3.1 Threaded NCFP-tree

The Threaded NCFP-tree is same as NCFP-tree [13]. Here, we no need to merge all sub trees because all sub trees are constructed simultaneously. Table III shows the compact representation of frequent itemsets which are stored in NCFP-tree.

Table I. Transaction Database

TID	TRANSACTION
1	a, c, e, f, m, p
2	a, b, f, m, p
3	a, b, d, f, g
4	d, e, f, h, p
5	a, c, d, m, v
6	a, c, h, m, s
7	a, f, m, p, u

Table II. Frequent Itemsets

Frequent Itemsets(Compact Form) (min_sup = 40%)
cma:3
d:3
pfma:3 pf:4
fma:3 fa:4 f:5
ma:5
a:6

Table III. Compact Itemsets

Closed Patterns (min-sup=40%)
f:5, a:6
pf:4, fa:4,ma:5
cma:3
pfma:3

The procedures such as finding C(X)s and determining representative pattern sets are done either by using MinRPset [11] or by using ModifiedRPset [15]. Because ModifiedRPset follows the same procedures for finding C(X)s and applying Greedy method from MinRPset. The C(X) contains the subsets of X that can be$\in$-covered by X, for every X $\in$ F$\hat{}$ where F$\hat{}$ denotes frequent patterns with support ≥ min_sup * (1−$\in$) and F denotes frequent patterns with support ≥ min_sup. The C(X) is determined only for closed patterns. Therefore, for each itemset belongs to Table III is first checked whether it is closed or not then C(X) is determined. This example determines C(X)s only for the itemsets which are given in Table IV.

Table IV. Closed Frequent Itemsets

Closed Patterns (min-sup=40%)
f:5, a:6
pf:4, fa:4,ma:5
cma:3
pfma:3

Finally, the minimum representative pattern sets are discovered. In this example, the minimum representative pattern sets are { {pfma:3}, {cma:3}, {d:3}}.

4. EXPERIMENTAL RESULTS

The experiment is carried out on the computer with the configuration such as Intel(R) Core(TM) i3CPU, 3 GB RAM, 2.53 GHz Speed and Windows 7 Operating System. Three approaches such as MinRPset, ModifiedRPset and ParallelRPset are implemented in java. The experiment is evaluated on mushroom dataset. The mushroom dataset contains the characteristics of various species of mushrooms. It

has 119 items and 8124 transactions. The minimum, maximum and average length of its transaction is 23. It is obtained from the UCI repository of machine learning databases.

4.1 EXECUTION TIME

Here, the experiment considers execution time for finding the performance of the proposed algorithm ParallelRPset. The proposed algorithm is compared with MinRPset and ModifiedRPset. The algorithms are tested on the mushroom dataset. The figures Fig 4.1 and Fig 4.2 show the execution time of MinPRset, ModifiedRPset and Parllel-RPset algorithms. Fig 4.1 shows the execution time when considering sample data which is given in Table I. The min-supp is varied from 20% to 100% in increments of 20% and $\in$ = 0.25. We observed that there is a small difference between ModifiedRPset and MinRPset but, ParallelRPset has better performance comparing to both approaches. Fig.4.2 shows the execution time when considering mushroom dataset. Here, $\in$ is changed from 0.1 to 0.5 and min-supp is fixed as 0.4. The ParallelRPset gives better performance than MinRPset and Modified-RPset.

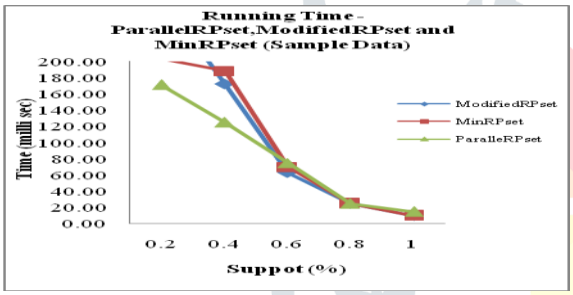

Fig 4.1. Execution Time when $\in$=0.25

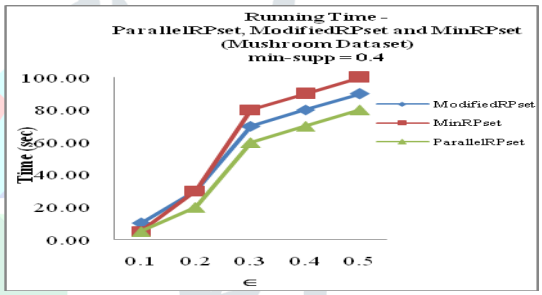

Fig 4.2. Execution Time when min-supp=0.4

5. CONCLUSION

In this paper, we constructed threaded NCFP-tree and applied it in the ModifiedRPset instead of NCFP-tree to find minimum representative pattern sets. The proposed algorithm is implemented in java programming language. It gives better execution time comparing to both algorithms MinPRset and ModifiedRPset. In the future, we intend to improve this algorithm based on our findings and run our parallel algorithm for higher number of processors with larger data sets, based on the resource availability on the machine environment.

REFERENCES

[1] V. Chvatal, —A greedy heuristic for the set-covering problem, ‖Math. Oper. Res., vol. 4, no. 3, pp. 233–235, 1979.
[2] R. Agrawal, T. Imielinski, and A. N. Swami, “Mining association rules between sets of items in large databases”, in Proc. SIGMOD, Washington, DC, USA, pp. 207–216, 1993.
[3] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, In Proceedings of ACM SIGMOD’00, pp 1–12, May 2000.
[4] O. R. Zaïane, Mohammad E.l-Hajj, and Paul Lu. Fast Parallel Association Rule Mining without Candidacy Generation. Proc. of the IEEE 2001 International Conference on Data Mining (ICDM'2001), San Jose, CA, USA, November 29-December 2, 2001.
[5] M. T. Yang, R. Kasturi, and A. Sivasubramaniam., —An AutomaticScheduler for Real-Time Vision Applications‖, In Proceedings of the International Parallel and Distributed Processing Symposium(IPDPS), 2001.
[6] J. Wang, J. Han, Y. Lu, and P. Tzvetkov, —TFP: An efficient algorithmfor mining top-k frequent closed itemsets,‖ IEEE Trans. Knowl. Data Eng., vol. 17, no. 5, pp. 652–664, May 2005.
[7] D. Xin, J. Han, X. Yan, and H. Cheng, —Mining compressed frequen-pattern sets,‖ in Proc. 31st Int. Conf.VLDB, Trondheim, Norway, pp. 709–720,2005.
[8] D. Xin, H. Cheng, X. Yan, and J. Han,—Extracting redundancy-aware top-k patterns,‖ in Proc. KDD,Philadelphia, PA, USA,pp. 444–453,2006.
[9] G. Liu, H. Lu, and J. X. Yu, —CFP-tree: A compact disk-based structure for storing and querying frequent itemsets‖, Inf. Syst., vol. 32, no.2, pp. 295–319, 2007.
[10] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. Pfp: parallel fp-growth for query recommendation. In Proceedings of the 2008 ACM conference on Recommender systems, RecSys '08, pages 107{114, New York, NY, USA, 2008. ACM.
[11] Guimei Liu, Haojun Zhang, and Limsoon Wong, —A Flexible Approach to Finding RepresentativePattern Sets‖, IEEE Transactions on Knowledge and Data Engineering,Vol. 26, No. 7,pp 1562-1574,, July 2014.
[12] Arpan H. Shah and Pratik A. Patel, “Optimum Frequent Pattern Approach for Efficient Incremental Mining on Large Databases using Map Reduce”, International Journal of Computer Applications (0975 – 8887) Volume 120 – No.4, June 2015.
[13] R.Prabamanieswari, D.S.Mahendran, T.C. Raja Kumar, —NCFP-tree: A Non Recursive Approach to CFPtree using Single Conditional Database‖, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 5 Issue XI November 2017.
[14] Swathika.K and Helen W.R, “An efficient parallel frequent itemset mining approach using Mapreduce framework”, International Journal of Pure and Applied Mathematics, Volume 118 No. 20 2018, 477-486
[15] R.Prabamanieswari, D.S.Mahendran, T.C. Raja Kumar, “A Modified Algorithm for finding Representative Pattern Sets”, International Journal of Engineering Research in Computer Science and Engineering(IJERCSE), Vol.5, Issue 3, March 2018, pp. 201-205.