

Parallel Buddy Prima – A Hybrid Parallel Frequent itemset mining algorithm for very large databases

Dr.S.N.Sivanandam¹, Dr.S.Sumathi², Ms.T.Hamsapriya³, Mr.K.Babu⁴
P.S.G College of Technology
Coimbatore, Tamilnadu, India
E-Mail:prish_67@yahoo.co.in

- 1.Professor &Head, Department of Computer Science &Engineering
- 2.Senior Lecturer, Department of Electrical &Electronics Engineering
- 3.Lecturer, Department of Computer Science &Engineering
- 4.Undergraduate Student, Department of Computer Science &Engineering

Abstract

Frequent itemset mining is essential for the discovery of association rules, strong rules, episodes, and minimal keys. This paper describes a Parallel approach for association mining, based on Buddy Prima algorithm, that combines bottom up and top down approach. Apriori algorithm, the widely used association mining technique uses the breadth-first search, bottom up approach. The Apriori algorithm performs well only when the frequent itemsets are short. Algorithms with top down approach are suitable for long frequent itemsets. This Parallel Buddy Prima algorithm combines both bottom-up and top-down approach. The PRIMA representation consumes less memory as each transaction is replaced with the product of the equivalent prime numbers of their items. It reduces the time taken to determine the support count of the Itemset. Candidate distribution technique is adopted to handle large datasets with large itemsets. The performance of this algorithm is compared with the other existing algorithms and the results are tabulated. The proposed algorithm reduces the time and data complexity. Experimental results of this algorithm on Microsoft Anonymous Data show that this parallel approach outperforms the existing algorithms approximately by a factor of two.

Nomenclature: Parallel data mining, Association mining, top-down approach, Candidate distribution.

1.Introduction

The explosive growth in data collection in business and scientific fields has literally forced the need to analyze and mine useful knowledge from it. Data mining refers to the entire process of extracting useful and novel patterns/models from large datasets. It is a process through which interesting and previously unknown patterns and correlations can be extracted automatically from a large database of information. The implicit information within databases, and mainly the interesting association relationships among sets of objects, that lead to association rules, may disclose useful patterns for decision support, financial forecast, marketing policies, medical diagnosis and many other applications. Association rule mining involves detecting items, which tend to occur together in transactions, and the association rules that relate them. Association rule mining has applications in cross-marketing, attached mailing, add-on sales, store layout and customer segmentation based on buying patterns. Mining frequent itemsets is a fundamental and essential operation in data mining applications including discovery of association rule, strong rules, correlations, sequential rules and episodes. Due to the huge size of data and amount of computation involved in data mining, high-performance computing is an essential component for any successful large-scale data mining application.

Association Rule mining finds the set of all subsets of items or attributes that frequently occur in many database records or transactions, and additionally extracts rules on how a subset of

items influences the presence of another subset. Consider $I = \{i_1, i_2 \dots i_m\}$ as a set of items. Let D , the task relevant data, is a set of database transactions where each transaction T is a set of items such that T is a subset of I . Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if A is a subset of T .

An association rule is an implication of the form $A \Rightarrow B$, where A and B are subsets of I and $A \cap B$ is also a subset of I . The rule $A \Rightarrow B$ holds in the transaction set D with support s , where s is the percentage of transactions in D that contain $A \cup B$ (i.e., both A and B). This is the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has *confidence* c in the transaction set D if c is the percentage of transactions in D containing A that also contain B . This is taken to be the conditional probability, $P(B|A)$. That is,

$$\text{Support}(A \cup B) = P(A \cup B) \quad (1)$$

$$\text{Confidence}(A \Rightarrow B) = P(B|A) \quad (2)$$

The definition of a frequent pattern relies on the following considerations. A set of items is referred to as an itemset (pattern). An itemset that contains k items is a k -itemset. The set $\{X, Y\}$ is a 2- itemset. The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known as the frequency or the support count of an itemset. An itemset satisfies *minimum support* if the occurrence frequency of the itemset is greater than or equal to the *minimal support threshold value* defined by the user. The number of transactions required for the itemset to satisfy *minimum support* is therefore referred to as the *minimum support count*. If an itemset satisfies *minimum support*, then it is a *frequent itemset* (*frequent pattern*).

A frequent itemset is called closed if it does not have any superset with the same support. A frequent itemset is said to be maximal if it has no supersets that are frequent. The collection of maximal frequent itemsets is a subset of the collection of closed frequent itemsets, which is a subset of the collection of all frequent itemsets. Maximal frequent itemsets are necessary for generating association rules.

The most common approach to find association rules is to break up the problem into two parts:

1. **Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a pre-determined minimum support count.
2. **Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

The problem of mining association rules is to find all strong association rules. Since it is easy to generate the strong association rules from all frequent itemsets, almost all current studies for association discovery concentrate on *how to find all frequent itemsets efficiently*. Given m items, there are potentially 2^m frequent itemsets. However, only a small fraction of the whole space of itemsets is frequent. Discovering the frequent itemsets requires a lot of computation power, memory and I/O, which can only be provided by parallel computers. As data is increasing in terms of the dimensions (number of items) and size (number of transactions), one of the main attributes needed in an association rule-mining algorithm is scalability, the ability to handle massive data stores. Sequential algorithms cannot provide scalability, in terms of the data dimension, size, or runtime performance, for large databases. Therefore, high-performance parallel and distributed computing can be employed for enhancing the performance, scalability.

Parallel Association mining is needed to find frequent itemsets in a reasonable time. There has been relatively less work in parallel mining of associations. Three different parallelization methods of Apriori on a distributed-memory machine are presented in [2]. The **Count Distribution algorithm** is a straight-forward parallelization of Apriori. In this method, each processor generates the partial support of all candidate itemsets from its local database partition. At the end of each iteration, the global supports are generated by exchanging the partial supports among all the processors. The **Data Distribution algorithm** partitions the candidates into disjoint sets, which are assigned to different processors. However to generate the global support each processor must scan the entire database (its local partition, and all the remote partitions) in all iterations. It thus suffers from huge communication overhead. The **Candidate Distribution algorithm** also partitions the candidates, but it selectively replicates the database, so that each processor proceeds independently. The local portion is scanned once during each iteration. Parallel Buddy Prima algorithm adopts Candidate distribution technique to distribute the candidates among all nodes.

Most of the algorithms for mining the frequent items [1][4][12] are based on bottom-up search approach. In this approach, the search starts from 1 itemsets and extends one level in each pass until all maximal frequent itemsets are found. This approach performs well if the length of the maximal itemset is short. If the maximal itemset is longer, top down search is suitable. For a transaction with a medium sized maximal frequent set, a combination of both these approaches performs well. This algorithm integrates both the bottom up search as well as the top-down search. This algorithm is suitable for itemsets of any size. It uses the top down approach to find the frequent subsets of itemsets. The bottom up approach is used to find the supersets of the frequent itemsets. The support count of the supersets and the subsets are found effectively from the Prima representation. The Prima representation reduces the memory needed for storing the items of the transactions by assigning an equivalent prime number for each item.

2. Prima Representation

This representation uses Prime numbers to represent the items in the Transaction. Each item is assigned an unique Prime number. Each transaction is represented by the product of the corresponding prime numbers of individual items in the transaction. Since the product of the prime numbers is unique, modulo Division of a Transaction's prime product by the prime product of the itemset can check the presence of itemset in the transaction.

- If the remainder is Zero, then the itemset is present in the Transaction.
- If the remainder is Non-zero, then the itemset is not present in the Transaction.

By checking the presence of itemset in transactions using the above method, Support count can be calculated very quickly. Each transaction in the database can be represented in a single number by using Prime representation.

2.1 Illustration-I

Consider a sample set of transactions as shown in Table 1. In Prima representation, every item is assigned a unique prime number as shown in Table 2. Table 3 shows the transaction table with the itemset replaced by the product of the equivalent prime numbers of the itemset.

Table 1 Transactions

Tid	Transaction
T1	1, 2, 3,4
T2	1, 2
T3	3, 4, 5, 6, 7, 8
T4	1, 3, 6, 8
T5	1, 4, 5, 6, 7
T6	2, 8, 9
T7	1, 10, 11, 12, 13
T8	1, 4, 6, 7
T9	1, 3, 5, 7, 8
T10	3, 5,13
T11	1,3,5
T12	1,2,3

Table 2 Equivalent Prime Assignments

Items	1	2	3	4	5	6	7	8	9	10	11	12	13
Prime Number Equivalent	2	3	5	7	11	13	17	19	23	29	31	37	41

Table 3 PRIMA Representation

Tid	Transaction	Transaction Multiple
T1	$2 * 3 * 5 * 7$	210
T2	$2 * 3$	6
T3	$5 * 7 * 11 * 13 * 17 * 19$	1616615
T4	$2 * 5 * 13 * 19$	2470
T5	$2 * 7 * 11 * 13 * 17$	34034
T6	$3 * 19 * 23$	1311
T7	$2 * 29 * 31 * 37 * 41$	2727566
T8	$2 * 7 * 13 * 17$	3094
T9	$2 * 5 * 11 * 17 * 19$	35530
T10	$5 * 11 * 41$	2255
T11	$2*5*11$	110
T12	$2*3*5$	30

Support count of an item or an itemset can be easily determined by performing modulo division operation with the transaction's prime product and the item's prime or the itemset's prime product. If the modulo operation gives a zero remainder, it indicates that the item or itemset is in the transaction. If the remainder is non-zero, it indicates that the item or itemset is not present in the transaction.

In this illustration, support count of itemset $\{3, 5\}$ can be found by performing the modulo division of each transaction's Prime product by the product '55' of item '3's corresponding prime number '5' and item '5's corresponding prime number '11' as shown in Table 4.

Table 4 Support count determination for {3,5}

Tid	Modulo Division	Remainder	Item's Presence
T1	210 mod 55	Non-Zero	No
T2	6 mod 55	Non-Zero	No
T3	1616615 mod 55	0	Yes
T4	2470 mod 55	Non-Zero	No
T5	34034 mod 55	Non-Zero	No
T6	1311 mod 55	Non-Zero	No
T7	2727566 mod 55	Non-Zero	No
T8	3094 mod 55	Non-Zero	No
T9	35530 mod 55	0	Yes
T10	2255 mod 55	0	Yes
T11	110 mod 55	0	Yes
T12	30 mod 55	Non-zero	No

The support count of itemset {3,5} is found to be '4' as the modulo division operation of the four transactions with 55 gives zero remainder and the modulo operation with the other transactions resulted in a Non-zero remainder.

The weakness of this representation is that the product of the prime number is a very large number for a transaction with more number of items. However the product of the primes can be stored in floating point representation.

3. Buddy Prima Algorithm

Buddy Prima algorithm, a frequent itemset mining algorithm, uses hybrid approach to mine frequent itemset efficiently. In the first pass the algorithm scans the data set and computes the support count of all 1-item sets. The infrequent 1-itemsets are removed from further evaluation. Each item is represented by a unique prime number and each transaction is represented by the multiple of the equivalent prime number of the items in the itemset. The maximal length itemset M is found and the support count is found for all the transactions with the same length. The support count is found using the PRIMA representation. The two possibilities are as follows:

- If the support count is greater than the minimal support count, it is treated as the maximal frequent set and the procedure ends.
- If the support count for the itemsets of length M is less than the minimal support count, the subsets of length equal to $N=M/2$ is generated and their support count is determined

This again leads to the following possibilities

- If the support count of the sets of size N is greater than the minimal support count, all possible supersets of size $N + N/2$ are generated and their support count is determined.
- If the support count is less than the minimal support count, the subsets of length equal to $N/2$ is generated and their support count is determined.

This procedure is repeated in the same manner until the maximal frequent itemset is found. Support count is determined using the PRIMA representation throughout this algorithm.

This algorithm requires lot of computation power, memory and I/O for large-scale association mining. To overcome these problems, Parallel Buddy Prima algorithm, a parallel version of Buddy Prima algorithm with Candidate distribution technique is developed. This algorithm provides scalability, in terms of the data dimension, size, or runtime performance, for large databases.

4.Parallel Buddy Prima Algorithm

This algorithm is a parallel implementation of Buddy Prima algorithm in a master-slave architecture using candidate distribution technique. Candidate distribution technique reduces the communication between master and slave nodes.

Candidate distribution technique assigns the candidate itemsets generated from different parts of database to different processors and each processor is assigned disjoint candidates, independent of other processors. At the same time, the database is shared among all processors, so that each processor can generate global counts independently.

ALGORITHM FOR MASTER NODE:

1. Find the infrequent items of length 1 and store them in **IF1**
2. Remove the infrequent 1-items as denoted by **IF1** in all the transactions.
3. Assign separate Prime Number P_i to each unique item IT_i for n-items.
4. Represent the itemsets in **PRIMA Representation** as follows:
 - a) Replace each Transaction's item IT_i by corresponding Prime Number P_i .
 - b) Represent each Transaction T_i of size m by the multiple M_i of all the prime number representation P_i of the items in the transaction ($P_1 \times P_2 \times \dots \times P_m$) and store them in shared memory.
5. Find the size *Maxlen* of maximal size transaction in Database and put it in shared memory.
6. For each node i in the Cluster
 - Divide the transactions equally based on the number of nodes and assign to i -th Node.
 - Connect to i -th node's server program to initiate process.
 - End loop
7. For each node i in the Cluster
 - Wait until result comes from i -th node
 - Show the result from i -th node
 - End loop


```

Else
    End = i
    i = Round( Start + End ) / 2
    If i = Start - 1 then
        Send all Itemset AllFrequentItemset to master
        Exit the Do loop.
    End if
End If
End Do loop

```

Fig 1 Parallel Buddy Prima algorithm

Fig 1 shows the Parallel Buddy Prima algorithm with its functionality divided across its master and slaves.

The Master node prunes the transactions by removing 1-infrequent itemsets and stores the Prime multiple for each transaction in shared memory. It finds the Maximal length transaction size *Maxlen* and puts in shared memory. It divides the transactions equally to each node for candidate generation. Though horizontal partitioning, vertical partitioning and checkerboard partitioning methods can be used to divide and distribute the transactions, horizontal partitioning method is adopted, as it demands minimum communication for this application.

If there are **k** number of slaves and **n** number of transactions, then **n/k** number of transactions are assigned to each slave if **n** is a integral multiple of **k**. Otherwise, **k-1** slaves will be assigned **n/k** transactions and **kth** slave will be assigned (**n/k + mod(n/k)**) transactions. Master connects to each slave node and initiates the process of finding the frequent itemset. Finally, the master node shows the global frequent itemsets after gathering the local frequent itemsets.

After the Master node initiates the slave node, it reads the allotted number of transactions and Maximal length transaction size *Maxlen*. It uses the buddy approach to find the maximal frequent itemset and Prima representation to quickly find support count of an itemset. Then, it returns the frequent itemsets to Master node.

4.1 Illustration-II

For the set of transactions given in Table 1, the master removes the 1-infrequent items {10,11,12} since their support count is less than the minimum support count of 2. The transactions are then represented with their equivalent prime numbers and stored in common memory. The transactions are divided equally and sent to the clients. In this illustration,

```

Total number of transactions n=12
Number of clients k=3
Number of transactions sent to each client=4
Slave node 1 will process candidates from 1 – 4 transactions
Slave node 2 will process candidates from 5 – 8 transactions
Slave node 3 will process candidates from 9 – 12 transactions
Maxlen = 6

```

The transactions after removing the 1 –infrequent items are shown in Table 5. Table 6 illustrates the Prima Representation of the transactions after pruning the 1-infrequent items.

Table 5 Transactions after pruning

Tid	Transaction
T1	1, 2, 3,4
T2	1, 2
T3	3, 4, 5, 6, 7, 8
T4	1, 3, 6, 8
T5	1, 4, 5, 6, 7
T6	2, 8, 9
T7	1, 13
T8	1, 4, 6, 7
T9	1, 3, 5, 7, 8
T10	3, 5,13
T11	1,3,5
T12	1,2,3

Table 6 Prima Representation after pruning

PRIMA Representation		
Tid	Transaction	Transaction Multiple
T1	2 * 3 * 5 * 7	210
T2	2 * 3	6
T3	5 * 7 * 11 * 13 * 17 * 19	1616615
T4	2 * 5 * 13 * 19	2470
T5	2 * 7 * 11 * 13 * 17	34034
T6	3 * 19 * 23	1311
T7	2 * 41	82
T8	2 * 7 * 13 * 17	3094
T9	2 * 5 * 11 * 17 * 19	35530
T10	5 * 11 * 41	2255
T11	2*5*11	110
T12	2*3*5	30

Slave node 1 determines the support count of the candidate itemset of length=Maxlen ie 6. Since the support count of {3,4,5,6,7,8} is 1, $Maxlen = \frac{Maxlen}{2}$
 $= \frac{6}{2} = 3$

It generates the candidate itemsets of length three as {1,2,3} {2,3,4} {3,4,5} {4,5,6} {5,6,7} {6,7,8} {1,3,6} {3,6,8} {1,6,8} {4,7,8} and determines their support count. If no candidate itemset has a support count greater than 2, $Maxlen = \text{round}(Maxlen/2)$, else $Maxlen = \text{round}(Maxlen + Maxlen/2)$. All the nodes proceed in this manner till the maximum frequent itemset is found. Master finally receives all frequent itemset from nodes and displays them.

5.Implementation

Parallel Buddy prima algorithm is implemented in a homogeneous cluster of four 1.50GHz Pentium IV machines with 128MB RAM running Linux OS. The cluster consists of one master and three slave nodes. Normally data distribution time is the most important overhead in parallel architectures. Here, data distribution time is reduced by sharing the files and data among all nodes using NFS (Network File System). This algorithm is tested with Microsoft Anonymous dataset. This dataset was obtained from Microsoft Anonymous Web data. This dataset represents the areas of Microsoft.com visited by 30,000 anonymous, randomly selected users in one-week time frame. Statistics of this dataset is found in Table 7.

Table 7 Statistics of dataset

Name of the Dataset	Total number of transactions	Total number of items	Maximum length transaction
Microsoft Anonymous web dataset	10000	255	26
	10000	251	35
	10000	261	31

Table 8 shows the performance of the Parallel Buddy Prima algorithm over its sequential Buddy Prima algorithm and Apriori algorithm for the Microsoft Anonymous Dataset. The Apriori algorithm used here was an optimized one using trie structure implemented by Bart Goathels [15]. The proposed algorithm outperforms its sequential algorithm and the Apriori algorithm. The experiment was conducted by varying the support counts from 5% to 30%. The computation time of maximal frequent itemset mining of the Parallel Buddy Prima algorithm was less than its sequential implementation and the Apriori algorithm. This algorithm performs better than its sequential approach and the Apriori implementation even for short maximal frequent itemsets.

Table 8 Parallel Buddy Prima Algorithm's Performance

Microsoft Anonymous Data	Algorithm	Time in seconds					
	Minimum support count	30%	25%	20%	15%	10%	5%
1 - 30000 transactions 258 Items	Parallel Buddy Prima	0.4	0.4	0.5	0.6	0.9	1.2
	Buddy Prima	1.7	1.9	1.9	2.3	2.3	2.5
	Apriori	2.5	2.7	2.7	2.8	2.8	2.9

Results for 30000 transactions

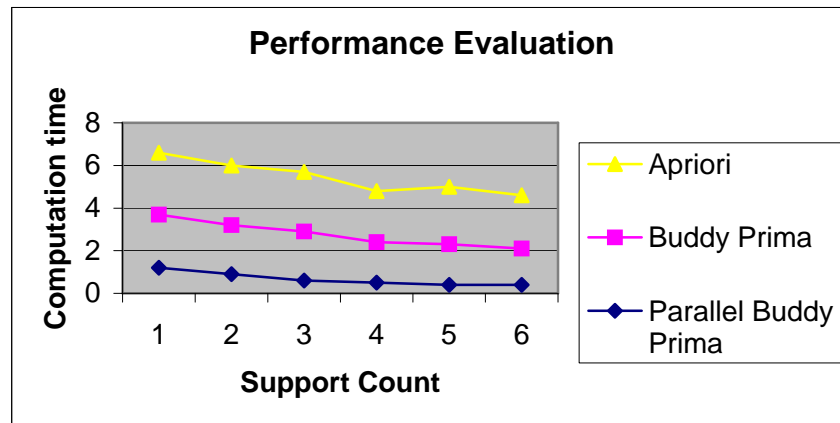


Fig. 2 Performance Evaluation of Parallel Buddy Prima vs. Sequential Buddy Prima algorithm on Microsoft Anonymous data

6. Conclusion

In this paper, a new Parallel hybrid algorithm namely Parallel Buddy Prima algorithm was proposed. The implementation of this algorithm showed that it outperformed its sequential version and the optimized Apriori algorithm by approximately a factor of two for Microsoft anonymous dataset. The innovative Prima representation uses less memory as it stores only one number for each transaction. The computational complexity is reduced as the product of their equivalent prime numbers represents each candidate itemset. The support count of any set is found without any additional scan of the database. The pruning of the infrequent items in the first scan reduces the size of the dataset in the main memory. Communication overhead is greatly reduced due to the candidate distribution technique. Due to its efficient mining approach and low memory requirements, the Parallel Buddy Prima algorithm exhibits high scalability and can be efficiently used to find low support frequent itemsets within the large database characterized by short or long length patterns.

References

- [1] R. Agrawal, T. Imielinski, and R. Srikant. "Mining association rules between sets of items in large databases " SIGMOD, May 1993.
- [2] R. Agrawal and J. C. Shafer. "Parallel Mining of association rules: Design, Implementation and Experience". IBM Research Report RJ10004, Feb. 1996.
- [3] J. Han, Y. Fu. "Discovery of multiple-level association rules from large databases". In 21st VLDB, Sept. 1995.
- [4] R. Agrawal and R. Srikant. "Fast algorithms for mining association rules in large databases", In Proc. 20th VLDB, Sept. 1994.

- [5] Burdick, D., Calimlim, M., Gehrke, J., "MAFIA :A Maximal Frequent Itemset Algorithm for Transactional databases", In Intl. Conf. on Data Engineering 2001.
- [6] Lin D I., Kedem, Z M., "Pincer Search: A new algorithm for discovering the maximum frequent set", Intl Conf. on Extending database technology,1998.
- [7] A. Sarasere, E. Omiecinsky, and S. Navethe. "An efficient algorithms for mining association rules in large databases". In Proc. 21st VLDB, Sept. 1995.
- [8] H. Toivonen. "Discovery of frequent patterns in large data collections". Technical Report A-1996-5 of the Department of Computer Science, University of Helsinki, Finland, 1996.
- [9] J. S. Park, M.-S. Chen and P. S.Yu. "Efficient Parallel Data Mining for Association rules", IBM Research Report, RC 20156, August 1995.
- [10] J. S. Park, M.-S. Chen and P. S.Yu. "An Effective Hash based Algorithm for Association rules". Proceedings of ACM SIGMOD, May, 1995. IBM Research Report, RC 20156, August 1995.
- [11] Zheng, Z., Kohavi, R., and Mason, L. "Real world performance of association rule algorithm" , In Proc. 7-th Int. Conf. on Knowledge Discovery and Data Mining. 2001.
- [12] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Berkamo. "Finding interesting rules from large sets of discovered association rules". In Proc. Third International Conference on Information and Knowledge Management, Nov. 1994.
- [13] A. Mueller. "Fast sequential and parallel algorithms for association rule mining: A comparison". Technical Report No. CS-TR-3515 of CS Department, University of Maryland-College Park.
- [14] R. Srikant and R. Agrawal. "Mining generalized association rules". In 21st VLDB, Sept. 1995.
- [15] Website : www.cs.helsinki.fi/u/goethals/