Comparison of Frequent Pattern Mining Algorithms for Data Streams: A Survey

[1]Raghavendra Naik, [2]Pramilarani K
[1]Student, [2]Senior Assistant Professor
[1]Department of Computer Science & Engineering,
[1]New Horizon College of Engineering Bangalore, India

Abstract: Frequent Itemset Mining is the problems in most of the data mining applications. There are different frequent patterns mining algorithms available for parallel execution of items, such as FP-Growth, Apriori, RARM algorithms. However, as these parallel mining algorithms which have lack of features like automated parallelization, proper load balancing, and distribution of data on large clusters. We have done different analysis of frequently used algorithms for finding frequent patterns with the intension of finding how different algorithms can be implemented and used to get frequently used patterns in bigger transactional databases. This has been represented done a comparative survey on few of the following algorithms: Frequent Pattern Growth algorithm, Apriori algorithm, Rapid Association Rule Mining (RARM), Associated Sensor Pattern Mining of Data Stream (ASPMS) and ECLAT algorithm frequent pattern mining algorithms. This study finds each algorithm's performance, advantages, and disadvantages for large scale of item sets in database systems**.**

Index Terms **- Frequent Pattern Growth, Apriori, Rapid Association Rule Mining (RARM), ECLAT, Data Mining, Frequent Patterns, MapReduce.**

I. INTRODUCTION

The main subject matter in data mining is mining the frequent patterns. There are a lot of researches have been made and lots of efficient algorithms have been designed to search frequent pattern in the large transactional database. Agrawal et al. (1993) for the first time proposed a concept of pattern mining in the form of market-based analysis for finding the relation between items that are fetched in a market. This concept used transactional databases and other data repositories in order to extract association's casual structures, interesting correlations or frequent patterns among the set [1]. Frequent patterns are the items or itemsets which repeatedly occur in database transactions with a user-specified frequency. An itemset whose occurrence frequency is greater than the minimum threshold will be considered as the frequent pattern. For example in market based analysis if the minimum threshold is 30% and bread appears with eggs and milk more than three times or at least three times then it will be a frequent itemset [2].

In mining pattern stage different techniques are applied to find candidates for frequent patterns and then frequent patterns are generated. There are two main problems with frequent pattern mining techniques. The first problem is that the database is repeatedly scanned for each search, and the second is complex candidate datasets are generated for each scan and process to scan the amount is huge. These two are the main problems in frequent pattern mining. Studies demonstrate that a lot of efforts have been performed for devising best techniques and worth mentioning approaches are Apriori, RARM, ECLAT, FP Growth, and ASPMS algorithms [3].

II. PROBLEM STUDY

Various studies have been acknowledged on Frequent Pattern itemsets in the field of Data mining as it has a broad range of applications in sequential pattern searching, correlations, association rules in mining, frequent pattern based graph constraints, and various data mining tasks. It is most crucial to find an efficient mining algorithm for frequent itemsets and numerous result patterns. As the frequency of the items are higher and amount of data generated, it results in generating a huge number of result sets are generated. For this reason pruning methodologies are used to eliminate unwanted patterns and efficiency and speed are increased in mining process. As a result of this, the main aim for us to optimize the process of data mining for frequent patterns and also yields in scalable, efficient and can find the important patterns which is suitable in different areas and methods.

III. RELATED WORK

Frequent pattern mining techniques have become an obvious need in many real world applications e.g. in market basket analysis, advertisement, medical field, monitoring of patients routines etc [9]. To make a comparison among these algorithms, we use the same transactional database for all algorithms, this transactional database is based on data of a smart home where sensors are installed on daily usage objects and patients while performing their daily tasks, touch these objects and these sensor items are maintained in database. Studies of Frequent pattern mining is acknowledged in the data mining field because of its applicability in mining sequential patterns, structural patterns, mining association rules, constraint based frequent patterns mining, correlations and many other data mining tasks [10]. Efficient algorithms for mining frequent Itemsets are crucial for mining association rules as well as for some other information mining assignments [13]. The Problem of mining frequent itemset ascended first as sub-problem of mining association rules [5].

A database consists of transactions and a transaction is denoted by T. Let there is an itemset I= {I1, I2,…In} consist of n items. A Transaction T contains a subset of items from itemset I [4]. Association rule is in the form of inference stating such that if x then y($x \rightarrow y$) where x and y both are subset of items in the itemset I. As transactional database is large and we are interested in those items that are used frequently, there is an important parameter "support" that helps in identifying those items that are of interest. Support for an association rule ($x \rightarrow y$) is defined as no of transactions or percentage which contains xUy over total number of transactions in database. Minimum lower bound of support for association rule is set by user and this support value is set as a minimum threshold and itemset whose number of transactions is above than defined threshold is considered as frequent itemset. As this threshold is a large value, more valuable knowledge is obtained and if this threshold is a minimum value, a large number of Itemsets are generated. Therefore irrelevant information should be pruned, that is the main goal of frequent pattern

mining. In order to analyze different frequent pattern mining algorithms in coming paragraphs comparative analysis of these algorithms have discussed with the purpose to investigate their strengths and weaknesses in order to utilize their effectiveness in respective field.

Frequent item set mining used in wide range of application areas such as decision support, web usage mining, bioinformatics, etc. There are varieties of algorithms proposed by different researchers for Frequent Itemset Mining. Each of it has its own advantages and disadvantages. Following is the review of some of the research papers from various conferences and publications.

The proposed research work [1] presents the problem related with extraction of frequent items from huge datasets. It gives the rules that have the minimum confidence and least transactional support. The proposed algorithm calculates the itemsets for the very first pass and automatically has to adjust among the number of passes that are for itemsets and data. The calculation here their use is to shortening technique to avoid certain itemsets. So, it gives proper related itemsets/datasets from considerably greater size databases. Advantage of this algorithm is, buffer management techniques is the items which could not in the memory in current pass will be shifted to next pass, also it improves the execution due to enhances parallelization technique, but it has inability of automatic parallelization. Parallel Frequent Pattern Growth algorithm using balanced partitioning (BPFP) [8] is another great system proposed for pattern mining. It works in two stages, in the first stage of BPFP, load is computed based on the conditional pattern. In the other stages, the load is divided into many groups. For this reason, MapReduce operation is used on frequent itemset mining algorithm.

IV. Association Rule Mining Algorithm

4.1 Problem Statement of ARM

Association Rule mining is one of the important data mining algorithm used in many applications.

4.1.1 Generation of Frequent Itemsets

Frequent itemsets from different data sources will be collected into a global database. As there are a lot of data are combined and moved the frequent data to global database will leads to increases in, the number of messages that need to be processed to find frequent n itemset. The major drawback with the frequentset mining methods is the explosion of the numerous results and therefore it is difficult to find the most suitable frequent itemsets. Therefore the concepts of finding frequent itemsets from a database are the major challenges.

4.1.2 Mining associations Rules

From the formal statement of association rule mining, rule can be defined as: Let I = {i1, i2, ..., in} be a set of n number of binary attributes called items. Let a set of transactions D can be defined as D = {t1, t2, ..,tn} which are called the database. For each transaction in D has given a unique transaction ID and contains a subset of the items in I. A association rule is defined as an implication of the form X->Y where X, Y ⊆I and X ∩ Y_ = ∅. The sets of items (for short itemsets) X and Y are set of items in I and consequent of the rule another side respectively. The association rule would be defined like associating one item likely by the characteristic of another or nearest item. The problem is to generate all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence. In the first pass, the support of each individual item is counted, and the large ones are determined. In each subsequent pass, the large itemsets determined in the previous pass is used to generate new itemsets called candidate itemsets. The support of each candidate itemset is counted, and the large ones are determined. This process continues until no new large itemsets are found.

In short we can say that ARM is a two steps process - (i) Generation of frequent itemsets whose support is greater than or equal to the minimum support threshold set by user from database D and (ii) Strong association rules that have confidence greater than or equal to the minimum confidence threshold set by user are generated from these frequent itemsets [2], [3].

4.2 Parallel Apriori Algorithm

Apriori is the basic and most popular algorithm proposed by R. Agrawal and R. Srikant [3] which generates Candidate datasets and key to find frequent itemsets. Candidate datasets are itemsets having all the frequent itemsets. The algorithm is based on the properties of Apriori which denotes that all non-empty subsets of a frequent itemset should be frequent [2]. The core step of the algorithm is generation of candidate k-itemsets Cj from frequent (j-1)-itemsets Lj-1 and it consists of join and prune actions. In join step, the conditional join of Lj-1. Lj-1 is assigned to a candidate dataset Cj and the size of Cj is reduced by pruning some steps using Apriori property [2].

The disadvantages of serial algorithms become high due to continuous and repeatedly scanning of database and large memory consumption and cost of I/O operations will be high. To improve the performance of Apriori algorithm, there are many parallel and sequential algorithms have been introduced. Few among the popular sequential approaches are Partitioning, Transaction reduction, Hash-based technique, Dynamic itemset counting (DIC) and Sampling [15], [16], [17], [18], [19]. According to R. Agrawal and J. Shafer's [7] proposal, there are three parallel version of Apriori algorithm, Data Distribution (DD), Candidate Distribution, and Count Distribution (CD).

Count Distribution and Data Distribution algorithms are placed under task parallelism and data parallelism where as the Candidate Distribution algorithm is the combination of and task and data parallelism [11].

V. Apache Hadoop MapReduce Framework

The large-scale distributed batch processing infrastructure for parallel processing of big data on a huge cluster of valuable computers is called as Hadoop. [13]. Hadoop is an open source project of Apache [13]. This has implemented Google's File system [14] as Hadoop Distributed File System (HDFS) and Google's MapReduce [17] as Hadoop MapReduce programming.

5.1 Hadoop Distributed File System

A distributed file system that holds a large quantity of data in terabytes or petabytes scale, also provides fast and scalable access to such huge data, this is called as Hadoop Distributed File System (HDFS) [13]. This can store the files in a replicated manner across different machines. This also helps in providing fault tolerance and high availability during the execution of parallel applications [13].

HDFS file system is in a block-structured manner, this breaks a single file into fixed size blocks (default block size is 64MB) to store across several machines. Hadoop makes use of 2 types of machine working in a master-worker fashion. Those are, (i)NameNode as master machine (ii) number of DataNodes as worker machines. The work of NameNode is to assign block ids to the blocks of a file and stores metadata (file name, permission, replica, location of each block) of the file system in its main memory. This helps in providing quick access to the information stored. DataNodes are used to store and retrieve the replicated blocks of multiple files, these are individual machines in a cluster [13].

5.2 Hadoop MapReduce

In a large number of machines, a program that can be used for parallel processing of large volumes of data by breaking the work into independent tasks is called as MapReduce. The list processing languages e.g. LISP are the inspiration for MapReduce. MapReduce makes use of two list processing phrases: map and reduce. Based on the processing phrases, MapReduce program consists of two functions called as Mapper and Reducer. These runs on all machines in a cluster of Hadoop. The input and output of Mapper/Reducer functions must be in form of (key, value) pairs [13].

The input (k1, v1) for the function mapper is taken from HDFS and which helps in producing a list of intermediate (k2, v2) pairs. To reduce the communication cost of relocating intermediate outputs of mappers to reducers are applied from optional Combiner function. Mapper's output pairs are locally organized and gathered on same key and which feeds to the combiner to make local amount. The combiner's intermediate output pairs are hobbled and swapped between the machines to group all the pairs with the same key to a single reducer. This would be the single communication step that takes place and which is handled by the Hadoop MapReduce platform. There is no other communication that will be taking place between the mappers and the reducers. The new pairs (k3, v3) are produced when the reducer takes (k2, list (v2)) values as input and make sum of the values in list (v2). [13], [15].

All the parallelization, intermachine communication and fault tolerance are handled by run-time system. Hence the MapReduce is recognized as a simplified programming model. [17].

VI. APRIORI ALGORITHM ON HADOOP MAPREDUCE

There are 2 main tasks to implement an algorithm on MapReduce framework. Firstly we need to design two independent map and reduce functions for the algorithm and we would need to convert the datasets in the form of (key, value) pairs. All the mapper and reducer on different machines are executed in parallel fashion in the MapReduce programming but the final result will be obtained after the completion of reducer. If the algorithm is recursive, then we need to execute multiple segment of map-reduce to get the final result [20].

6.1 Traditional Apriori to MapReduce Based Apriori

Apriori algorithm is an iterative process. There are 2 main components of Apriori algorithm, candidate itemsets generation and frequent itemsets generation. In each scan of database, mapper creates local candidates and the reducer calculates the local count, results in frequent itemsets. On Hadoop, the count distribution parallel version of Apriori is best suited, whereas to instrument data distribution algorithm we need to control the distribution of data and this is automatically controlled by Hadoop. [6].

The first step of the algorithm where it needs to generate frequent 1-itemsets L1. Transactional database is broken into blocks using the HDFS and distribute it to all mappers running on the machines. Each transaction is transformed to (key, value) pairs, where key is the TID and value is the list of items i.e. transaction. Mapper does read one transaction at a time and output (key', value') pairs, where key' stands for each item in transaction and value' represents 1. The combiner job is to combine the pairs with the same key' and makes the local totality of the values for each key'. The combiner's output pairs are shuffled and swapped to create a list of values associated with same key, as (key', list (value'')) pairs. Reducers will take these pairs and calculates the values of respective keys. Reducers output (key', value''') pairs where key' is item and value''' is the support count >= minimum support, of that item [17], [14], [13]. Final frequent 1-itemsets L1 is obtained by merging the output of all reducers.

6.2 Analysis of Various Proposed Implementations of Apriori on MapReduce

In MapReduce framework, the various implementation of Apriori have been proposed since the beginning of MapReduce, introduced by Google. These algorithms are classified into two categories: 1-phase of map-reduce and k-phase of MapReduce [19]. Also, few algorithms will makes use of all the three functions mapper, reducer and combiner while some used only mapper and reducer function.

6.3 Using Functionality of Combiner inside Mapper

In traditional algorithms mapper outputs a (itemset, 1) pair each time of execution, if the itemset is found in the transaction that is assigned to that mapper. F. Kovacs and J. Illes [13] proposed a different way to achieve this. In his algorithm, mapper finally outputs only (itemset, itemset.counter) pairs for one time for each itemset where the itemset.counter is local support of itemset to count. Inside mapper, the itemset.counter is incremented every time the execution loop, if it finds itemset in transaction assigned to mapper. In this way, mapper produces output (itemset, local support) and this is passed to the reducer as (itemset, list (local support)).

Table 1. Comparative Analysis with other Pattern Algorithms

Comparative Analysis with other Pattern Algorithms				
Sl. No.	**Techniques**	**Abstract**	**Pros**	**Cons**
1	Apriori	Apriori property for Pruning and Breadth first search approach. Database is scanned for each time when a candidate itemset is generated. Execution time is considerable as time consumed in scanning the database for each candidate itemset generation. Data Format: *Horizontal* Storage Structure: *Array*.	It uses large itemset property. It is easy to Implement.	Many Candidate Itemsets. Too many passes over database and requires large memory space.
2	RARM	Depth first search on SOTrieIT to generate 1-Itemset & 2-Itemset. The database is scanned	No candidate itemsets are generated. Speeds up the	Difficult in interactive system mining. It is difficult to

Comparative Analysis with other Pattern Algorithms				
Sl. No.	**Techniques**	**Abstract**	**Pros**	**Cons**
		only few times to construct a SOTrieIT Tree structure. Takes less execution time as compared to Apriori Algorithm and FP Growth algorithm. Data Format: *Horizontal*. Storage Structure: *Tree*.	process for generating candidate 1-itemset & 2-Itemset.	use in incremental Mining.
	ECLAT	Intersection of transaction ids to generate candidate itemset and it is a Depth first Search approach. Database is scanned few times (Best case=2). Execution time is less compared to Apriori algorithm Data Format: *Vertical*. Storage Structure: *Array*	This does not need to scan the whole database each time as a candidate Itemset is generated as supports the count information will be obtained from previous Itemset.	It requires the virtual memory to perform the transformation.
	FP-Growth	Divide and conquer method. Database is scanned only two times. Takes less time as compared to Apriori algorithm. Data Format: *Horizontal*. Storage Structure: *Tree (FP Tree).*	Database is scanned only two times. No candidate generation.	FP-Tree is expensive to build and Consumes more memory.
	ASPMS	This is a BSM (Branch Sort Method) using merge sort. Database is scanned only One time and takes less execution time as compared to FP growth algorithm. Data Format: *Horizontal*. Storage Structure: *Tree(ASP Tree)*	Highly suitable for interactive mining and it requires less memory because of its Compression feature in ASPs tree.	

VII. Advantages And Limitations Of MapReduce

MapReduce operation is a flexible, efficient, and simple model that is used for large scale of data for computing different problems in datasets. As other techniques, MapReduce also has advantages and limitations and also its purely depends on, for solving which type of problems using it.

7.1 Advantages

Some of the major advantages of MapReduce are as follows and are not only considered for data mining problems or pattern matching problems but are general for most of the type of data processing problems.

7.1.1 Automatic Parallelization, Fault Tolerance, Data Distribution, and Workload Balance

The MapReduce runtime system makes the execution of mapper and reducer function on a number of machines parallaly. It splits the datasets into a specific number of fixed sized units and replicates with some factor of replication to provide the highest availability and no loss of the data. It shares the tasks between the busy nodes or from the slower nodes to the idle nodes, so that workload balancing can be achieved and throughput also can be increased. MapReduce method provides higher fault-tolerating capacity by re-executing a damaged task without re-executing the other tasks in the dataset. It re-assigns the tasks from unsuccessful nodes to active or idle nodes [15]. The developer or the programmer can give more concentration into algorithm without worrying about such operations.

7.1.2 Reduced Network Bandwidth Consumption

Hadoop replicates datasets across multiple nodes, allowing data to be read from local disks and write single-copy staged data to local disks to save network bandwidth [17].

7.1.3 Combining Computing Power and Distributed Storage

Hadoop provides a combined platform for the distributed storage system and high computing power.

7.1.4 Extremely Scalable

MapReduce enables applications to run parallelly on a large Hadoop cluster having thousand different nodes and process a large scale of data say in terms of petabytes [12].

7.2 Limitations

Even though MapReduce has many advantages, but it also has some limitations which cannot be eliminated. We listed some of the major limitations which are also specific to some other algorithms like Apriori.

7.2.1 It is working on (key, value) Pairs

MapReduce model only works on data structures of the type (key, value) pairs. All input datasets must be converted into such type of structure.

7.2.2 Blocking Operation

The result of a map-reduce phase cannot be said complete without completion of reducer operation. In n-phase of map-reduce operation, transition to the next phase from the previous phase cannot be possible until all reducers operations have finished. Therefore, it cannot work on pipeline parallelism operation [15].

7.2.3 Implicit Data Distribution

The data distribution version of Apriori cannot be implemented on Hadoop because the distribution of data is automatically controlled by Hadoop [20]. As the count distribution version of Apriori does not exchange data and only exchange the count between nodes, therefore only this is suitable for Hadoop [21]

VIII. CONCLUSION

There are different frequent itemsets extraction algorithms available each with their own advantages and limitations. Basically, Apriori and FP Growth are the algorithms used for mining patterns most of the time. But each has their own drawbacks. In Apriori algorithm, database needed to be scanned frequently and this results in generating many candidate keys which intern increases the I/O and synchronization problems in datasets. These problems are prevailed over by FP Growth algorithm, but this algorithm again has the drawback of constriction of trees in in-memory.

The drawbacks of both Apriori and FP-Growth are taken under control by the approach of FIUT algorithm. FIUT scans the database only two times and reduces the search space. In FIUT, we do not need to traverse the entire tree to check leaves to find frequent items or patterns.

Hadoop MapReduce paradigm system uses the hash-based algorithm by introducing some enhancement in Apriori algorithm. It trims out the itemsets and datasets by pruning rare itemsets. So the efficiency can be achieved. The applications of frequent itemset mining can be range from sentiment analysis, web mining, medical data extraction, knowledge-driven business decisions, to find accident patterns, web link analysis, market basket analysis etc.

Compared to other algorithms and methodologies for mining frequent itemsets in a large scale of data, Hadoop MapReduce paradigm has more advantages and most suitable for mining along with new approaches like FIUT algorithm.

IX. REFERENCES

[1] Sourav S. Bhowmick Qiankun Zhao, "Association Rule Mining: A Survey," Nanyang Technological University, Singapore, 2003.

[2] Yaling Xun, Jifu Zhang, and Xiao Qin, "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce", IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 46, No. 3, March 2016.

[3] Shamila Nasreen, Muhammad Awais Azam, Khurram Shehzad, Usman Naeem, Mustansar Ali Ghazanfar, "Frequent Pattern Mining Algorithms for Finding Associated Frequent Patterns for Data Streams: A Survey," The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, EUSPN-2014.

[4] J.R.Jeba, Dr.S.P.Victor, "Comparison of Frequent Item Set Mining Algorithms", International Journal of Computer Science and Information Technologies, Vol. 2 (6) , 2011.

[5] Jiawei Han • Hong Cheng • Dong Xin • Xifeng Yan, "Frequent pattern mining: current status and future Directions, "Data Mining Knowl Discov, vol. 15, no. I, p. 32, 2007.

[6] Iqbal Gondal and Joarder Kamruzzaman Md. Mamunur Rashid, "Mining Associated Sensor Pattern for data stream of wireless networks," in PM2HW2N '13, Spain, 2013, p. 8.

[7] "Data Mining Algorithms In R/Frequent Pattern Mining/The FP-Growth Algorithm" Wikibooks, open books for an open world.

[8] Seema Tribhuvan, Bharti. P. Vasgi, "Parallel Frequent Itemset Mining for Big Datasets using Hadoop-MapReduce Paradigm", International Journal of Advanced Research in Computer and Communication Engineering (ISO 3297:2007) Certified Vol. 6, Issue 6, June 2017

[9] M.A. Azam, Loo J., Naeem, Usman and Khan, S.K.A. and Lasebae, A. and Gemikonakli Azam, "A Framework to Recognise Daily Life Activities with Wireless Proximity and Object Usage Data", In 3rd IEEE International Symposium on Personal, Indoor and Mobile Radio Communication 2012, Sydney, Australia, 2012, p.6.

[10] Imielienskin T. and Swami A. Agrawal R., "Mining Association Rules Between set of items in largedatabases", in Management of Data, 1993, p. 9.

[11] M. H. Dunham, Y. Xiao, L. Gruenwald and Z. Hossain, "A Survey of Association Rules," http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.91.1602.

[12] M. Chen, and P.S. Yu J.S. Park, "An Effective Hash Based Algorithm for Mining Association Rules" in ACM SIGMOD Int'l Conf. Management of Data, May, 1995.

[13] Yahoo! Hadoop Tutorial, http://developer.yahoo.com/hadoop/tutorial/index.html

[14] S. Ghemawat, H. Gobioff and S. Leung, "The Google File System", in ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 29-43, 2003.

[15] K-H. Lee, Y-J. Lee, H. Choi, Y. D. Chung and B. Moon, "Parallel Data Processing with MapReduce: A Survey", in ACM SIGMOD Record, vol. 40, no. 4, pp. 11-20, 2011.

[16] F. Kovacs and J. Illes, "Frequent Itemset Mining on Hadoop" in Proceedings IEEE 9th International Conference on Computational Cybernetics (ICCC), Hungry, 2013, pp. 241-245.

[17] M-Y. Lin, P-Y. Lee and S-C. Hsueh, "Apriori-based Frequent Itemset Mining Algorithms on MapReduce" in Proceedings 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC '12), ACM, New York, 2012, Article 76.

[18] S. Moens, E. Aksehirli and B. Goethals, "Frequent Itemset Mining for Big Data" in Proceedings IEEE International Conference on Big Data, 2013, pp. 111-118.

[19] J. Dean, and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters" in ACM Commun., vol. 51, pp. 107-113, 2008.

[20] K-H. Lee, Y-J. Lee, H. Choi, Y. D. Chung and B. Moon, "Parallel Data Processing with MapReduce: A Survey" in ACM SIGMOD Record, vol. 40, no. 4, pp. 11-20, 2011.

[21] X. Y. Yang, Z. Liu and Y. Fu, "MapReduce as a Programming Model for Association Rules Algorithm on Hadoop" in Proceedings 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), 2010, vol. 99, no. 102, pp. 23-25.