Complete Algorithms on Minimum Vertex - Cover

K.V.R.Kumar, Narendhar Maaroju, Deepak Garg

Abstract

The vertex cover problem is a classical NP-complete problem for which the best worst-case approximation ratio is 2 - o(1). This paper analyzes the hierarchical Bayesian optimization algorithm (hBOA) on minimum vertex cover for standard classes of random graphs. The performance of hBOA is compared with that of the branch-and-bound problem solver (BB), the simple genetic algorithm (GA) and the parallel simulated annealing (PSA). The results indicate that BB is significantly outperformed by all the other methods, which is expected as BB is a complete search algorithm and minimum vertex cover is an NP-complete problem. The best performance is achieved with hBOA; nonetheless, the performance differences between hBOA and other evolutionary algorithms are relatively small, indicating that mutation-based search and recombination-based search lead to similar performance on problem instances.

Keywords

Minimum vertex cover, hierarchical BOA, genetic algorithm, simulated annealing, branch and bound.

1. Introduction

The classical minimum vertex-cover problem involves graph theory and finite combinatorics and is categorized under the class of NP-complete problems in terms of its computational complexity [1, 8]. Minimum vertex cover has attracted researchers and practitioners because of the NP-completeness and because many difficult real-world problems can be formulated as instances of the minimum vertex cover. Examples of the areas where the minimum vertex cover problem occurs in real world applications are bioinformatics and communications [2]. However, only few studies exist that analyzes the performance of evolutionary algorithms on this important class of problems.

[1]The purpose of this paper is to compare several simple and advanced evolutionary algorithms on an extensive set of minimum vertex cover problem instances. Specifically, consider the hierarchical Bayesian optimization algorithm (hBOA) [3], the simple genetic algorithm (GA) [4], the parallel simulated annealing (PSA) [5], and the complete branch-and-bound solver (BB) [6]. As problem instances, standard random graph models are used [7, 12]. Each algorithm has been discussed on different graphs. The paper is organized as follows. Section 2 briefly describes the minimum vertex cover problem and its theoretical background. Section 3 outlines the hierarchical Bayesian optimization algorithm, the simple genetic algorithm, the parallel simulated annealing and the complete branch-and-bound solver. Section 4 discusses the graph models used for algorithms. Section 5 provides the analysis and their complexity. Section 6 gives future work ideas. Section 7 summarizes and concludes the paper.

2. Minimum Vertex Cover

A vertex cover of an undirected graph $G = (V, E)$, is a subset $S \subseteq V$ such that if $(u, v) \in E$ then either $u \in S$ or $v \in S$ or both. In other words, a vertex cover is a subset of vertices that contains at least one node of each edge. The size of the vertex cover is the number of vertices in it. There are two versions of the minimum vertex cover problem: the decision version and the optimization one. In the decision version, the task is to verify for a given graph whether there exists a vertex cover of a specified size. On the other hand, in the optimization version of this problem, the task is to find a vertex cover of minimum size. To illustrate the minimum vertex cover, consider the problem of placing guards [9] in a museum where corridors in the museum correspond to edges and the task is to place a minimum number of guards so that there is at least one guard at the end of each corridor.

Minimum vertex cover is NP-complete [8]. It is also a special case of the set cover problem which takes as input an arbitrary collection of subsets $S = (S_1, S_2, .., S_n)$ of the universal set V , and the task is to find a smallest subset of subsets from S that cover V.

The minimum vertex cover problem is also closely related to many other hard graph problems and so it interests the researchers in the design of optimization and approximation algorithms. For instance, the independent set problem is similar to the minimum vertex cover problem because a minimum vertex cover defines a maximum independent set and vice versa. Another interesting problem that is closely related to the minimum vertex cover is the edge cover which seeks the smallest set of edges such that each vertex is included in one of the edges.

Recently, the attention of physicists was drawn to the study of NP-complete problems like vertex cover and satisfiability. The reason is that, when studied on

[1] Manuscript received April15, 2009.

K.V.R.Kumar is with the Thapar University, Patiala, India (e-mail:skumar98765@gmail.com).

Narendhar Maaroju is with the Thapar University, Patiala, India (e-mail:maroju.83@gmail.com).

Dr.Deepak Garg is with the Thapar University, Patiala, India (e-mail:deepakgarg@ieee.org).

suitable random ensembles, these problems exhibit phase transitions in the solvability
[9] Which often coincide with peaks in the typical computational complexity or changes of the typical complexity from exponential to polynomial or vice-versa. Concepts and methods from statistical physics have helped to understand these models better, calculate typical complexities of algorithms analytically [1, 10] and have even lead to the design of more efficient probabilistic algorithms [16].
In this paper, consider the optimization version of minimum vertex cover with the goal of analyzing performance of various evolutionary algorithms and the complete branch and bound algorithm on this class of problems.

3. Comparison of Algorithms

There are two types of algorithms: incomplete and complete ones. Complete algorithms guarantee to find the optimum or true solution; hence the solution space is searched in principle completely. For incomplete algorithms, it is not ensured that the true solution or the global optimum is found. But they are very often sufficient for practical applications. This section outlines the algorithms compared in this paper: (1) the branch-and-bound algorithm (BB), (2) the hierarchical Bayesian optimization algorithm (hBOA), (3) the genetic algorithm (GA), and (4) the parallel simulated annealing (PSA).

3.1 Branch-and-bound algorithm

The branch-and-bound (BB) algorithm is a complete algorithm, meaning that it guarantees the exact solution even though the time complexity may increase exponentially with the graph size. As is also supported by the results presented in this paper, the algorithm is often outperformed by stochastic methods, which can often reliably locate the optimum after evaluating only a small portion of the search space.
The branch-and-bound algorithm recursively explores the full configuration space by deciding about the presence or absence of one node in the cover in each step of the recursion and recursively solving the problem for the remaining nodes. The full configuration space can be seen as a tree where each level decides about the presence or absence of one node and for each node there are two possible branches to follow; one corresponds to selecting the node for the cover whereas the other corresponds to ignoring the node. Technically, a covered node and all adjacent edges are removed, while an ignored node remains, but may not be selected in deeper levels of the recursion. The recursion explores the tree and backtracks when there are no more edges to cover or when the bounding condition is met, as described shortly. When backtracking, covered nodes are reinserted into the graph. Subsets of nodes that provide valid vertex covers are identified and the smallest of them is the minimum vertex cover. It is easy to see that in the worst case, the complexity of BB is upper-bounded by the total number of nodes in the recursion tree, which is proportional to 2^n.i.e o (2^n).
The bound applied in the following algorithm uses the current vertex degree d(i), which is the number of uncovered neighbours at a specific stage of the calculation. By covering a vertex *i* the total number of uncovered edges is reduced by exactly d(i). If several vertices $j_1, j_2. . . j_k$ are covered, the number of uncovered edges is at most reduced by $d(j_1) + d(j_2) + ... + d(j_k)$. Assume that at a certain stage within the backtracking tree, there are uncover edges uncovered and still k vertices to cover. Then a lower bound M for the minimum number of uncovered edges in the subtree is given by $M=Max[0,uncov\text{-}max\ d(j_1)+d(j_2)+. . .+d(j_k)]$.
The algorithm can avoid branching into a subtree if M is strictly larger than the number opt of uncovered edges in the best solution found so far.

3.2 Hierarchical BOA (hBOA)

The hierarchical Bayesian optimization algorithm (hBOA) is an estimation of distribution algorithm, where standard recombination and mutation operators are replaced by building and sampling probabilistic models. hBOA represents candidate solutions by *n*-bit binary strings, where *n* is the number of vertices in the graph; 1 represents the presence of a particular node in the minimum vertex cover while 0 represents its absence. HBOA starts by generating a population of candidate solutions at random with uniform distribution over all possible *n*-bit binary strings. At each iteration a set of promising solution is selected using any common selection method such as tournament and truncation selection. Here binary tournament selection is used without replacement. The selected solutions are used in building a Bayesian network with decision trees. New solutions are generated by sampling the built Bayesian network. The new solutions are then incorporated into the original population using restricted tournament replacement (RTR). The run is terminated when the termination criteria are met. There is a local search heuristic overlaid on top of hBOA, which updates every solution in the population to ensure that it represents a valid vertex cover. The update is done by adding nodes of uncovered edges to the current cover in random ordering until a valid cover is obtained. After adding new nodes to the cover, some of the selected nodes may be redundant; the redundant nodes are removed by parsing the nodes and deleting those that are unnecessary. Since every candidate solution represents a valid cover, in the selection process of better candidate solutions, the number of nodes selected for the cover in a solution directly corresponds to solution quality; the fewer nodes, the better the solution. Other heuristics are tried to repair invalid solutions, but their effects on performance were insignificant.

3.3 Genetic algorithm

GA is an optimization technique based on the natural evolution. It maintains a population of strings, called chromosomes that encode candidate solutions to a problem. The algorithm selects some parent chromosomes from the population set according to their fitness value, which are calculated using fitness function. The fittest chromosomes have more chances of selection for genetic operations in next generation. Different types of genetic operators are applied to the selected parent chromosomes; possibly according to the probability of operator, and next generation population set is produced. In every generation, a new set of artificial creatures is created using bits and pieces of the fittest chromosomes of the old population.

In the genetic algorithm (GA), use the same representation of candidate solutions and the same repair operator (local heuristic) like in hBOA.

GA starts by generating a random population of candidate solutions. At each iteration a population of promising solutions is first selected. Variation operators are then applied to this selected population to produce new candidate solutions. Specifically, crossover is applied to exchange partial solutions between pairs of solutions and mutation is used to perturb the resulting solutions. Here uniform crossover and bit-flip mutation is used to produce new solutions. The new solutions are substituted into the original population using restricted tournament replacement (RTR). The run is terminated when the termination criteria are met.

GA and hBOA differ only in the way they process the selected solutions. GA applies variation operators inspired by natural evolution and genetics, whereas hBOA learns and samples a Bayesian network with local structures.

The fitness function plays an important role in GA because it is used to decide how good a chromosome is. Fitness function is the number of vertices used to cover all the edges of the graph.

$$M=\sum_{i=1}^{v} Vi \text{ where } V_i=1 \text{ if } V_i \in V_{cover} \text{ else } 0$$

In HGA, one offspring is produced from two parent chromosomes. So, in that way best 50% chromosomes will directly go in the next generation using reproduction. All the chromosomes are used to create offspring using HVX. Because we believe that each chromosome has some important genes, which may become useful to obtain global optimal solution. Then mutation operator is applied to offspring. Mutation is used to avoid local minima and it should be applied on all the offspring.

3.4 Parallel simulated annealing

Simulated annealing is a fairly robust stochastic optimization algorithm based on local search. Simulated annealing derives inspiration from the physical process of annealing in metallurgy. Essentially annealing is a process in which equilibrium conditions in metals are attained by heating up and then cooling down the material in a controlled fashion. If the cooling is slow enough and reaches very low temperatures, the material will be with high probability in a ground state, i.e. a configuration with the lowest energy. Simulated annealing follows a similar analogy to solve optimization problems, by identifying the negative of the quality of the solutions (here the size of the vertex cover) with the energy.

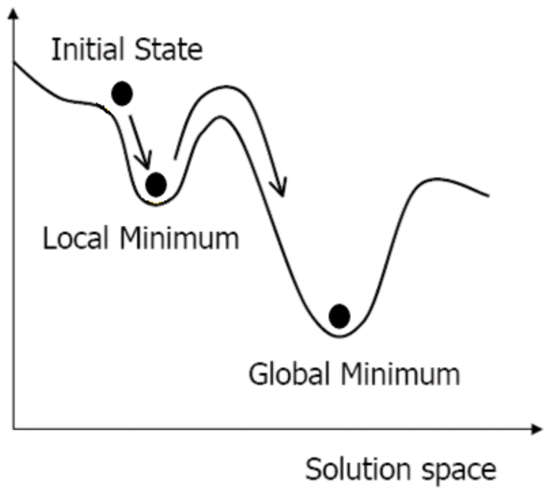

Fig.1 Shows the simulated annealing

In detail, simulated annealing starts initially with an arbitrary solution and then repeatedly tries to make improvements to it locally. A new solution is accepted with a probability that is based on the difference $Q_{old} - Q_{new}$ between the quality of the old and new solutions and on an (artificial) temperature T, which is gradually decreasing throughout the process. The algorithm always accepts better solutions but the probability of accepting a worse solution from the local step decreases exponentially fast with the ratio of the decrease in solution quality and the temperature T:

$$P_{accept} = \min \{1, \exp ((Q_{old} - Q_{new}) / T)\}$$

Initially, *T* is relatively large and thus SA accepts nearly all new solutions regardless of their quality. As *T* decreases, the probability of accepting worse solutions decreases as well. This means, when cooling slowly enough, the system will end up with high probability in a solution of best quality, i.e. in this case with a vertex cover of minimum size. Parallel simulated annealing (PSA) simulates multiple runs of simulated annealing in parallel and thus becomes more robust as the probability of reaching the global optimum increases.

In this paper, PSA is used that represents candidate solutions by n-bit binary strings and it initializes all solutions to represent a full cover (all nodes are selected) as suggested in [11]. Quality of a solution is determined by the number of nodes it uses in the cover; the fewer the nodes, the better the solution. Only local steps that lead to valid covers are accepted, so there is no need for a repair operator.

In each step of each SA run of PSA, first decide whether to add or remove a node from the current cover (with equal probability). If a node is to be removed, randomly choose one of the nodes that can be removed without affecting validity of the cover and remove the node; if no node can be removed, the cover remains unchanged. If a node is to be added into the cover, randomly choose a node to add and add the node with probability $\exp(-\mu)$. The parameter μ is initially equal to 0 and in each step; μ is increased by a constant $\delta_\mu > 0$.

4. Graph Models

This section outlines the graph models used in comparing the performance and scalability of the optimization algorithms introduced in the previous section. The graph models used are

(1) G(n, m) graphs [12]

(2) G(n, p) graphs [12]

4.1 G(n, m) Model

The G(n, m) model consists of all graphs with n vertices and m edges. The number of vertices, n and the number of edges, m are related by $m = nc$, where $c > 0$ is a constant.

To generate a random G(n, m) graph, start with a graph with no edges. Then, cn edges are generated randomly using a uniform distribution over all possible graphs with cn edges. Each node is thus expected to connect to 2c other nodes on average.

4.2 G(n, p) Model

The G(n, p) model, also called binomial Erdos Renyi random graph model, consists of graphs of n vertices for which the probability of an edge between any pair of nodes is given by a constant $p > 0$. To ensure that graphs are almost always connected, p is chosen so that $p \gg \log(n)/n$.

To generate a G(n, p) graph, start with an empty graph. Then, iterate through all pairs of nodes and connect each of these pairs with probability p.

The expected number of edges of a G(n, p) graph is $\binom{n}{2}p$. For given constants p and c, the number of edges for G(n, p) graphs grows faster than the number of edges for G(n,m) graphs; while for G(n,m) the number of edges is bounded by $\Theta(n)$, for G(n, p) the expected number of edges is bounded by $\Theta(n^2)$.

The classical G(n, m) and G(n, p) models are homogeneous in the sense that the node degrees tend to be concentrated around their expected value. For G(n, p) the expected degree is np whereas for G(n, m) it is 2c. The properties of G(n, m) model are similar to those of G(n, p) with

$2m/n^2 = p$.

5. Analysis

This section presents and discusses the analysis of all presented algorithms. The section starts by describing the graph instances and finally, the complexity is presented.

5.1 Graph instances

The graph instances have been generated using the two graph models described in the previous section.

For the G(n,m) model, based on the relation $m = cn$, c is varied from 0.5 to 4 in steps of 0.25 and for each value of c, n is varied from 50 to 300 in steps of 50. For each combination of n and c, 1000 random graphs are generated.

For the G(n, p) model, graphs are generated for $p = 0.25$ and $p = 0.5$. For each value of p, n is varied from 50 to 200 in steps of 50. For each combination of values of n and p, 1000 random instances are generated.

5.2 Complexity

All generated graphs have been first solved using BB because BB is a complete algorithm that is ensured to find the minimum vertex cover. All graphs have then been discussed using the remaining algorithms included in the comparison,

that is, hBOA, GA, and PSA. Time complexity of BB and PSA is measured by the number of steps until the optimal cover is found; for hBOA and GA, time complexity is measured by the overall number of candidate solutions examined until the optimum is found.

For hBOA and GA, the bisection method[13] is used to determine the minimum population size [15, 16] required to find the global optimum in 10 out of 10 independent runs (the population size is determined independently for each graph instance). The number of evaluations is then averaged over the 10 independent runs with the minimum population size obtained by bisection. Both hBOA and GA use binary tournament selection and new solutions are incorporated into the original population using restricted tournament replacement with window size $w = \min\{n, N/20\}$ where n is the number of nodes in the input graph and N is population size.

For PSA, 10 independent runs have been performed for each graph and the results have been averaged over the 10 runs. Parameters of PSA have been set according to initial experiments to ensure reliable convergence to the optimum vertex cover in a wide range of problem instances. Specifically, the number of parallel SA runs is set to the overall number of nodes in the graph, *n*. The probability of accepting an addition of a node into the cover is equal to $\exp(-\mu)$ where initially $\mu = 0$ and in each iteration, μ is increased by $\delta\mu = 0.05$. A number of alternative strategies are tried for modifying the acceptance probability for steps that decrease solution quality, but no approach lead to significantly different results than the strategy described above.

5.3 Time Complexity

The BB on random G(n, m) graphs, where the ratio of the number of edges and the number of nodes is fixed to a constant c. These indicate that the time complexity of BB grows exponentially fast with problem size for all values of c and that it increases with c. As problem size increases, BB is outperformed by all other compared algorithms in terms of both the number of evaluated candidate solutions. This is not surprising since BB is the only complete algorithm used in the comparison. Note that an extension of BB, the leaf-removal algorithm, is still a complete algorithm but it achieves a typically polynomially growing running time [14] for the average number of edges per node of at most $e \approx 2.7183$ (corresponding to $c \approx 1.3591$), while it still behaves exponentially for larger values of c. The performance of hBOA on G(n, m) test instances. Similarly as in the case of BB, the time complexity of hBOA increases with both c and n. However, for smaller values of c, the number of evaluated solutions appear to grow only polynomially fast with problem size as opposed to the exponential growth with BB.

The performance of all algorithms on G(n, m) for c = 2 and c = 4. GA and hBOA perform nearly the same for small values of n. However, for large problems, the growth of the number of evaluations required by GA becomes faster than that required by hBOA. Although the number of steps required by PSA is significantly greater than the number of evaluations for hBOA and GA, it is important to note that a single evaluation in hBOA and GA requires at least n steps, while in PSA the number of computational steps in each iteration of the algorithm on G(n, m) is upper bounded by a constant. After incorporating this factor into the analysis, we can conclude that for all values of c, hBOA, GA and PSA perform well and they significantly outperform BB. This indicates that performance of stochastic optimization techniques on G(n, m) is relatively efficient regardless of whether the search is based primarily on recombination or mutation.

The effects of c on performance of hBOA on G(n, m). The results indicate that as the problem size increases, the influence of c on increasing time complexity of hBOA grows.

Performance of hBOA and GA appears to follow a similar pattern on G(n, p) for all values of p. That indicates that the fluctuations in time complexity as the problem size grows are due to the distribution of problem instances, which appear to vary in difficulty more than in the case of G(n, m). Due to the variation in problem complexity, a higher number of problem instances should lead to more stable predictions of time complexity.

The observations for G(n, p) model test instances can be summarized as follows:

• As p increases the time complexity of all the algorithms decreases.

• All algorithms perform relatively well in all cases.

• BB overtakes PSA more quickly (for smaller instances) when p is small, and the growth of BB time complexity slows down with growing p.

Figure 2 compares the performance of hBOA, BB, GA, and PSA for the graphs. BB is outperformed by other methods but hBOA, GA and PSA perform well.

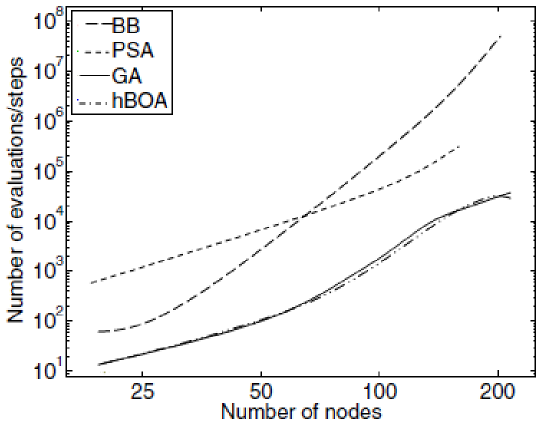

Fig.2 Comparison of hBOA, BB, GA and PSA

6. Future work

Future work would be to study performance of evolutionary algorithms for minimum vertex cover of various other classes of graphs. One of the most important challenges is to find features that make various instances of minimum vertex cover and other similar problems difficult and study the effectiveness of various stochastic optimizers in dealing with these features. The results presented in this report indicate that graph connectivity is one of these factors, but they also show that for many standard classes of graphs, most operators appear to perform relatively well. Finally, in both hBOA and GA, used a simple local repair operator, but it is straightforward to incorporate more advanced local searchers to further improve performance of GA and hBOA as was the case with other difficult classes of NP-complete problems, such as spin glasses.

7. Summary and Conclusion

This paper analyzed performance of the branch-and-bound (BB) algorithm and several evolutionary algorithms on minimum vertex cover for three classes of graphs. In addition to BB, considered the hierarchical Bayesian optimization algorithm (hBOA), the simple genetic algorithm (GA), and the parallel simulated annealing (PSA). In most cases, hBOA, GA and PSA outperformed BB, which is not a surprising result because BB is a complete method that guarantees that the global optimum is found. Nonetheless, the results indicated that in most cases, hBOA, GA and PSA performed comparably well, indicating that mutation-based search and recombination-based search perform similarly on the studied classes of graph instances.

8. References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill, New York, 2nd edition, 2001.

[2] X. Huang, J. Lai, and S. F. Jennings. Maximum common subgraph: Some upper bound and lower bound results. *BMC Bioinformatics*, 7(Suppl 4):S6, 2006.

[3] M.Pelikan. Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms. Spr inger-Verlag, 2005.

[4] R. Arakaki, and L. Lorena, "A Constructive Genetic Algorithm for the Maximal Covering Location Problem", in Proceedings of Metaheuristics International Conference, 2001, pp 13-17.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. Science, 220:671–680, 1983.

[6] R. Luling and B. Monien. Load balancing for distributed branch and bound algorithms. In The 6th International Parallel Processing Symposium, pages 543–549, Los Alamitos, USA, 1992. IEEE Computer Society Press.

[7] P. Erdos and A. Renyi. On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci., 5:17, 1960.

[8] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York, 1979.

[9] M. Weigt and A. K. Hartmann. The number guards needed by a museum – a phase transition in vertex covering of random graphs. Phys. Rev. Lett., 84:6118, 2000.

[10] M. Weigt and A. K. Hartmann. The typical-case complexity of a vertex-covering algorithm on finite-connectivity random graphs. *Phys. Rev. Lett.*, 86:1658, 2001

[11] K.Hartmann and M.Weigt.Phase Transitionsin Combinatorial Optimization Problems. Wiley-VCH, Weinheim, 2005.

[12] Bollobas. Random Graphs. Cambridge University Press, Cambridge, UK, 2nd edition, 2001.

[13] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, Department of General Engineering, Urbana, IL, 2001. Also IlliGAL Report No. 2002004.

[14] Miroslav Chleb and Janka Chleb, "Crown reductions for the Minimum Weighted Vertex Cover problem" Electronic Colloquium on Computational Complexity, Report No. 101 (2004).

[15] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. Annals of Discrete Mathematics, 25:27–45, 1985.

[16] M. Mezard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812, 2002.

K.V.R.kumar Author

This author became a member of IEEE in 2008. Pursuing Master of Engineering (2007-09) in Computer science from Thapar University, Patiala, Punjab, India. Member of organizing committee of IACC'09 .

Narendhar Maaroju Author

This author became a member of IEEE in 2008. Pursuing Master of Engineering (2007-09) in Computer science from Thapar University,Patiala,Punjab India. Member of organizing committee of IACC'09 .

Dr.Deepak Garg Author

has done his Ph.D. in the area of efficient algorithm design. He is certified on various technologies from Sun, IBM and Brain bench. He is Senior Member of IEEE and is Executive Member of IEEE Delhi Section. He is Life Member of ISTE, CSI, IETE, ISC, British Computer Society and ACM. He is also serving at different levels in various social and spiritual organizations. He has taught various core subjects at graduate and undergraduate levels and is guiding PhD students.

He has 18 publications in various International Journals and conferences. He is President of Creative Computer Society and Coordinator of various Industry collaborations in the University. He has served in various committees and has been a member of Senate of Thapar University. He has 10 years of teaching/research/development experience that includes working in IBM Corporation, USA. He has edited two books and is on the editorial board of an International Journal. He provides consultancy in the area of software development and technical education.