Missouri Estimation of Distribution Algorithms Laboratory

Hybrid Evolutionary Algorithms on Minimum Vertex Cover for Random Graphs

Rajiv Kalapala, Martin Pelikan and Alexander K. Hartmann

MEDAL Report No. 2007004

January 2007

Abstract

This paper analyzes the hierarchical Bayesian optimization algorithm (hBOA) on minimum vertex cover for standard classes of random graphs and transformed SAT instances. 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 tested 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 by 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 the tested classes of minimum vertex cover problems.

Keywords

Minimum vertex cover, hierarchical Bayesian optimization algorithm, branch and bound, genetic algorithm, parallel simulated annealing.

Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis
One University Blvd., St. Louis, MO 63121
E-mail: medal@cs.umsl.edu
WWW: http://medal.cs.umsl.edu/

Hybrid Evolutionary Algorithms on Minimum Vertex Cover for Random Graphs

Rajiv Kalapala
Missouri Estimation of Distribution Algorithms Lab (MEDAL)
Dept. of Math and Computer Science,
University of Missouri at St. Louis
One University Blvd., St. Louis, MO 63121
rkdnc@umsl.edu

Martin Pelikan
Missouri Estimation of Distribution Algorithms Lab (MEDAL)
Dept. of Math and Computer Science, 320 CCB
University of Missouri at St. Louis
One University Blvd., St. Louis, MO 63121
pelikan@cs.umsl.edu

Alexander K. Hartmann
Institut für Theoretische Physik
Universität Göttingen
Friedrich-Hund-Platz 1
37077 Göttingen, Germany
hartmann@physik.uni-goettingen.de

Abstract

This paper analyzes the hierarchical Bayesian optimization algorithm (hBOA) on minimum vertex cover for standard classes of random graphs and transformed SAT instances. 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 tested 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 by 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 the tested classes of minimum vertex cover problems.

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 (Garey & Johnson, 1979). Minimum vertex cover has attracted researchers and practitioners because of the NP-completeness and because many difficult real-life problems can be formulated as

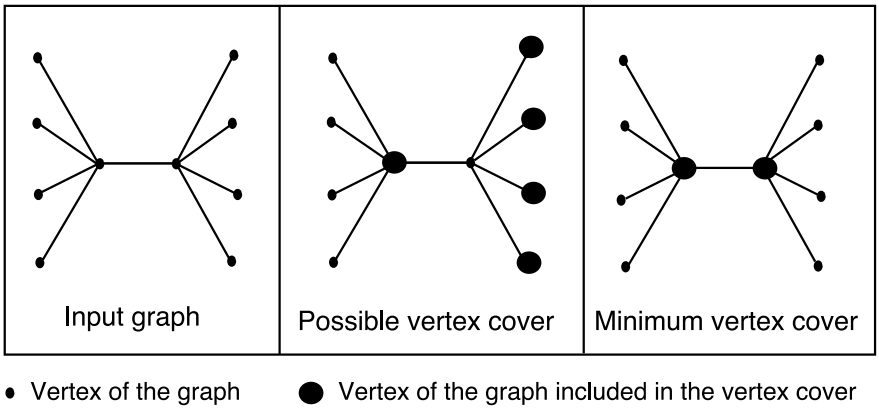


Figure 1: Minimum vertex cover

instances of the minimum vertex cover. Examples of the areas where the minimum vertex-cover problem occurs in real world applications are communications, civil and electrical engineering, and bioinformatics. However, only few studies exist that analyze the performance of evolutionary algorithms on this important class of problems (Khuri & Bäck, 1994; Hongwei, Xuezhou, Jin, & Zheng, 2000; Kotecha & Gambhava, 2003).

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, we consider the hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan, 2005), the simple genetic algorithm (GA) (Holland, 1975; Goldberg, 1989), the parallel simulated annealing (PSA) (Kirkpatrick, Gelatt, & Vecchi, 1983; Das & Chakrabarti, 2005), and the complete branch-and-bound solver (BB) (Lawler & Wood, 1966; Luling & Monien, 1992; Weigt & Hartmann, 2000a). As problem instances, we use standard random graph models (Erdös & Rényi, 1960) and transformed SAT graphs (Xu, Boussemart, Hemery, & Lecoutre, 2005). Each algorithm has been tested on more than 180,000 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 outlines the graph models used in the experiments. Section 5 provides the experiments done and their results. Section 6 gives a glimpse of future work ideas. Section 7 summarizes and concludes the paper.

2 Minimum Vertex Cover

The formal definition of a vertex cover of a graph **G** with vertex set **V** and edges **E** is stated as follows:

> *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. The size of the vertex cover is the number of vertices in it.*

$$S \subseteq V : \forall\ u,v \in E : u \in S \vee b \in S$$

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 minimum vertex cover, consider the problem of placing guards (Weigt & Hartmann, 2000b) 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. Figure 1 depicts the problem in brief.

Minimum vertex cover is one of the Karp's 21 diverse combinatorial and graph theoretical problems (Karp, 1972), which were proved to be NP-complete. Minimum vertex cover is a special case of the set cover problem (Thomas H. Cormen & Stein, 2001) 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 whose union is V.

The minimum vertex cover problem is also closely related to many other hard graph problems and so it interests the researchers in the field of design of optimization and approximation algorithms. For instance, the independent set problem (Karp, 1972; Garey & Johnson, 1979) 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 suitable random ensembles, these problems exhibit phase transitions in the solvability (Monasson, Zecchina, Kirkpatrick, Selman, & Troyansky, 1999; Weigt & Hartmann, 2000b; Hartmann & Rieger, 2004), which often coincide which 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 (Hartmann & Weigt, 2005), calculate typical complexities of algorithms analytically (Monasson & Cocco, 2001; Weigt & Hartmann, 2001) and have even lead to the design of more efficient probabilistic algorithms (Mézard, Parisi, & Zecchina, 2002; Hartmann & Weigt, 2005).

In this paper we consider the optimization version of minimum vertex cover with the goal of analyzing performance of various evolutionary algorithms on this class of problems. All algorithms are also compared to a complete method based on the branch-and-bound algorithm.

3 Compared Algorithms

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) parallel simulated annealing (PSA).

3.1 Branch-and-bound algorithm

The branch-and-bound (BB) algorithm (Lawler & Wood, 1966; Luling & Monien, 1992) is a complete algorithm, meaning that it guarantees the exact solution even though the time complexity may increase exponentially with the graph size. Added to the potential complexity, the algorithm is often outperformed by stochastic methods that only evaluate a small portion of the search space and are often capable of locating the global optima reliably.

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 lower bounded by the total number of nodes in the recursion tree, which is proportional to 2^n.

Bounding enables the algorithm to eliminate branches that provably do not lead to better than best-so-far covers, improving efficiency. The bounding condition is based on the degree $d(i)$ of the current vertex i, which represents the number of neighbors of node i. If *uncov* is the number of edges yet to be covered and we have still k nodes to cover, the lower bound M for the minimum number of uncovered edges is given by

$$M = \max\left[0, uncov - \max_{j_1,j_2,..,j_k} d(j_1) + d(j_2) + .. + d(j_k)\right]$$

Any branch that violates the above condition can be cut (avoided). Our BB implementation is based on Weigt and Hartmann (2000a).

3.2 Hierarchical Bayesian optimization algorithm

The hierarchical Bayesian optimization algorithm (Pelikan & Goldberg, 2001; Pelikan, 2005) is an estimation of distribution algorithm (Larrañaga & Lozano, 2002; Baluja, 1994; Mühlenbein & Paaß, 1996; Pelikan, Goldberg, & Lobo, 2002; Pelikan, Sastry, & Cantú-Paz, 2006), where standard recombination and mutation operators are replaced by building and sampling probabilistic models. hBOA represents candidate solutions by n-bit binary string, 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 we use binary tournament selection. This means iteratively pairs of solutions are taken randomly from the current population, and the better solution of the two is "selected". The selected solutions are used in building a Bayesian network with decision trees (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999). New solutions are generated by sampling the built Bayesian network. The new solutions are then incorporated into the original population using restricted tournament replacement technique (RTR) (Harik, 1995). 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.

3.3 Genetic algorithm

In the genetic algorithm (GA) (Holland, 1975; Goldberg, 1989), we use the same representation of candidate solution and the same repair operator 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 we use uniform crossover and bit-flip mutation 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.

3.4 Parallel simulated annealing

Simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983; Das & Chakrabarti, 2005) 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.

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 $Q_{\text{old}} - Q_{\text{new}}$ 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:

$$p_{\text{accept}} = \min\{1, \exp\left((Q_{\text{old}} - Q_{\text{new}})/T\right)\} \tag{1}$$

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, we use a PSA that represents candidate solutions by n-bit binary strings and it initializes all solutions to represent a full cover (all nodes are selected). 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.

Algorithm 4.1: GENERATE(G, n, c)

```
Initialize graph G(V, E)
m ← n * c
for i ← 1 to m
   repeat
    e ← random edge
   until e not present in E
   E ← E ∪ {e}
return (G)
```

Figure 2: The pseudocode for generating $G(n, m)$ graphs.

In each step of each SA run of PSA, we first decide whether to add or remove a node from the current cover (with equal probability). If a node is to be removed, we 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, we 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 (Bollobás, 2001; Weigt & Hartmann, 2000a),

(2) $G(n, p)$ graphs (Bollobás, 2001), and

(3) transformed SAT graphs (Xu, Boussemart, Hemery, & Lecoutre, 2005).

The first two models are standard random graph models from the graph theory, while the third model is created by transforming hard instances of SAT problem into minimum vertex cover. All graphs used in this work are undirected but the presented algorithms can be extended and applied to directed graphs in a straightforward manner.

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, we 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. The pseudo-code for the random graph generation is shown in figure 2.

Algorithm 4.2: GENERATE(G, n, p)

Initialize graph $G(V, E)$
for $i \leftarrow 1$ **to** n
  for $j \leftarrow i + 1$ **to** n
    add edge (i, j) *to* E *with probability* p
return (G)

Figure 3: The pseudocode for generating $G(n, p)$ graphs.

4.2 G(n,p) Model

The $G(n, p)$ model, also called binomial Erdős Rényi random graph model (Bollobás, 2001), 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 we start with an empty graph. Then, we iterate through all pairs of nodes and connect each of these pairs with probability p. The pseudocode for generating $G(n, p)$ graphs is shown in figure 3.

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 $2m/n^2 = p$.

4.3 Transformed SAT graphs (TSAT)

The satisfiability (SAT) problem consists of deciding whether a given Boolean formula in conjunctive normal form is satisfiable, that is, whether there exists an assignment of the variables that satisfies the formula (Garey & Johnson, 1979). The transformed SAT graphs used in this paper are generated by transforming forced satisfiable SAT benchmarks of model RB (Xu & Li, 2000; Xu & Li, 2003), where vertices correspond to variables and edges correspond to binary clauses in SAT instances.

Model RB is the revised model B, which is one of the standard models for generating random binary constraint satisfaction problems (CSPs) (Tsang, 1993). By varying control parameters of the model, we can ensure that the generated instances are from the phase transition region where the probability of any instance being satisfiable is about 50%; instances from the phase transition region are known to be the most difficult ones for most algorithms. The control parameters for model RB are n, which denotes the number of variables; $\alpha > 0$, which determines the domain size $d = n^\alpha$; $r > 0$ which determines the number $m = rn \ln n$ of constraints; and $1 < p < 0$, which denotes the tightness of the constraints.

The model RB used here is of interest because random instances of any *arity* can be generated and the phase transition is guaranteed with a limited restriction on domain size and constraint tightness. The critical probability p_{cr} where the transition occurs is given by $p_{cr} = 1 - e^{-\alpha/r}$. The algorithm for generating the transformed SAT graphs is shown in figure 4: The size of the minimum

1. Generate n disjoint cliques of size n^α.
2. Randomly select two different cliques and then generate without repetitions $pn^{2\alpha}$ random edges between these two cliques.
3. Run Step 2 (with repetitions) for another $rn\mathbf{ln}n - 1$ times.

Figure 4: Generating transformed SAT (TSAT) graphs.

vertex cover of TSAT graphs is approximately equal to $n(n^\alpha - 1)$.

5 Experiments

This section presents and discusses the experimental results. The section starts by describing the test instances and the experiments done. Finally, the results are presented.

5.1 Tested graph instances

The test graph instances have been generated using the three graph models described in the previous section. We have generated and tested more than 180,000 different graphs.

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 and tested.

For the $G(n, p)$ model, graphs are generated for $p = 0.25$, 0.5 and 0.75. 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 and tested.

For the TSAT model, the number cliques in the generated graphs is varied from 5 to 20. The number of nodes is then determined by the number of cliques and other parameters. The remaining parameters are set as in Xu, Boussemart, Hemery, and Lecoutre (2005): $\alpha = 0.8$, $r = 2.7808$, and $p = 0.25$. Here p is the critical value where the hardest instances occur. For each problem size, 1000 random instances are generated and tested.

5.2 Description

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 solved 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 has been 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 (Sastry, 2001) is used to determine the minimum population size required by the algorithm to converge in 10 out of 10 independent runs (the population size is determined independently for each graph instance). For each problem instance, the time complexity of hBOA and GA has been measured by the number of evaluations until successful convergence to the global optimum in the 10 successful runs with the population size obtained

with 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 $\delta_\mu = 0.05$. We tried a number of alternative strategies 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 Results

Figure 5 presents the results of 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 results 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 as well as the overall CPU time per instance. This result is not surprising since BB is the only *complete* algorithm used in the comparison. Note that an extension of BB, the *leaf-removal algorithm* (Bauer & Golinelli, 2001), is still a complete algorithm but it achieves a typically polynomially growing running time for $c \leq e \approx 2.7183$, while it still behaves exponentially for larger values of c.

Figure 6 presents 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 most values of c, the number of evaluated solutions appears to grow only polynomially fast with problem size as opposed to the exponential growth with BB. Note that hBOA is required to converge in all runs, 10 independent runs per problem instance; each point in the graph thus corresponds to 10,000 successful runs.

Figures 7 compares the performance of all tested 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 $G(n, m)$ bounded from above 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.

Figure 8 analyzes 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.

The observations from the test results for $G(n, m)$ model test instances can be summarized as follows:

- As c increases the complexity of all the algorithms increases.
- hBOA, GA and PSA outperform BB in all the cases.
- BB exhibits nearly exponential complexity in all the cases as expected.

- hBOA, GA and PSA perform similarly well and exhibit polynomial complexity for most values of c.
- Operators based on recombination and local search lead to similar performance.

Figure 9 shows the results of the experiments done for the $G(n, p)$ model with increasing p. The results indicate that hBOA and GA perform nearly equally well in all cases and since evaluating one solution in hBOA or GA is slower than one local step of BB or PSA, we can conclude that all algorithms perform well. A comparison of the results for $G(n, m)$ and $G(n, p)$ indicates that $G(n, p)$ graphs are easier to solve for all methods, assuming that c and p are fixed to a constant.

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 (see figure 9a-b) 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 from the test results for $G(n, p)$ model test instances can be summarized as follows:

- As p increases the 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 the time complexity of BB slows down with increasing p.

In general we see that graphs with few edges as well as graphs similar to a clique are relatively easy to solve even with complete algorithms like BB. Graphs with covers that are neither small nor large are expected to be much more difficult.

Figure 10 compares the performance of hBOA, BB, GA, and PSA for the transformed SAT graphs. On TSAT, BB is outperformed by other methods but hBOA, GA and PSA perform well.

6 Future work

Methodology presented in this paper can be extended to other generalized NP-complete problems, such as the set cover problem, independent set problem, number-partitioning problem or the satisfiability problem. The other direction to extend the current 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 paper 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. Understanding the main sources of problem difficulty would facilitate the design of better optimizers for minimum vertex cover and similar problems. Finally, in both hBOA and GA, we 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 (Pelikan & Hartmann, 2006).

7 Summary and conclusions

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, we 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 and.

Acknowledgments

This project was sponsored by the National Science Foundation under CAREER grant ECS-0547013, by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, by the Volkswagenstiftung within the program "Nachwuchsgruppen an Universitäten", and by the University of Missouri in St. Louis through the High Performance Computing Collaboratory sponsored by Information Technology Services, and the Research Award and Research Board programs.

The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. Most experiments were completed at the Beowulf cluster at the University of Missouri–St. Louis.

References

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning* (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA: Carnegie Mellon University.

Bauer, M., & Golinelli, O. (2001). Core percolation in random graphs: a critical phenomena analysis. *Eur. Phys. J. B*, *24*, 339.

Bollobás, B. (2001). *Random graphs* (2nd ed.). Cambridge, UK: Cambridge University Press.

Chickering, D. M., Heckerman, D., & Meek, C. (1997). *A Bayesian approach to learning Bayesian networks with local structure* (Technical Report MSR-TR-97-07). Redmond, WA: Microsoft Research.

Das, A., & Chakrabarti, B. K. (Eds.) (2005). *Quantum annealing and related optimization methods*, Volume 679. New York: Springer.

Erdös, P., & Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, *5*, 17.

Friedman, N., & Goldszmidt, M. (1999). Learning Bayesian networks with local structure. In Jordan, M. I. (Ed.), *Graphical models* (pp. 421–459). Cambridge, MA: MIT Press.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness*. New York: Freeman.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Harik, G. R. (1995). Finding multimodal solutions using restricted tournament selection. *Proceedings of the International Conference on Genetic Algorithms (ICGA-95)*, 24–31.

Hartmann, A. K., & Rieger, H. (Eds.) (2004). *New optimization algorithms in physics*. Weinheim: Wiley-VCH.

Hartmann, A. K., & Weigt, M. (2005). *Phase transitions in combinatorial optimization problems*. Weinheim: Wiley-VCH.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Hongwei, H., Xuezhou, X., Jin, X., & Zheng, B. (2000). Solving vertex covering problems using hybrid genetic algorithms. In *Proceedings of the 5th International Conference on Signal Processing* (pp. 1663–1666).

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Symposium on the Complexity of Computer Computations* (pp. 85–103). Yorktown Heights, NY: Plenum, NY.

Khuri, S., & Bäck, T. (1994). An evolutionary heuristic for the minimum vertex cover problem. *Genetic Algorithms within the Framework of Evolutionary Computation: Proceedings of the KI-94 Workshop*, 86–90.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Kotecha, K., & Gambhava, N. (2003). A hybrid genetic algorithm for minimum vertex cover problem. In Prasad, B. (Ed.), *The First Indian International Conference on Artificial Intelligence* (pp. 904–913). Hyderabad: IICAI.

Larrañaga, P., & Lozano, J. A. (Eds.) (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Boston, MA: Kluwer.

Lawler, E. L., & Wood, D. E. (1966). *Branch-and-bound methods: a survey* (Operational Research 14:699). Ann Arbor, MI: University of Michigan, Ann Arbor.

Luling, R., & Monien, B. (1992). Load balancing for distributed branch and bound algorithms. In *The 6th International Parallel Processing Symposium* (pp. 543–549). Los Alamitos, USA: IEEE Computer Society Press.

Mézard, M., Parisi, G., & Zecchina, R. (2002). Analytic and algorithmic solution of random satisfiability problems. *Science*, *297*, 812.

Monasson, R., & Cocco, S. (2001). rajectories in phase diagrams, growth processes, and computational complexity: How search algorithms solve the 3-satisfiability problem. *Phys. Rev. Lett.*, *86*, 1654.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., & Troyansky, L. (1999). Determining computational complexity from characteristic phase transitions. *Nature*, *400*, 133.

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In Eiben, A., Bäck, T., Shoenauer, M., & Schwefel, H. (Eds.), *Parallel Problem Solving from Nature* (pp. 178–187). Berlin: Springer Verlag.

Pelikan, M. (2005). *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Springer-Verlag.

Pelikan, M., & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 511–518. Also IlliGAL Report No. 2000020.

Pelikan, M., Goldberg, D. E., & Lobo, F. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, *21*(1), 5–20. Also IlliGAL Report No. 99018.

Pelikan, M., & Hartmann, A. (2006). Searching for ground states of Ising spin glasses with hierarchical BOA and cluster exact approximation. Springer.

Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.) (2006). *Scalable optimization via probabilistic modeling: From algorithms to applications*. Springer.

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

Thomas H. Cormen, Charles E. Leiserson, R. L. R., & Stein, C. (2001). *Introduction to algorithms* (2nd ed.). New York: McGraw-Hill.

Tsang, E. (1993). *Foundations of constraint satisfaction*. London: Academic Press.

Weigt, M., & Hartmann, A. K. (2000a). Minimal vertex covers on finite-connectivity random graphs — A hard-sphere lattice-gas picture. *Phys. Rev. E*, *63*, 056127.

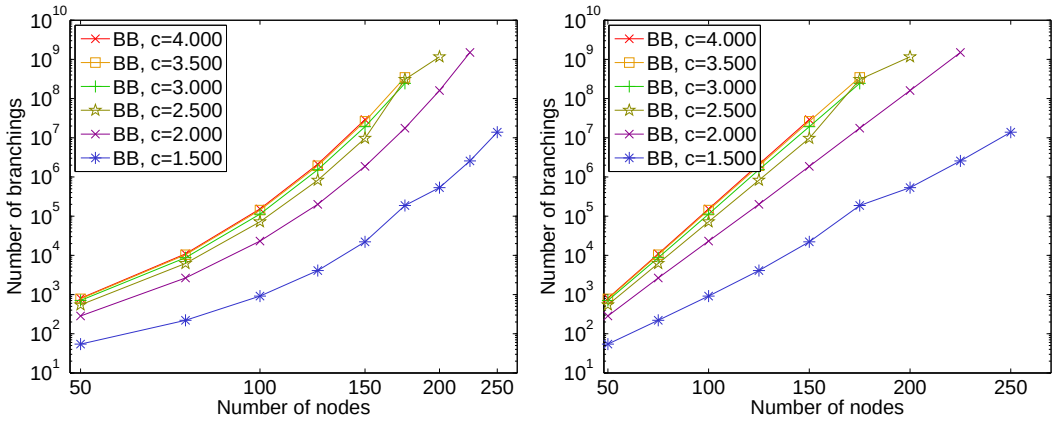
Weigt, M., & Hartmann, A. K. (2000b). The number guards needed by a museum – a phase transition in vertex covering of random graphs. *Phys. Rev. Lett.*, *84*, 6118.

Weigt, M., & Hartmann, A. K. (2001). The typical-case complexity of a vertex-covering algorithm on finite-connectivity random graphs. *Phys. Rev. Lett.*, *86*, 1658.

Xu, K., Boussemart, F., Hemery, F., & Lecoutre, C. (2005). A simple model to generate hard satisfiable instances. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 337–342). Edinburgh, Scotland.

Xu, K., & Li, W. (2000). Exact phase transitions in random constraint satisfaction problems. *Artificial Intelligence Research*, *12*, 93103.

Xu, K., & Li, W. (2003). *Many hard examples in exact phase transitions with application to generating hard satisfiable instances* (Technical Report cs.CC/0302001).



(a) Log-log scale. (b) Semilog scale.

Figure 5: BB on $G(n, m)$

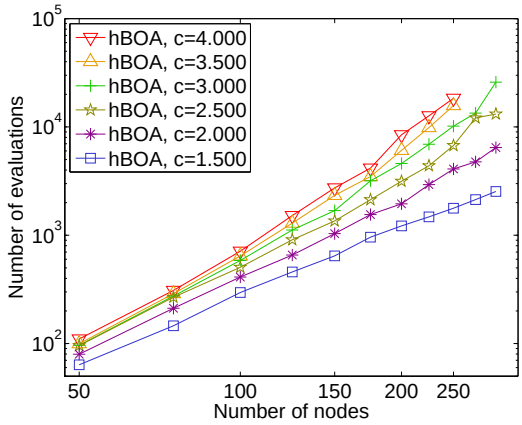
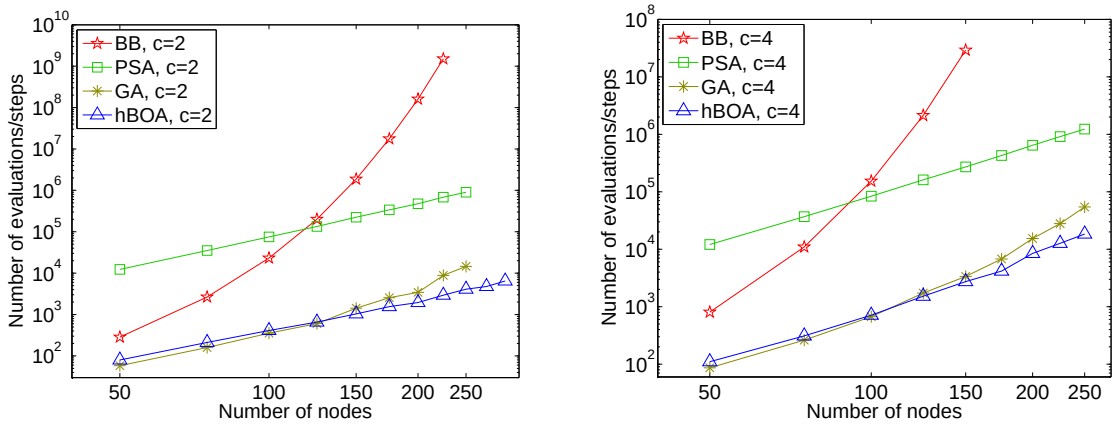


Figure 6: hBOA on $G(n, m)$



(a) $c = 2$ (b) $c = 4$

Figure 7: hBOA, BB, GA, PSA on $G(n, m)$.

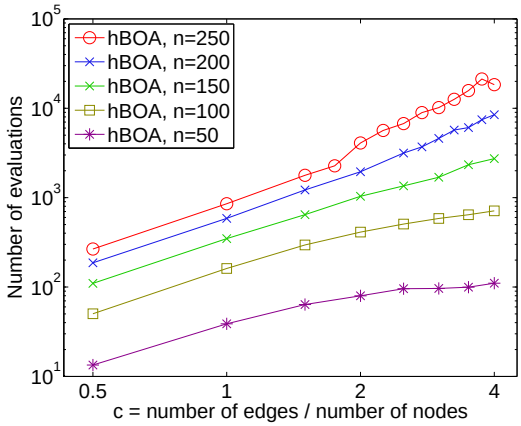
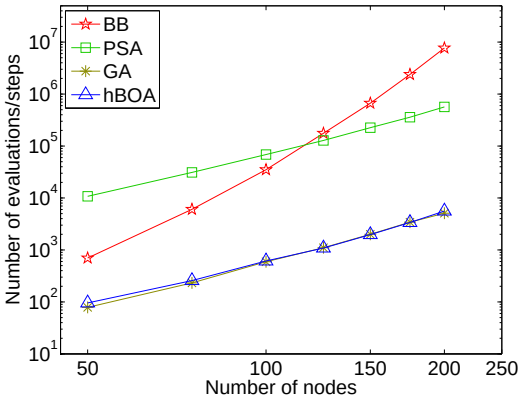
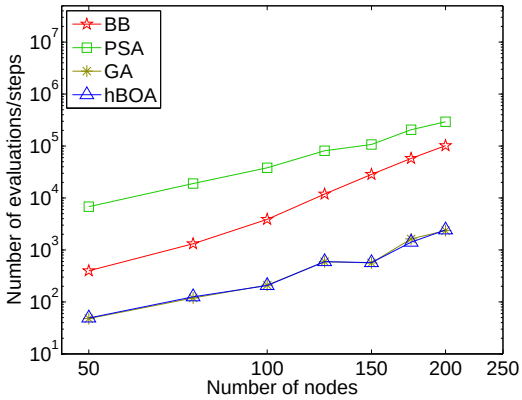


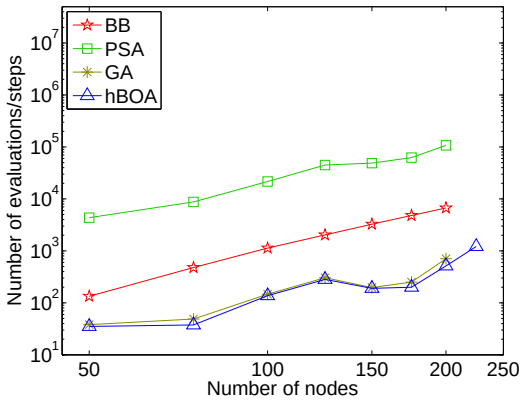
Figure 8: Effects of c on hBOA performance on $G(n, m)$



(a) $p = 0.25$



(b) $p = 0.50$



(c) $p = 0.75$

Figure 9: hBOA, BB, GA, PSA on $G(n, p)$. Note that the performance of hBOA and GA is nearly identical.

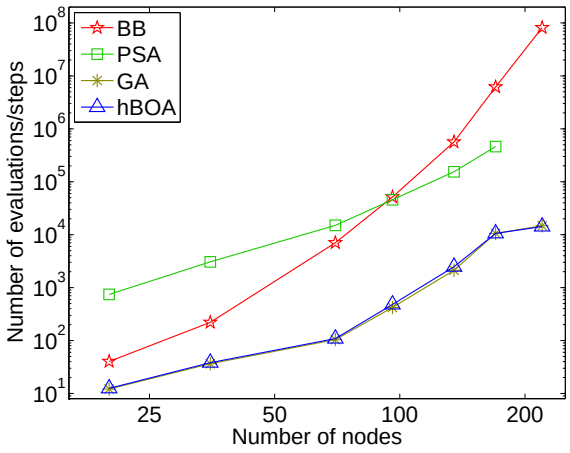


Figure 10: Comparison of hBOA, BB, GA and PSA on TSAT instances. Note that the performance of hBOA and GA is nearly identical.