Weighted Support Association Rule Mining using Closed Itemset Lattices in Parallel

A.M.J. Md. Zubair Rahman and P. Balasubramanie

Kongu Engineering College, Perundurai, Tamilnadu, India

Summary
In this paper, we propose a new algorithm which associates weight to each item in the transaction database based on the significance of the corresponding item. Weighted support is calculated using the weight and the frequency of occurrence of the item in the transactions. This weighted support is used to find the frequent itemsets. We partition the database among 'N' processors and generate closed frequent itemsets in parallel. The parallel algorithm used minimizes communication by exchanging only weighted supports among the processors. We generate closed frequent itemsets to reduce the number of itemsets and also as all frequent itemsets can be obtained from closed frequent itemsets, we are not losing any interesting and significant itemsets. The performance of the proposed algorithm is compared to count distribution algorithm in terms of scaleup, speedup, sizeup and is shown that the proposed algorithm performs better.

Key words:
frequent itemsets, support, confidence, association rules.

1. Introduction

During recent years, one of active research topic is Association rule discovery. This was first introduced in [2]. The association rule discovery is used to identify relationships between items in very large databases, to extract interesting correlations, associations among sets of items in the transaction databases or other data repositories. For example, given a market basket database, it would be interesting for decision support to know the fact that 30% of customers who bought coca powder and sugar also bought butter. This analysis may be used to provide some basis if is required to increase the sales and introduce from free schemes like, if 3 kg of sugar is bought then 100g butter free. In a census database, we should discover that 20% of persons who worked last year earned more than the average income, or in a medical database, that 35% of patients who have cold also have sinus [3]. There are many areas in which association rules are widely used. We list out some of areas such as telecommunication networks, market and risk management, inventory control etc. After the publication of Agrawal, Imielinski and Swami and Agrawal and Srikant papers [2, 1], discovering association rules for the given threshold of minimum support and confidence have become one of the most active research topics.

There are two sub-problems in association rule mining. Finding frequent or large itemsets is the first sub-problem. Frequent itemsets are those itemsets whose frequency of occurrence or support is greater than the minimum support provided by the user in the database. Generating association rules from the frequent itemsets generated in the first step is the second problem. Association rules must satisfy the minimal confidence constraint. Suppose one of the large itemsets is Lk, Lk = {I1, I2, … , Ik}, this itemset can be used to generate the association rules like the following: the first rule is {I1, I2, … , Ik-1}→ {Ik}, this rule can be determined whether it is interesting or not by using the confidence constraint [6]. The different rules are generated from this rule itself by deleting the last items on the left side of the rule and appending it to the right of the rule. The confidence constraint is used now to determine the corresponding rule's interestingness. This process continues till all the items come to the right of the rule. The second sub-problem is simpler when compared to the first sub-problem. Once the frequent items are found, it is a straight forward procedure to generate association rules with the provided support. Hence, the problem of mining association rules is reduced to the problem of finding frequent itemsets.

Frequent items can be generated in two steps. Firstly, candidate large itemsets are generated and secondly frequent itemsets are generated using these candidate itemsets. The itemsets whose support is greater than the minimum support are referred as frequent itemsets. The itemsets that are expected to be large or frequent are termed as candidate itemsets. The drawback in validating the large number of association rules that are generated limits the applications of data mining. There is very vast literature survey done to reduce the number of association rules. Some of these algorithms stated that the association rules can be generated without duplicates or only interesting rules or based on strength [6]. Mostly all algorithms that are proposed to generate association rules are based on Apriori mining method [1]. The performance of such algorithms is good for weakly correlated data as market basket data but is bad for correlated data such as

Manuscript received March 5, 2009
Manuscript revised March 20, 2009

census data. The significance of the attributes in a transaction within the whole item space is considered to be same without its significance in traditional association rule mining. If the association rules are generated in this fashion, some interesting rules are missed. For example, [wine → salmon, 1%, 80%] may be more important than [bread → milk, 3%, 80%] even though the former holds a lower support [7]. This is because those items in the first rule usually come with more profit per unit sale, but the standard ARM simply ignores this difference. The model in [4] also considers only whether an item is present in a transaction, but does not take into account the weight/intensity of an item within a transaction. For example, a customer may purchase 13 bottles of coke and 6 bags of snacks and another may purchase 4 bottles of coke and 1 bag of snacks at a time. The conventional association rule approach treats the above two transactions in the same manner, which could lead to the loss of some vital information. Assume, for example, that if a customer buys more than 7 bottles of coke, he is likely to purchase 3 or more bags of snacks. Otherwise, the purchase tendency of coke is not strong. The traditional association rule cannot express this type of relationship. With this knowledge, the supermarket manager may set a promotion such as if a customer buys 8 bottles of coke; he can get two free bags of snacks. So, weight is associated for the items through which rules are found. Weighted Association Rules cannot only improve the confidence in the rules, but also provide a mechanism to do more effective target marketing by identifying or segmenting customers based on their potential degree of loyalty or volume of purchases [4]. The main challenge of adapting traditional association rule mining model in a weighted setting is the invalidation of the "downward closure property", which is used to justify the efficient iterative process of generating and pruning large itemsets from its subsets.

The organization of the rest of the paper is as follows: Next section gives the work done in this area before; third section describes the proposed work, fourth section shows the performance evaluation and last section concludes the paper.

2. Background

The basic concepts of association rule mining and its preliminaries are discussed in [6] by Sotiris Kotsiantis et., al. They also had survey of the existing association rule mining techniques.

The new and efficient algorithm, Close is proposed by Nicolas Pasquier et., al [3]. This algorithm is based on the pruning the closed set lattice. Closed itemset lattice is a sub-order of the subset lattice and is closely related to Wille's concept lattice in formal concept analysis.

Traditional association rule problem is extended in [4] by Wei Wang et.al. In this algorithm, the intensity of the item in the transaction is considered and a weight attribute is associated with each item based on its intensity. The rule generated from the items associated with weight is referred as weighted association rule (WAR). They also discussed how the confidence of the rules can be improved and effective target marketing can be achieved if the customers are divided based on their potential degree of loyalty or the volume they purchase. In WAR, the frequent itemsets are found by ignoring the weight and the weight is associated during the generation of association rules.

Rakesh Agrawal et., al [1] presented two new algorithms for solving the problem of discovering association rules between items in a large database of sales transactions, that are fundamentally different from the known algorithms. Empirical evaluation shows that these algorithms outperform the known algorithms by factors ranging from three for small problems to more than an order of magnitude for large problems. The best features of the two proposed algorithms can be combined into a hybrid algorithm, called AprioriHybrid.

Feng Tao et., al [7] addressed the issues of discovering significant binary relationships in transaction datasets in a weighted setting. Traditional model of association rule mining is adapted to handle weighted association rule mining problems where each item is allowed to have a weight. The focus is to those significant relationships involving items with significant weights rather than being flooded in the combinatorial explosion of insignificant relationships. The challenge in this approach is using weights in the iterative process of generating large itemsets. The problem of invalidation of the "downward closure property" in the weighted setting is solved by using an improved model of weighted support measurements and exploiting a "weighted downward closure property". A new algorithm called WARM (Weighted Association Rule Mining) is developed based on the improved model. The algorithm is both scalable and efficient in discovering significant relationships in weighted settings as illustrated by experiments performed on simulated datasets.

Rakesh Agrawal et.al., [5] presented three algorithms and explore tradeoffs between computation, communication, memory usage and synchronization. The three algorithms are count distribution, data distribution and candidate distribution. Count distribution algorithm is the motivation of our proposed work. In count distribution algorithm, Agrawal considers only the support of the item to generate frequent itemsets. We consider the significance of the item also and generated closed frequent itemsets. These are the two variations in the proposed work from the count distribution algorithm. By considering the significance of the item, we don't loose important items whose support

may be less than the minimum support provided by the user because of its durability. By generating closed frequent itemsets, the number of itemsets in each level reduces, so that time taken will be reduced and moreover we will not loose any interesting or significant itemsets as all frequent itemset can be generated from closed frequent itemsets.

3. Proposed work

In general, the association rules are generated in two steps. First, frequent itemsets are found and secondly, rules are generated using the frequent itemsets found in first step. Frequent itemsets are also very huge in order to perform any analysis or for generating association rules. Instead, we are generating closed frequent itemsets from which association rules can be formed. Generally, in generating closed frequent itemsets, minimum support is only considered. But if minimum support alone is considered, some interesting / important items whose support < minimum support are lost. So, we consider a special attribute referred as weight which is associated with each item and has a value based on its durability / expiry / significance. For example, the lifetime of bread, cheese is less when compared to the lifetime of wheat, rice etc in market basket database. The quantity of purchase also can be considered as a weight. To improve the performance, parallel algorithm – count distribution is used to generate closed frequent itemsets and rules are also generated in parallel. The focus of the count distribution algorithm is on minimizing communication. It does so at the expense of carrying out redundant computations in parallel. The principle of allowing "redundant computations in parallel on otherwise idle processors to avoid communication" is followed in this algorithm.

3.1 Proposed algorithm

3.1.1 Generating Closed Frequent itemsets

1. Minimum weighted support is provided by the user.
2. Weight is associated with each item in the transaction database.
3. The database is distributed among 'N' different processors.
4. In first pass, each processor scans its local database and generates local candidate itemset.
5. Support of each item is found based on its frequency in the local database of a particular processor.
6. Weighted support of each item in the database is calculated as weight * support.
7. The weighted supports (count) of all items are normalized to fix in some defined range of values.
8. The local counts are communicated among all the processors to develop global counts.
    a. Local candidate set of each processor P_i is maintained in a closed hash-table. For each tuple, every item is hashed and its corresponding count in the hash table is incremented; new entries are created if necessary.
    b. At the end of the pass, processor loads items and their counts from the hash table into a send buffer ItemsOfProcI and then gathers items and their support counts from all other processors.
    c. To do this, it must first gather the count of the total number of items residing in the send buffers of all other processors.
    d. Processor P_i puts the count of its own items in a CountBuf and calls AllGather(SendBuf=CountBuf, ReceiveBuf=CountArr, BlockLen=sizeof(integer)). The jth element of the CountArr now contains the number of items processor j has in its send buffer.
    e. Next, processor P_i calls AllGatherV( ) to collect all items and their counts into the receive buffer AllItems
    f. AllGatherV(SendBuf=ItemsOfProcI, ReceiveBuf=AllItems, BlockLen= sizeof(ItemsOfProcI), ReceiveBlockLen= CountArr)
    g. AllGatherV() is the variable length counterpart of AllGather() in which a processor receives messages of different sizes from other processors. SendBuf is of size BlockLen, ReceiveBuf is an array of N messages, and the size of the i^{th} receive buffer is given by the i^{th} element of the ReceiveBlockLen array. If AllItems array becomes too large, we have an intermediate step using ReduceScatter() to reduce the number of duplicate entries. We omit this detail for brevity.
    h. P_i now hashes items from the receive buffer into a new hash table. If the same item was counted by more than one processor, it will hash to the same bucket and the support count for this

item is accumulated. Thus, P_i now has the entire candidate set C_1, complete with global counts.

9. These global counts are used to generate frequent itemsets L_k.
10. Closed frequent itemsets CL_k are found from these frequent itemsets by removing every frequent itemset that is a proper subset of, and carries the same support as, an existing frequent itemset.
11. When k >1 then
12. Each processor P_i generates the complete candidate set C_k using the closed frequent itemset CL_{k-1} created at the end of pass k – 1. Now the candidate set is identical in all the processors because of identical CL_{k-1}.
13. Local counts (weighted supports) for the candidate set are developed by making a pass over its local database.
14. Weighted Support of an itemset is calculated as the average weight of all the items in the set (sum of weights of all items/number of items in the set)
15. All processors exchange local counts with the other to generate global counts. Here synchronization of processors is forced.
    a. The candidates are kept in a hash-tree to allow efficient counting when making a pass over the data.
    b. To exchange local counts, each processor asynchronously extracts its local counts of candidate sets into a count array LCntArr.
    c. Since candidate set C_k is identical for all processors, if every processor traverses C_k in exactly the same order, corresponding elements of the count arrays will correspond to identical candidate itemsets.
    d. Thus we do not have to communicate itemsets themselves but only their counts. We also save on computation because we can sum these local counts using simple vector summation rather than having to compare and match candidates.
    e. Having created LCntArr, processors now do ReduceScatter() communication to perform a partitioned vector-sum of the count arrays. ReduceScatter(SendBuff=LCntArr, ReceiveBuf=PartGCntArr, BlockLen= PartSize, ReductionFunction=add)
    f. As the result of this operation, processor P_i receives the global counts of all the items in the i^{th} LCntArr partition of all the processors.
    g. The number of items in each partition, PartSize, will be sizeof(LCntArr)/N.
    h. Each processor now gathers into GCntArr the global counts of items belonging to all other partitions by calling AllGather(SendBuf=PartGCnt, ReceiveBuf=GCntArr, BlockLen=PartSize).
    i. Thus giving each processor the global counts for all candidates in C_k.
16. Each processor P_i now computes L_k from C_k.
17. Then closed frequent itemset CL_k is generated from L_k by each processor.
18. The decision to terminate or continue to the next pass will be taken independently by each processor. The decision will be identical as the processors all have identical frequent itemsets.

Thus the processors scan their local data asynchronously in parallel in every pass. At the end of each pass, they are synchronized to develop global counts.

3.1.2 Association Rule Generation in Parallel

We now present our parallel implementation of the second subproblem – the problem of generating rules from closed frequent itemsets. Generating rules is much less expensive than discovering closed frequent itemsets as it does not require examination of the data. Given a closed frequent itemset L, rule generation examines each non-empty subset a and generates the rule $a \Rightarrow (L - a)$ with support = support(L) and confidence = support(L)/support(a). This computation can efficiently be done by examining the largest subsets of L first and only proceeding to smaller subsets if the generated rules have the required minimum confidence [4]. For example, given a closed frequent itemset ABCD, if the rule $ABC \Rightarrow D$ does not have minimum confidence, neither will $AB \Rightarrow CD$, and so we need not consider it.

Generating rules in parallel simply involves partitioning the set of all closed frequent itemsets among the processors. Each processor then generates rules for its partition only using the algorithm above. Since the number of rules that can be generated from an itemset is sensitive to the itemsets size, we attempt equitable balancing by partitioning the itemsets of each length equally across the processors. Note that in the calculation of the confidence of a rule, a processor may need to examine the weighted support of an itemset for which it is not responsible. For this reason, each processor must have access to all the closed frequent itemsets before rule generation can begin. This is not a problem because all the processors have all the closed frequent itemsets at the end of the last pass.

4. Performance Evaluation

Synthetic datasets of varying complexity are used for experiments. The datasets on which the experiments are carried out is D1456K.T15.I4. Experiments were repeated many times to obtain stable values at each data point. The characteristics like scaleup, sizeup and speedup are examined and compared with existing count distribution algorithm. The results clearly indicate that the proposed algorithm performs better when compared to the count distribution algorithm.

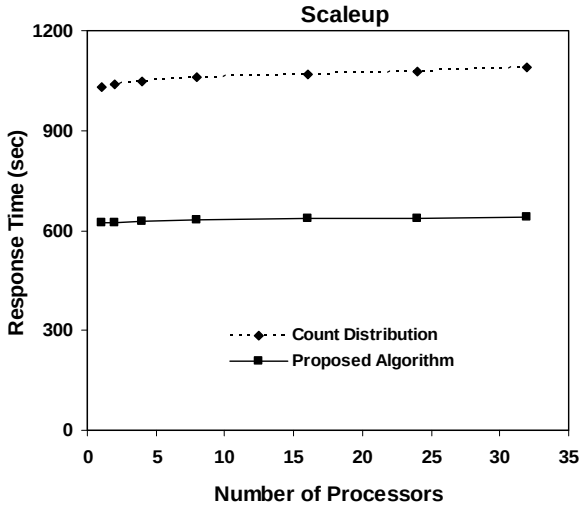

Fig. 1 Comparative Performance of Count Distribution and the proposed algorithm in terms of absolute response time with varied number of processors and varied database sizes

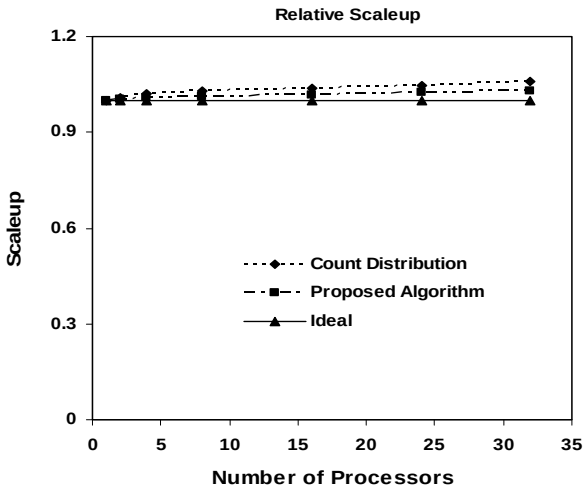

Fig. 2 Comparative Performance of Count Distribution and the proposed algorithm in terms of Scaleup with varied number of processors and varied database sizes

Scaleup experiments were done in order to test how the proposed algorithm handles the larger datasets when more processors are available. The results shown in Fig. 1 indicate clearly that the proposed algorithm performs well by maintaining response time almost constant as the database and the number of processors increase. We have also shown the results in terms of scaleup which is the response time normalized with respect to the response time for a single processor in Fig. 2.

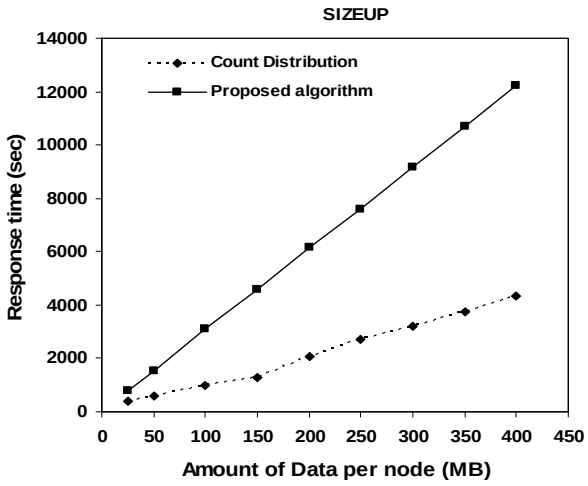

Fig. 3 Comparative Performance of Count Distribution and the proposed algorithm in terms of absolute response time with varied database sizes and 16 nodes.

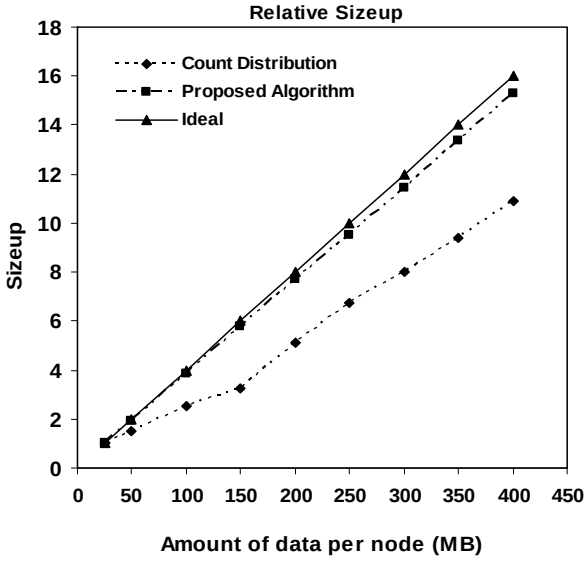

Fig. 4 Comparative Performance of Count Distribution and the proposed algorithm in terms of sizeup with 16 processors and varied database sizes

The experiments in terms of sizeup are done by fixing the size of the multiprocessor at 32 nodes while growing the database from 25MB per node to 400 MB per node and results are shown in Fig. 3. The sizeup is the response time normalized with respect to the response time for 25MB per node. The results show that algorithm performs better as the database size is increased. The performance in terms of sizeup is shown in Fig. 4.

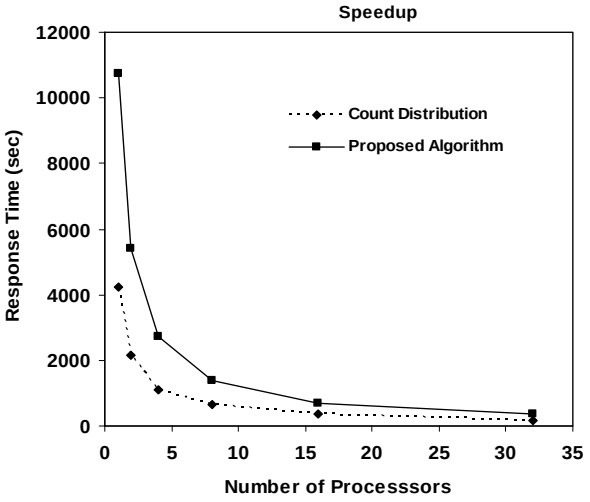

Fig. 5 Comparative Performance of Count Distribution and the proposed algorithm in terms of absolute response time with varied number of processors and constant database size of 400MB

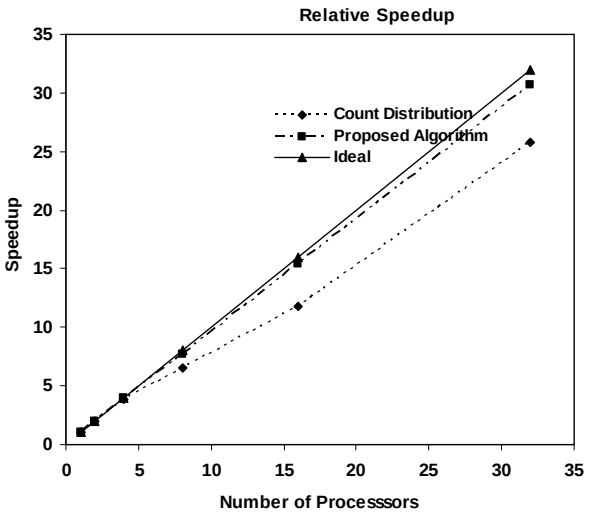

Fig. 6 Comparative Performance of Count Distribution and the proposed algorithm in terms of speedup with varied number of processors and constant database size of 400MB

The speedup experiments are done by keeping the database as constant and varying the number of processors and results are shown in Fig. 5. The database size is fixed at 400MB. The speed up is the response time normalized with respect to the response time for a single processor and the performance is shown in Fig. 6. We noticed from the experiments that the more data a node processes, the less significant becomes the communication time giving us better performance.

5. Conclusions

As there are wide applications of association rules in data mining, it is important to provide good performance. In view of this, we proposed a new algorithm which considers the significance of the item also but not only support of the item. The database is partitioned across different processors to generate the frequent itemsets in parallel. Moreover, we have concentrated in generating closed frequent itemsets instead of frequent itemsets because the number of closed frequent itemsets will be smaller than the number of frequent itemsets. As, all frequent itemsets can be obtained from the closed frequent itemsets, it is enough to find the closed frequent itemsets. Similar to the count distribution algorithm, the proposed algorithm exchanges only weighted supports of the items among the processors and minimizes communication. The performance of the proposed algorithm is compared with the existing count distribution algorithm in terms of scaleup, sizeup and speedup and is shown that the proposed algorithm exhibits better performance.

References

[1] Rakesh Agrawal and Ramakrishnana Srikant, "Fast Algorithms for Mining Association Rules", in Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

[2] Agrawal R, Imielinski T and Swami A.N, "Mining Association rules between sets of items in large databases", in proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216.

[3] N. Pasquier, Y.Bastide, R.Taouil, and L.Lakhal, "Efficient Mining of association rules using closed itemset lattices", Information systems, Vol. 24, No. 1, 1999, pp. 25–46.

[4] W.Wang, J.Yang and P.Yu, "Efficient mining of weighted association rules (WAR)", Proc. of the ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, 270-274, 2000.

[5] Agrawal R., Shafer J, "Parallel Mining of Association Rules", in IEEE Knowledge and Data Engineering, 8(6):962-969, 1996.

[6] Sotiris Kotsiantis, Dimitris Kanellopoulos, "Association Rules Mining: A Recent Overview", in

GESTS Int'l Transactions on Computer Science and Engineering, Vol. 32, 2006, pp. 71-82

[7] Feng Tao, Fionn Murtagh, Mohsen Farid, " Weighted Association Rule Mining using Weighted Support and Significance Framework", SIGKDD 2003.

Prof A.M.J.Md.Zubair Rahman has completed his M.S.,(Software Systems) in BITS Pilani , India in 1995. Then he completed M.E.,(Computer Science & Engineering) in Bharathiar Univeristy , Tamilnadu , India in 2002 . Now he is doing research in the field of Association Rule Mining algorithms . Currently, he is working as Assistant Professor in the Department of Computer Science & Engineering , Kongu Engineering College , Tamil Nadu, India. He has completed 18 years of teaching service. He has published 15 articles in International / National journals.

Dr.P.Balasubramanie has completed his P.hD., degree in Theoretical Computer Science in 1996 in Anna University. He was awarded Junior Research Fellow in the year 1990 by CSIR. Currently he is a professor in the department of Computer science & Engineering in Kongu Engineering College, Tamilnadu, India. He has completed 14 years of teaching service. He has published more than 50 articles in International / National Journals. He has authored six books with reputed publishers. He has guided 3 P.hD., scholars and guiding 20 research scholars. He is a referee for ACCST Research Journal, Science Direct Journal and so on.