International Journal of Computer and Electrical Engineering, Vol. 2, No. 4, August, 2010
1793-8163

Decision Tree Induction: An Approach for Data Classification Using AVL-Tree

Devi Prasad Bhukya[1] and S. Ramachandram[2]

***Abstract*— Classification is considered to be one of the important building blocks in data mining problem. The major issues concerning data mining in large databases are efficiency and scalability. This paper addresses these issues by proposing a data classification method using AVL trees, which enhances the quality and stability. Researchers from various disciplines such as statistics, machine learning, pattern recognition, and data mining considered the issue of building a decision tree from the available data. Specifically, we consider a scenario in which we apply the multi level mining method on the data set and show how the proposed approach tend to give the efficient multiple level classifications of large amounts of data. The results specify the improved performance of the proposed algorithm which acquires designing rules from the knowledge database.**

***Index Terms*— Decision Tree Induction, Generalization, Data Classification, Multi Level mining, Balanced Decision Tree Construction.**

I. Introduction

Data Mining is an automated extraction of hidden predictive information from databases and it allows users to analyze large databases to solve business decision problems. Data mining is most useful in an exploratory analysis scenario in which there are no predetermined notions about what will constitute an "interesting" outcome. Data classification, an important task of data mining, is the process of finding the common properties among a set of objects in a database and classifies them into different classes. Decision Trees are widely used in classification [5]. A decision tree method chooses an attribute, which maximizes certain and fixes the time. Then values of the attribute are split into several branches recursively, until the termination is reached. The efficiency of the existing decision tree algorithms, such as ID3 [5], C4.5 [6] and CART [3], has been established for relatively small data sets [7]. These algorithms have the restriction that the training tuples should reside in the main memory, thus limits the scalability and efficiency. The induction of decision trees from very large training sets has been previously addressed by the SLIQ [9] and SPRINT [10] decision tree algorithms. However, the data stored in databases without generalization is usually at the primitive concept level which is including continuous values for numerical attributes. Classification model construction process performed on huge data, so most decision tree algorithms may result in very bushy or meaningless results. In the worst case, the model cannot be constructed if the size of the data set is too large for the algorithms to handle. Hence, we address this issue by using an approach, [4] consisting of three steps: 1) *attribute-oriented induction,* [11] where the low-level data is generalized to high-level data using the concept hierarchies, 2) *relevance analysis,* [12] and 3) *multi-level mining,* where decision trees can be induced at different levels of abstraction. The integration of these steps leads to efficient, high quality and the elegant handling of continuous noisy data. An inherent weakness of C4.5 [6] is that the information gain attribute selection criterion has a tendency to favor multi valued attributes. By creating a branch for each decision attribute value, C4.5 encounters the over-branching problem caused by unnecessary partitioning of the data. Therefore, we propose an algorithm called *Node_Merge*, which allows merging of nodes in the tree thereby discouraging over- partitioning of the data. This algorithm also uses the concept of Height-Balancing in the tree using AVL trees depending on the priority checks for every node. This enhances the overall performance, as the final decision tree constructed is efficient enough to derive the classification rules effectively. This paper is organized as follows. Section 2 describes about the classification using Decision Tree Induction, Section 3 presents the Decision Tree Construction using the proposed approach. Section 4 illustrates the use of the proposed Decision Tree method compares the results of this method with those of other classification techniques. We conclude our study in Section 5 and discuss the possible extensions based on our current work.

Manuscript received January on 8, 2010.

[1] Devi Prasad Bhukya is with the Osmania University, Hyderabad, Andhra Pradesh, India; phone: +91-4027097577; fax: +91 (40) 27095179; e-mail: deviprasad@osmania.ac.in.

[2] Prof. S Ramachandram. is with the Osmania University, Hyderabad, Andhra Pradesh, India, phone: +91-4027097577; fax: +91 (40) 27095179; e-mail: schandram@osmania.ac.in.

II. Classification Using Decision Tree Induction

We address efficiency and scalability issues regarding the data mining of large databases by using a technique which is composed of the following three steps [4].

1) *Generalization by attribute-oriented induction,* to compress the training data. This includes storage of the generalized data in a multidimensional data cube to allow fast accessing,
2) *Relevance analysis,* to remove irrelevant data attributes, thereby further compacting the training data,
3) *Multilevel mining*, which combines the induction of decision trees with knowledge in concept hierarchies. This section describes each step in detail.

A. Attribute-Oriented Induction (AOI)

Attribute-oriented induction [11], a knowledge discovery tool which allows generalization of data, offers two major advantages for the mining of large databases. First, it allows the raw data to be handled at higher conceptual levels. Generalization is performed with the use of attribute concept hierarchies, where the leaves of a given attribute's concept hierarchy correspond to the attribute's values in the data (referred to as primitive level data) [7]. Generalization of the training data is achieved by replacing primitive level data by higher level concepts. Hence, attribute-oriented induction allows the user to view the data at more meaningful abstractions. Furthermore, attribute-oriented induction [11] addresses the scalability issue by compressing the training data. The generalized training data will be much more compact than the original training set, and hence, will involve fewer input/output operations. With the help of AOI, many-valued attributes in the selection of determinant attributes are avoided since AOI can reduce large number of attribute values to small set of distinct values according to the specified thresholds. Attribute-oriented induction also performs generalization by *attribute removal* [11]. In this technique, an attribute having a large number of distinct values is removed if there is no higher level concept for it. Attribute removal further compacts the training data and reduces the bushiness of the resulting trees. In addition to allowing the substantial reduction in size of the training set, concept hierarchies allow the representation of data in the user's vocabulary. Hence, aside from increasing efficiency, attribute-oriented induction may result in classification trees that are more understandable, smaller, and therefore easier to interpret than trees obtained from methods operating on ungeneralized (larger) sets of low-level data. The degree of generalization is controlled by an empirically set *generalization threshold*. If the number of distinct values of an attribute is less than or equal to this threshold, then further generalization of the attribute is halted.

We consider a simple example to explain all the detail steps to generalize the final classification tree and find out the classification rules. The following table 1 depicts a raw training data of a class of average education level in relation with the family's income and the country they live in around the world. We illustrate the ideas of attribute-oriented induction with an example using the described database.

TABLE 1: TRAINING SET EXAMPLE DATA

Average Education level	**Region**	**Family income per year**
Illiterate	Cuba. north	$ 789
4 years college	USA .east	$ 40000
4 years college	USA south	$ 36000
4 years college	USA.middlewest	$ 34000
2 years college	USA.middle	$ 40400
Graduate school	Swiss.south	$ 38979
Element school	Lao.north	$ 354
High school	India.capital	$ 7639
4 years college	……	……
Graduate school	……	……
Junior High	……	……
2 years college	……	……
4 years college	……	……
Graduate school	……	……
Ph. D	……	……
Illiterate	……	……
2 years college	……	……
Illiterate	Angle	$ 93

The Generalization using attribute-oriented induction [11] for the attributes family income and region is as follows:

{Lower income: < 100};
{Low income: < 1000};
{Average income: < 10000};
{Good income: < 20000};
{Better income: < 40000};

And for region, we combine all the regions, which belong to the same country. For example: we can combine USA.east, USA.west, USA.middlewest and USA.middle into the USA. Then we can combine all the attributes with USA and with different income and add an extra field count to this generalized record. But the record for USA.middle has different output class value. We still can generalize it into USA intermediate record as long as the large percent of the class output inside of this class is belong to a same type. The result of the generalized data is stored in a multi-dimensional data cube [2]. For the current example, we only have two arrays, so we can draw a two-dimensional graph here. The final generalized data after attribute oriented induction is like:

TABLE 2: FINAL GENERALIZED TABLE DATA AFTER AOI

Average Education level	**Region**	**Family income per year**	**count**
Illiterate	Cuba	$ 899	2
4 years college	USA	$ 30000	4
Graduate school	Swiss	$ 38000	3
……	……	……	…….

B. Relevance Analysis

The uncertainty coefficient U(A) [4] for attribute A is used to further reduce the size of the generalized training data. U(A) is obtained by normalizing the information gain of A so that U(A) ranges from 0 (meaning statistical independence between A and the classifying attribute) to 1 (strongest degree of relevance between the two attributes). The user has the option of retaining either the n most relevant attributes or all attributes whose uncertainty coefficient value is greater than a pre-specified uncertainty threshold, where n and the threshold are user-defined. Note that it is much more efficient to apply the relevance analysis [12] to the generalized data rather than to the original training data.

$$U(A) = \frac{I(p_1, p_2, \ldots, p_m) - E(A)}{I(p_1, p_2, \ldots, p_m)}$$

Where: $I(p_1, p_2, \ldots, p_m) = -\sum_{i=1}^{m} p_i p \log_2 p_i / p$

$$E(A) = \sum_{j=1}^{k} \frac{p_{1j} + \cdots + p_{mj}}{p} I(p_{1j}, \ldots, p_{mj})$$

Here P is the set of the final generalized training data,

where P contains m distinct values defining with the output distinct output class P_i (for i = 1, 2, 3,…,m) and P contains p_i samples for each P_i, then the expected information needed to classify a given sample is I (p1, p2,…, pm).

For example: we have the attribute A with the generalized final value $\{a_1, a_2, a_3, ..., a_k\}$ can be partition P into $\{C_1, C_2, C_3, ..., C_k\}$, where C_j contain those samples in C that have value a_j of A. The expected information based on partitioning by A is given by E(A) [4] equation, which is the average of the expected information. The gain (A) is the difference of the two calculations. If the uncertainty coefficient for attribute A is 0, it means no matter how we partition the attribute A, we won't get lose information. So the attributes A has no effect on the building of the final decision tree. If U (A) is 1, it means that we can use this attribute to classify the final decision tree. This is similar to find the max goodness in the class to find which attribute we can use to classify the final decision tree. After the relevance analysis, we can get rid of some attribute and further compact the training data.

C. Multi Level Mining

The third and the final step of our method is multilevel mining. This combines decision tree induction of the generalized data obtained in steps 1 and 2 (Attribute-oriented induction and relevance analysis) with knowledge in the concept hierarchies. The induction of decision trees is done at different levels of abstraction by employing the knowledge stored in the concept hierarchies. Furthermore, once a decision tree has been derived [4], the concept hierarchies can be used to generalize individual nodes in the tree and can reclassify data for the newly specified abstraction level. The main idea of this paper is to construct a decision tree based on these proposed steps and prune it accordingly. The basic Decision Tree Construction Algorithm 1 is shown in section 3, which constructs a decision tree for the given training data. Apart from *generalization threshold*, we also use two other thresholds for improving the efficiency namely, *exception threshold (€)* and *classification threshold (κ)*. Because of the recursive partitioning, some resulting data subsets may become so small that partitioning them further would have no statistically significant basis. These "insignificant" data subsets are statistically determined by the *exception threshold*. If the portion of samples in a given subset is less than the threshold, further partitioning of the subset is halted. Instead, a leaf node is created which stores the subset and class distribution of the subset samples.

Moreover, owing to the large amount, and wide diversity, of data in large databases, it may not be reasonable to assume that each leaf node will contain samples belonging to a common class. This problem is addressed by employing a *classification threshold*, κ. Further partitioning of the data subset at a given node is terminated if the percentage of samples belonging to any given class at that node exceeds the classification threshold.

III. Decision Tree Construction

The proposed Decision Tree Construction Algorithm 1 integrates attribute-oriented induction and relevance analysis with a slightly modified version of the C4.5 Decision Tree algorithm. The splitting-criterion in the algorithm 1 deals with both the threshold constraints and also information gain calculation for the data. In this process, the candidate with maximum information gain is selected as "test" attribute and is partitioned. The conditions, whether the frequency of the majority class in a given subset is greater than the classification threshold, or whether the percentage of training objects represented by the subset is less than the exception threshold, are used to terminate classification. Otherwise further classification will be performed recursively. The algorithm operates on training data that have been generalized to an intermediate level by attribute-oriented induction, and for which unimportant attributes have been removed by relevance analysis. In this way, the tree is first fully grown based on these conditions. Then under pruning process, we propose two other algorithms namely, *Node_Merge* and *BalanceHeight* procedures, which helps in enhancing the efficiency in case of dynamic pruning. The following algorithm 1 shows the procedure for the proposed Decision Tree Construction algorithm.

```
Algorithm 1: Decision Tree Construction
DecisionTree (Node n, DataPartition D)
{
    Apply AOI-Method to D to find
            splitting-criterion of node n
    Let k be the number of children of n
    if k>O do
        Create k children c1, c2,..., ck of n
        Use splitting-criterion to partition D into D1,
                D2..., Dk
         for i = 1 to k do
            DecisionTree(ci, Di)
        end for
    endif
    Assign priority to the nodes based on the level;
}
```

Fig 1: Decision Tree Construction Algorithm

As mentioned above, the tree is constructed using the data collected from the relevant set obtained from the first 2 steps i.e. AOI and relevance analysis. Considering the above mentioned training data, the decision tree constructed using the algorithm 1 is depicted in the following figure 2.

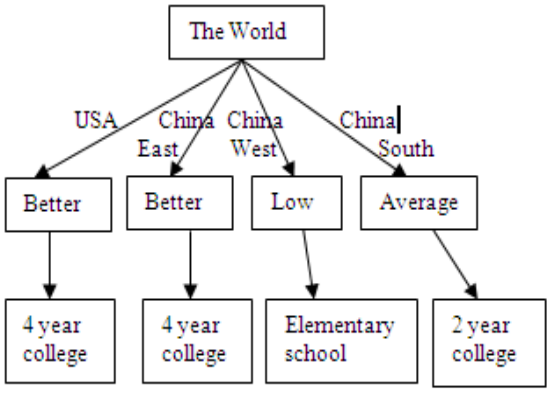

Fig 2: Decision Tree Constructed for the above Training Data using Algorithm 1.

From figure 2, the root node is the whole world's education level. The tree starts as a single node containing the training samples. Then the splitting criterion, which includes the threshold values check namely, *exception threshold* and *classification threshold*, is applied recursively to grow the tree, eventually priority is also assigned for every node based on its level in the tree. Then in the next level, we can see that there exists similar data for regions in China's education level. Hence we can merge these nodes into one. Merging of nodes is applicable, if some child nodes at a selected concept level share the same parent, and largely belong to the same class. This adjustment of concept levels for each attribute is an effort to enhance classification quality. The following algorithm 2 outlines the procedure to node merging in the tree.

```
Algorithm 2: Node Merge in Decision Tree
Node_Merge( NodeData_A, NodeData_B)
{
    Check priorities for node_A and node_B;
     if both the priorities > checkpoint then
      {
        link_AB = remove_link_joining(NodeData_A,
                                      NodeData_B);
       union = NodeData_A.merge_with(NodeData_B);
       for (related_node: nodes_incident_to_either
                        (NodeData_A, NodeData_B))
            link_RA = link_joining (related_node,
                                    NodeData_A);
            link_RB = link_joining (related_node,
                                    NodeData_B);
            disjoin (related_node, NodeData_A);
            disjoin (related_node, NodeData_B);
            join (related_node, union, merged_link);
      }
    else print (" Node have high priority, cannot be
                        merged");
    BalanceHeight (union, new_link_AB);
}
```

Fig 3: Node Merge Algorithm in Decision Tree

As mentioned above in algorithm 2, each time the merging of nodes is implemented the priority check is performed. A checkpoint is assigned based on the training data available. This checkpoint condition helps in checking the priority levels of the nodes i.e., if the priority of the nodes is greater than the checkpoint, it is assumed that the nodes have less priority and the merging of nodes is performed else it is skipped. The following figure 4 shows the decision tree constructed using the above mentioned algorithm 2.

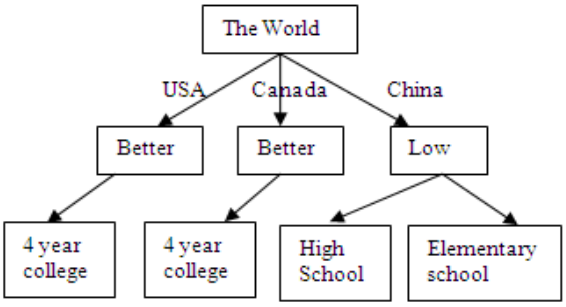

Fig 4: Decision Tree Constructed using the Algorithm 2

The above figure 4 depicts that all the nodes for region China are merged into one using the algorithm *Node_Merge*. Most operations on a Decision Tree take time directly proportional to the height of the tree, so it is desirable to keep the height small [1]. Usually, the primary disadvantage of ordinary tree is that they can attain very large heights in rather ordinary situations, such as when the new nodes are inserted. If we know all the data ahead of time, we can keep the height small on average by performing transformations on the tree. The procedure for height balance using AVL tree concept is outlined below.

```
Algorithm 3: Height-Balancing using AVL Tree
Concept
BalanceHeight (union, link_AB)
{
     Check whether the tree is imbalanced or not;
     if yes then
     {  if balance_factor ( R ) is heavy
     {
        if tree's right subtree is left heavy then
       perform double left rotation;
        else
       perform single left rotation;
     }
        else if balance_factor( L ) is heavy
             {
         if tree's left subtree is right heavy then
         perform double right rotation;
       else
                    perform single right rotation;
             }
     }
      print (" Tree is balanced  ");
     Check for path preservations;
     Generate Classification Rules;
}
```

Fig 5: Height-Balancing using AVL Tree Concept

The final Decision Tree is constructed using the above mentioned algorithm 3, *BalanceHeight*. From the figure 6, it is clear that the tree is well constructed and also balanced at every node. Considering the node low from figure 2, it has 2 children namely, high school and elementary school. After applying the algorithm BalanceHeight at this node, the sub tree is rotated to its right and the resulting tree is shown in figure 3. Here, besides single rotation to right operation, the

concept of *attribute removal* is also applied. In this way, the node elementary school is removed from the tree and is made balanced.

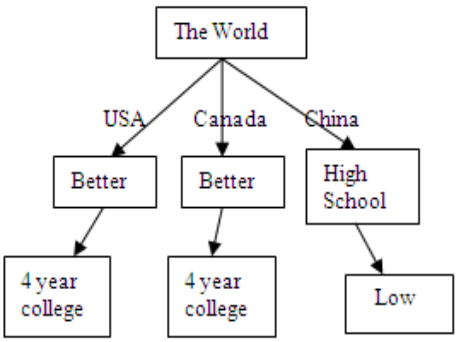

Fig 6: Final Decision Tree Constructed using the Algorithm 3

As mentioned in the algorithm, the path to different levels are updated and preserved accordingly. In this way, the enhanced Decision Tree is developed with the notion of improving the efficiency and scalability of the data classification.

IV. Performance Metrics

This section presents a brief overview of data mining algorithms and compares them to the proposed approach for Data Classification. Recall that prior to Decision Tree Induction, the task relevant data can be generalized to different levels, say *minimally generalized concept level*, based on the attribute generalization thresholds described above. A disadvantage of this approach is that, if each database attribute has many discrete values, the decision tree induced is likely to be quite bushy and large. On the other extreme, generalization may proceed to very high concept levels. Since the resulting relation will be rather small, subsequent decision tree induction will be more efficient than that obtained from minimally generalized data. However, by over-generalizing, the classification process may lose the ability to distinguish interesting classes or subclasses, and thus may not be able to construct meaningful decision trees. Hence, a trade-off, that we adopt, is to generalize to an intermediate concept level.

The previous best and latest algorithms identified by the IEEE International Conference on Data Mining (ICDM) 2006 are C4.5, k-means, SVM, Apriori, These algorithms [8] are among the most influential data mining algorithms in the research community. The basic C4.5 algorithm, encounters the over-branching problem caused by unnecessary partitioning of the data. This problem is addressed by the proposed algorithm called Node_Merge, which allows merging of nodes, thereby discouraging over partitioning of the data. Hence, it is clear that the proposed approach is an improvised version of all the currently best-known algorithms in data mining. The following table 3 presents the comparison between the working of existing algorithms with the proposed approach.

The Decision Tree Induction algorithm mentioned above is based on C4.5 [6] (an earlier version of which is known as ID3 [5]). To deal with pre-pruning and post pruning problem effectively, the proposed approach uses the concepts of dynamic pruning, thresholds in the splitting criterion. Furthermore, the concepts of height balancing using AVL trees, assigning priorities and path preservation are also discussed in the paper. These help in improving and enhancing the classification efficiency the scalability.

TABLE 3: Parameter Comparison of Data Mining Algorithms with the Proposed Approach

Algorithm / Parameters	CART	ID3& C4.5	SLIQ & SPRINT	Proposed approach
Measure	Gini Diversity Index	Entropy info-gain	Gini Index	Info gain & Uncertainity coefficient
Procedure	Constructs Binary Decision Tree	Top-Down Decision Tree Construction	Decision Tree Construction in a Breadth first manner	Decision Tree with concepts of node merging and Height –balance using AVL trees
Pruning	Post pruning based on cost-complexity measure	Pre-pruning using a single pass algorithm	Post pruning based on MDL principle	Dynamic pruning based on thresholds

V. Conclusions And Future Work

This paper proposes a simple approach for classification using Decision Tree Induction and it clearly shows how the algorithm generalizes the concept hierarchies from the raw training data by attribute-oriented induction (AOI). By generalization of the raw training data, it relaxes the requirement of the training data and makes the decision tree result meaningful using the AVL trees concept. Specifically, this paper gives a better way to solve the Decision Tree Merging algorithm. Moreover, the proposed algorithm provides a general framework that can be used with any existing Decision Tree Construction algorithms. In an effort to identify and rectify the restriction that limits the efficiency and scalability of other algorithms, we have proposed an efficient yet simple solution which will overcome them. Our future work involves further refinement of the proposed algorithm. For example, we plan to implement the adaptive learning of the system in order to assign the priorities. The other thing can be improved for this algorithm is that the different level of the scalability can be projected using hyper links from one level to the next level. This gives the user more level of hierarchy and can browse the information layer by layer and stop where he wants to stop.

REFERENCES

[1] V. K. Vaishnavi, "Multidimensional height-balanced trees," IEEE Trans. Comput., vol. C-33, pp. 334-343, 1984.
[2] Tomoki Watanuma, Tomonobu Ozaki, and Takenao Ohkawa. "Decision Tree Construction from Multidimensional Structured Data". Sixth IEEE International Conference on Data Mining – Workshops, 2006.
[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification of Regression Trees. Wadsworth, 1984.
[4] Micheline Kamber, Lara Winstone, Wan Gong, Shang Cheng, Jiawei Han, "Generalization and Decision Tree Induction: Efficient Classification in Data Mining", Canada V5A IS6, 1996.
[5] J. R. Quinlan. Induction of decision trees. Machine Learning, 1:81–106, 1986.
[6] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
[7] L. B. Holder. Intermediate decision trees. In Proc. 14th Intl. Joint Conf. on Artificial Intelligence, pages 1056–1062, Montreal, Canada, Aug 1995.
[8] XindongWu ·Vipin Kumar ·J. Ross Quinlan ·Joydeep Ghosh ·Qiang Yang ·Hiroshi Motoda ·Geoffrey J. McLachlan ·Angus Ng ·Bing Liu ·Philip S. Yu ·Zhi-Hua Zhou ·Michael Steinbach ·David J. Hand · Dan Steinberg. A survey paper on "Top 10 algorithms in data mining" 2007.
[9] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In Proc. 1996 Intl. Conf. on Extending Database Technology (EDBT'96), Avignon, France, March 1996.
[10] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In Proc. 22nd Intl. Conf. Very Large Data Bases (VLDB), pages 544–555, Mumbai (Bombay), India, 1996.
[11] J. Han, Y. Cai, and N. Cercone. Datadriven discovery of quantitative rules in relational databases. IEEE Trans. Knowledge and Data Engineering, 5:29–40, 1993.
[12] D. H. Freeman, Jr. Applied Categorical Data Analysis. Marcel Dekker, Inc., New York, NY, 1987.

Devi Prasad Bhukya (1981) totally has 4.6 years industry experience in the Supercomputing, Performance engineering in SAN/Virtualization. He is presently pursing PhD program in the Department of Computer Science, University College of Engineering, Osmania University, Hyderabad, India. Prior to joining Osmania University, he worked as Senior Software Engineer in Wipro Technologies, India from November 2006 to February 2009 and he also had a technical and Member of Technical Staff (MTS) positions at the Centre for Development of Advanced Computing (C-DAC), Supercomputer Division at Bangalore from August 2004 to October 2006, India. Devi Prasad applies his business and technical expertise in the Grid Computing, Server Virtualization and Storage Area Networks. He served/serving as a reviewer for international conferences. He guided several students in the supercomputing projects at C-DAC, Bangalore in India. He authored several research papers devoted to HPC and performance engineering of Server Virtualization and SAN. Devi Prasad holds a B.Tech (2002) in Computer Science and Information Technology from JNTU University Hyderabad, India, M.Tech (2004) in Computer Science and Engineering from Indian Institute of Technology (IIT), Madras, India. He is a Member of ACM, IEEE, IACSIT and IIT ALUMNI.

Dr. S. Ramachandram (1959) received his bachelor's degree in Electronics and Communication (1983), Masters in Computer Science (1985) and a Ph.D. in Computer Science (2005). He is presently working as a Professor and Head, Department of Computer Science, University College of Engineering, Osmania University, Hyderabad, India. His research areas include Mobile Computing, Grid Computing, Server Virtualization and Software Engineering. He has authored several books on Software Engineering, handled several national & international projects and published several research papers at international and national level. He also held several positions in the university as a Chairman Board of Studies, Nodal officer for World Bank Projects and chair of Tutorials Committee. He is a member of Institute of Electrical and Electronic Engineers (IEEE), Computer Society of India (CSI) and Institute of Electronics and Telecommunication Engineers (IETE).