Using Previous Models to Bias Structural Learning in the Hierarchical BOA

Mark Hauschild
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Mathematics and Computer Science
University of Missouri in St. Louis
mwh308@umsl.edu

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Mathematics and Computer Science
University of Missouri in St. Louis
pelikan@cs.umsl.edu

Kumara Sastry
Illinois Genetic Algorithms Laboratory (ILLiGAL)
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
ksastry@uiuc.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory (ILLiGAL)
University of Illinois at Urbana-Champaign
deg@uiuc.edu

ABSTRACT

Estimation of distribution algorithms (EDAs) are stochastic optimization techniques that explore the space of potential solutions by building and sampling probabilistic models of promising candidate solutions. While the primary goal of applying EDAs is to discover the global optimum (or an accurate approximation), any EDA also provides us with a sequence of probabilistic models, which hold a great deal of information about the problem. Although using problem-specific knowledge has been shown to significantly improve performance of EDAs and other evolutionary algorithms, this readily available source of information has been largely ignored by the EDA community. This paper takes the first step towards the use of probabilistic models obtained by EDAs to speed up the solution of similar problems in the future. More specifically, we propose two approaches to biasing model building in the hierarchical Bayesian optimization algorithm (hBOA) based on knowledge automatically learned from previous runs on similar problems. We show that the methods lead to substantial speedups and argue that they should work well in other applications that require solving a large number of problems with similar structure.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.6 [**Artificial Intelligence**]: Learning; G.1.6 [**Numerical Analysis**]: Optimization

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

General Terms

Algorithms, Performance

1. INTRODUCTION

Estimation of distribution algorithms (EDAs) [14, 20, 21] are among the most powerful and generally applicable genetic and evolutionary algorithms. EDAs replace traditional variation operators of genetic algorithms—such as crossover and mutation—by building a probabilistic model of promising solutions and sampling the built model to generate new candidate solutions. While EDAs have many advantages over standard genetic algorithms [14, 21], one in particular this paper will focus on is that at the end of an EDA run, a series of probabilistic models of our solution space have been built, which hold a great deal of information about the problem. Although such information should be useful for effective efficiency enhancement and it has often been argued that using problem-specific knowledge should significantly improve EDA performance [27, 1], the use of this readily available source of problem-specific information has been practically ignored by the EDA community.

This paper takes the first step toward the design of automated techniques for learning from experience and exploiting information included in probabilistic models learned in the past to speed up future EDA runs. This will be done by biasing model building using information gathered from probabilistic models obtained on similar problems in previous runs. While the techniques discussed in this paper are designed for the hierarchical Bayesian optimization algorithm (hBOA), the proposed techniques can be adapted to other EDAs based on multivariate probabilistic models in a straightforward manner.

The paper is organized as follows. Section 2 outlines hBOA. Section 3 discusses learning from experience in EDAs and hBOA. Section 4 proposes two methods for automatically biasing model building in hBOA using the probabilistic models learned in previous hBOA runs on similar problems. Section 5 presents experimental results. Finally, section 6 summarizes and concludes the paper.

2. HIERARCHICAL BOA (HBOA)

The hierarchical Bayesian optimization algorithm (hBOA) [18, 17] evolves a population of candidate solutions represented by fixed-length strings over a finite alphabet (for example, binary strings). The initial population is generated at random according to the uniform distribution over the set of all potential solutions. Each iteration (generation) starts by selecting promising solutions from the current population using any standard selection method of genetic and evolutionary algorithms. In this paper we use truncation selection with threshold $\tau = 50\%$.

After selecting the promising solutions, hBOA builds a Bayesian network [13, 16] with local structures [4, 6] as a model for these solutions. New solutions are generated by sampling the built network. These are then incorporated into the original population using restricted tournament replacement (RTR) [10], which ensures effective diversity maintenance. The window size w in RTR was set as $w = \min\{n, N/20\}$, where n is the number of decision variables and N is the population size, as suggested in ref. [17]. The next iteration is then executed unless some predefined termination criteria are met.

A Bayesian network (BN) [16, 13] consists of two components: (1) *Structure*, which is defined by an acyclic directed graph with one node per variable and the edges corresponding to conditional dependencies between the variables, and (2) *parameters*, which consist of the conditional probabilities of each variable given the variables that this variable depends on. A BN with n nodes encodes a joint probability distribution of n random variables $X_1, X_2, \ldots, X_n$:

$$p(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\Pi_i), \quad (1)$$

where Π_i is the set of variables from which there exists an edge into X_i (members of Π_i are called parents of X_i).

hBOA uses the set of selected solutions to automatically learn both the structure (edges) as well as the parameters (conditional probabilities) of the model. Learning the structure is the most challenging task of model building [17]. In this paper, we use the greedy algorithm for learning the structure of BNs with local structures [17]. To evaluate structures, the Bayesian-Dirichlet metric with likelihood equivalence for BNs with local structures [4] is used with an additional penalty for model complexity [6, 17]. For more details on learning and sampling BNs with local structures in hBOA, see refs. [4, 18, 17].

Incorporating local search often improves efficiency of evolutionary algorithms. Even a simple deterministic hill climber (DHC) was shown to lead to substantial speedups of hBOA [17]. That is why we decided to incorporate DHC into hBOA also in this study. DHC takes a candidate solution represented by an n-bit binary string on input. Then, it performs one-bit changes on the solution that lead to the maximum improvement of solution quality. DHC is terminated when no single-bit flip improves solution quality. Here, DHC is used to improve every solution in the population before the evaluation is performed.

3. LEARNING FROM EXPERIENCE

GEAs typically do not require any information about the problem being solved except for the representation of candidate solutions and the fitness function. Nonetheless, if problem-specific information is available, it may be possible to use this information to improve performance of these algorithms significantly. There are two basic approaches to speed up EDAs by incorporating problem-specific knowledge: (1) bias the procedure for generating the initial population [27, 24, 19] and (2) bias or restrict the model building procedure [27, 15, 1]. For both these approaches, we may either (1) hard code the modifications based on prior problem-specific knowledge or (2) develop automated procedures to improve EDA performance by learning from previous EDA runs on problems of similar type (learning from experience).

One technique used to bias the initial population towards good solutions (and, consequently, to also improve model quality) is called *seeding*. Seeding works by inserting high-quality solutions into the initial population. These high-quality solutions can be either obtained from previous runs on similar problems, provided by a specialized heuristic [27, 19], or created in some way from high-quality solutions of smaller instances of the same problem [24]. Seeding can often lead to dramatic improvements. For example, in atomic cluster optimization—where the goal is to find an atomic arrangement that minimizes energy—Sastry [24] dramatically lowered the asymptotic complexity of ECGA by seeding the population with atomic configurations with one fewer atom.

All prior work on biasing the model building in EDAs was based on using prior problem-specific knowledge and an assumed relation between the structure of the problem and adequate probabilistic models. The first attempt to bias model building in EDAs based on prior problem-specific knowledge was made by Schwarz & Ocenasek [27]. In this study, BOA application to graph bipartitioning was considered where the task is to split the nodes of a given graph into two equally sized partitions so that the number of connections between the two partitions is minimized. Since it can be hypothesized that the nodes that are connected in the underlying graph may be more likely connected in the probabilistic model as well, they proposed the use of a prior network to bias model building to structures that contain edges between the pairs of nodes connected in the underlying graph. While other dependencies were also allowed, the dependencies between nodes connected in the underlying graph were given higher priority. The bias towards any given prior network or a set of prior networks can be easily incorporated into Bayesian metrics used to build the Bayesian network and thus whenever the user has information about good candidate networks, a similar approach can be used.

Mühlenbein & Mahnig [15] also considered graph bipartitioning. However, instead of incorporating a prior network that provides a soft bias towards models that closely correspond to the underlying graph, here the models were restricted to only allow connections between nodes connected in the underlying graph. This can both restrict the model complexity as well as speed up the model building.

Finally, Baluja [1] discussed how to bias EDA model building on the problem of graph coloring, in which the task is to assign a color to each node out of the set of available colors so that no connected nodes have the same color. Similarly as in the study of Mühlenbein & Mahnig [15], even here the probabilistic models were restricted to only contain edges that were consistent with the underlying graph structure; all dependencies between unconnected nodes were simply disallowed. However, in this study, dependency-tree models were used while in the study of Mühlenbein & Mahnig [15], Bayesian networks were used.

Most experimental results on using prior knowledge to bias either the model building or the generation of the initial population suggest that prior knowledge leads to significant speedups of EDAs and allows EDAs to solve larger problems than they could solve otherwise. Nonetheless, there are three difficulties with applying the aforementioned techniques in hBOA and other EDAs. First of all, different problem instances of most problem types are expected to yield significantly different solutions; therefore, the utility of seeding and other approaches to biasing the generation of the initial population can be expected to be severely limited. Second, it may be difficult to process prior information about the problem structure to define adequate restrictions of model structure. Finally, even if prior information about the problem structure is relatively straightforward to obtain and process, it may not be clear how to use this information to provide adequate model restrictions.

4. BIASING THE MODEL BUILDING

Clearly, when an EDA is applied to an optimization problem, the most important result of the computation is the best solution found. However, EDAs provide us with much more than just the best solution—EDAs create a sequence of probabilistic models encoding populations of increasing quality, which hold a lot of information about the problem being solved. Despite that EDAs often provide us with this readily available source of problem-specific information and that problem-specific knowledge has been shown to often significantly improve EDA performance, the EDA community has practically ignored the potential of using probabilistic models discovered by EDAs in this way. This paper takes the first step toward the design of automated techniques for exploiting probabilistic models learned by hBOA for speeding up future hBOA runs on problems of similar type.

This section describes two approaches to biasing the model building based on probabilistic models obtained in previous runs on similar problems:

(1) Bias model building using the probability coincidence matrix, and

(2) bias model building using the distance threshold.

Both the proposed techniques start with a number of sequences of probabilistic models built by hBOA on one or more instances of the considered problem class. Then, the probabilistic models are processed to provide us with a method to bias probabilistic models in hBOA, which can be used to improve hBOA performance on future instances of the same problem type.

To provide examples of the application of these two approaches, we use two optimization problems: (1) 2D Ising spin glass with $\pm J$ couplings and periodic boundary conditions, and (2) MAXSAT.

The section starts by describing the test problems. We then describe how to bias model building using the probabilistic coincidence matrix. Next, we focus on biasing the model building using a distance threshold. Finally, the benefits of restricting model structure in hBOA are discussed.

4.1 Test Problems

This section describes test problems used in this study.

4.1.1 2D Ising Spin Glass with $\pm J$ couplings and periodic boundary conditions

Ising spin glasses are prototypical models for disordered systems. A simple model to describe a finite-dimensional Ising spin glass is typically arranged on a regular 2D or 3D grid where each node i corresponds to a spin s_i and each edge $\langle i,j\rangle$ corresponds to a coupling between two spins s_i and s_j. Each edge has a real value $J_{i,j}$ associated with it that defines the relationship between the two connected spins. To approximate the behavior of the large-scale system, periodic boundary conditions are often used that introduce a coupling between the first and the last elements in each row along each dimension.

For the classical Ising model, each spin s_i can be in one of two states: $s_i = +1$ or $s_i = -1$. Given a set of coupling constants $J_{i,j}$, and a configuration of spins C, the energy can be computed as

$$E(C) = \sum_{\langle i,j\rangle} s_i J_{i,j} s_j \ , \tag{2}$$

where the sum runs over all couplings $\langle i,j\rangle$.

Here the task is to find a spin configuration for a given set of coupling constants that minimizes the energy of the spin glass. The states with minimum energy are called *ground states*. The spin configurations are encoded with binary strings where each bit specifies the value of one spin (0 for a spin +1, 1 for a spin -1).

One generally analyzes a large set of random spin glass instances for a given distribution of the spin-spin couplings. For each spin glass instance, the optimization algorithm is applied and the results are analyzed. Here we consider the $\pm J$ spin glass, where each spin-spin coupling constant is set randomly to either $+1$ or -1 with equal probability. All instances of sizes up to 18×18 with ground states were obtained from S. Sabhapandit and S. N. Coppersmith from the University of Wisconsin who identified the ground states using flat-histogram Markov chain Monte Carlo simulations [5]. The ground states of the remaining instances were obtained from the Spin Glass Ground State Server at the University of Cologne [28].

4.1.2 MAXSAT

In MAXSAT the task is to find the maximum number of clauses which can be satisfied in a given propositional logic formula in conjunctive normal form. MAXSAT is an important problem in complexity theory and artificial intelligence because it is NP-complete and many other important problems can be mapped to MAXSAT in a straightforward manner. MAXSAT has also become popular in evolutionary computation and a number of researchers studied performance of various genetic and evolutionary algorithms on this class of problems [23, 9, 19, 2, 3].

MAXSAT problems are often expressed with logic formulas in conjunctive normal form with clauses of length at most k; formulas in this form are called k-CNF formulas. An interpretation of propositions assigns each proposition either true or false. The task is to find an interpretation that maximizes the number of satisfied clauses in the formula. MAXSAT for k-CNF formulas is NP-complete for any $k \geq 2$. hBOA encodes the interpretations with binary strings where each bit specifies an assignment of one proposition (0 for false, 1 for true) and the fitness is equal to the number of satisfied clauses.

In this paper we will consider instances of combined-graph coloring translated into MAXSAT [7], which are especially interesting because they are often difficult for standard MAXSAT heuristics such as WalkSAT [17] and because they represent a class of problems where randomness is combined with structure [7]. The graph-coloring problem instances were created by combining regular ring lattices and random graphs with a specified number of neighbors [7]. All tested instances of combined-graph coloring were downloaded from the Satisfiability Library SATLIB [12].

4.2 Biasing Models Using the Probability Coincidence Matrix

For this method we first compute what we will call a *probability coincidence matrix* (PCM) of size $n \times n$ where n is the string length. We denote the matrix by P and the elements of this matrix by P_{ij} where $i, j \in \{1 \dots n\}$. The value P_{ij} is defined as the proportion of probabilistic models in which ith and jth string positions are connected (in either of the two possible directions). The matrix P is thus symmetric. To compute the elements of P, we parse all available probabilistic models and use a counter for each position in the matrix which is incremented by 1 for each model that contains a connection between the ith and jth string positions. The value of P_{ij} is then set to the ratio of the resulting value of the counter and the total number of probabilistic models available on input. For example, if for any $i \neq j$, 25% of the available probabilistic models contain a connection between i and j, then $P_{ij} = 0.25$.

Of note is that runs of longer length (with respect to the number of iterations of the main hBOA loop) will be represented more strongly in the PCM than runs of shorter length. While we could have implemented a method that weighed all runs equally, it is not clear to us that this would always be beneficial. Not all runs are equal, some similar problems are harder than others, so it could be beneficial to have certain runs more strongly represented. Additionally, with this method, P_{ij} represents the actual percentage of models that connected i and j, whereas with other methods the meaning of P_{ij} would be more abstract.

Once we have computed the PCM, we can use this matrix to bias model building in future problems of similar type by only allowing edges between nodes that are contained in at least some percentage p_{min} of the provided sample models. For example, we can restrict the models by allowing only edges that appear in at least 20% of the sample models, disallowing a direct conditional dependence between any i and j for which $P_{ij} < p_{min} = 0.2$.

The main advantage of using PCM to restrict model structures over user-defined model restrictions based on prior problem-specific knowledge is that the approach proposed here is applicable automatically and the only parameter that needs to be specified by the user is the threshold p_{min}.

4.3 Biasing Models Using Distance

The PCM-based approach presented above has one main disadvantage, which restricts its utility in practice. Specifically, it is only applicable when the structure of the underlying problem does not change much between different problem instances. While this assumption is clearly satisfied in 2D Ising spin glasses, it does not hold in other important classes of problems. Nonetheless, both in the finite-dimensional Ising spin glasses as well as in other important classes of challenging problems—for example in MAXSAT and the minimum vertex cover—we may define a distance metric between different decision variables in the problem which should loosely correspond to the strength of the dependence between these variables. For example, in 2D spin glasses, we know that the shorter the shortest path between two spins is, the less frequent the dependencies between these spins are [11]; therefore, when studying models obtained by hBOA, it should be beneficial to use the distance metric based on the shortest distance between two spins across the 2D lattice. In MAXSAT, the distance of two Boolean variables may be defined as 1 for any two variables located in the same clause and for any other pair of variables, we may recursively define the distance between these two variables as the minimum distance between these variables passing through any other variable.

Given the distance metric , we can then process the given probabilistic models with respect to the distance. The result of such an analysis may be encoded by a vector Q that stores the proportion Q_d of dependencies at a specific distance d or a shorter one when averaging over all available models. More specifically, for any potential distance d, we first parse all models and compute the overall number of dependencies between variables at distance d or less; then, we compute Q_d by dividing the resulting count by the overall number of dependencies discovered.

One way to process the resulting vector is to define a threshold q_{min}, which defines the minimum value of Q_d for any distance d. That means that dependencies between variables at distance d for which $Q_d < q_{min}$ would be strictly disallowed.

4.4 Why are Model Restrictions Beneficial?

There are two main benefits of restricting model structure in hBOA and other multivariate EDAs. First of all, if we restrict model structures by disallowing some dependencies, the model building becomes significantly faster because the model-building algorithm has to examine much fewer dependencies. This can lead to substantial reduction in time complexity of model building, as is supported by experimental results in section 5. Second, if we restrict the probabilistic models by disallowing edges which are indeed unnecessary, then the search may become more effective.

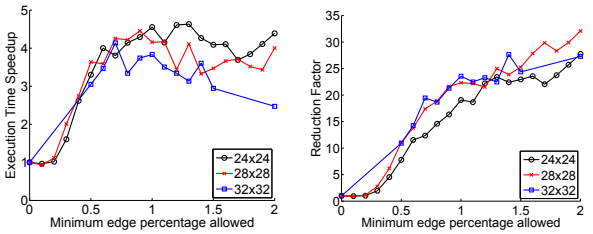
The following section provides a thorough empirical analysis of the above two approaches to automatically restricting model structure on 2D Ising spin glasses and MAXSAT.

5. EXPERIMENTS

This section covers the experiments using the two proposed approaches to biasing hBOA model building on 2D Ising spin glasses and MAXSAT. Since in MAXSAT, problem structure of different problem instances may vary substantially, for this class of problems we only consider the approach based on the distance metric. First the parameter settings and the experimental setup are discussed. The results are then presented.

5.1 Parameter Settings

For all problem instances, bisection [25, 17] was used to determine the minimum population size to ensure convergence to the global optimum in 5 out of 5 independent runs, with the results averaged over the 5 runs. The number of generations was upper bounded according to preliminary ex-



(a) Execution Speedup (b) Examination Reduction

Figure 1: Execution time speedups and reduction in the number of bits examined in model building based on model restrictions

periments and hBOA scalability thoery [22] by $n/4$ where n is the number of bits in the problem. Each run of hBOA is terminated when the global optimum has been found (success) or when the upper bound on the number of generations has been reached without discovering the global optimum (failure).

5.2 PCM Model Bias on 2D Spin Glass

While building a PCM is relatively straightforward, determining a threshold p_{min} for cutting off edges is not. To get a better idea of how p_{min} influences the benefits of restricting models with PCM, we considered a range of problem sizes from 16×16 (256 spins) to 32×32 (1024 spins). For each problem size, we used 100 random instances. In order to not use the same problem instances to both learn PCM as well as validate the resulting bias on model structure, we used 10-fold crossvalidation. For each problem size, the 100 instances were divided into 10 equally sized subsets and in each step of the crossvalidation, we used 9 of these 10 subsets to learn PCM and tested the resulting bias on model building on the remaining subset of instances. This was repeated 10 times, always leaving one subset of instances for validation. In this manner, the validation was done on different instances than those used for learning PCM and each subset of instances was used for validation.

Figure 1a shows the average execution-time speedup obtained from the 10-fold crossvalidation for four different problem sizes with varying threshold p_{min}. Note that the execution time does not include only the time spent in model building; it includes the overall time required for the entire execution of hBOA. The threshold for cutting of model dependencies was varied from $p_{min} = 0$ (no restrictions) to some maximum value, where the maximum value was set in order to ensure that no instances in the validation set become infeasible within reasonable computational time. We see that for all problem sizes, execution-time speedups of about 4–4.5 were obtained, which is a substantial speedup. It can be also be seen that the optimal value of p_{min} decreases with problem size. Nonetheless, even when the value of p_{min} is twice as large as the optimal one, the speedups still remain substantial. As problem size increases, the optimal speedup stays nearly constant (in the range of 4–4.5).

It is clear that the execution-time speedups obtained are caused by model building. But how are our restrictions affecting model-building? To quantify the effects of model building restrictions on the complexity of model building, we record the number of bits that must be checked to update model parameters during the entire model building procedure as this is the primary factor affecting time complexity

Size	Exec. speedup	p_{min}	% Total Dep.
256 (16×16)	3.89	2.0	6.4%
324 (18×18)	4.37	1.1	8.7%
400 (20×20)	4.34	2.0	7.0%
484 (22×22)	4.61	1.0	6.3%
576 (24×24)	4.63	1.3	4.6%
676 (26×26)	4.62	1.1	4.7%
784 (28×28)	4.45	0.9	5.4%
900 (30×30)	4.93	0.5	8.1%
1024 (32×32)	4.14	0.7	5.5%

Table 1: Optimal speedup and the corresponding PCM threshold p_{min} as well as the percentage of total possible dependencies that were considered for the 2D Ising spin glass.

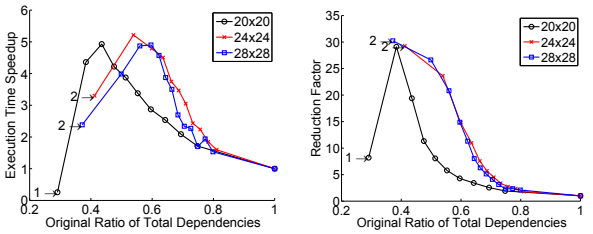
of model building in hBOA. If we restrict the number of potential edges ending in any particular node, after adding an edge into this node, the number of examined bits reduces by the same factor. For Bayesian networks with local structures, limiting the model structure leads to a decrease in the number of potential splits.

Figure 1b shows the average factor of decrease in the number of bits examined during the model building with respect to the value of the PCM cutoff threshold p_{min}. As we can see, model restrictions based on PCM dramatically decrease the number of bits that must be examined. One thing of note is that even after we reach the point of maximum execution-time speedup, the number of bits examined further decreases with increasing p_{min}. This indicates that after the optimal cutoff, other factors start to weigh more heavily on the execution time and the model-building bias becomes too restrictive.

Table 1 shows the best speedups obtained, the percentage of total possible dependencies considered by hBOA, and the corresponding cutoff values for all problem sizes examined. The results show nearly the same maximum speedup of about 4–4.5 for all problem sizes. The results also indicate that as the problem size increases the cutoff values must be slightly decreased to achieve optimal speedup. We believe that the reason for this is that for larger problems, the total number of dependencies increases, and to ensure that a sufficient fraction of dependencies is considered, the cutoff threshold must decrease. We also see that even as the cutoff threshold increases, hBOA only needs to consider a small percentage of the total dependencies. In fact this percentage is remarkably similar for the different problem sizes.

5.3 Distance-Based Bias on 2D Spin Glass

As was argued in section 4, restricting model structures should lead to significant speedups (which was supported by experiments using PCM-based model bias). Nonetheless, it was also argued that restricting models based on PCM can be expected to be ineffective if different problem instances vary in structure substantially. That is why another approach was suggested which is based on imposing a distance metric on the problem decision variables and restricting dependencies of large distances using some threshold. This section tests this approach on the 2D Ising spin glass with the distance metric defined as the minimum number of couplings one must pass to travel from one spin to another.



(a) Execution Speedup (b) Examination Reduction

Figure 2: Execution time speedups and reduction in the number of bits examined in model building based on distance restrictions on 2D Ising spin glasses

While it is clear that the probabilistic models discovered by hBOA contain mostly dependencies at shorter distances, setting an appropriate threshold for the maximum distance of dependencies remains a challenge. If the distances are restricted too severely, the bias on the model building may be too strong to allow for sufficiently complex models; this was supported also with results in ref. [11]. On the other hand, if the distances are not restricted sufficiently, the benefits of using this approach may be negligible.

To explore this tradeoff, we considered spin glass instances of sizes 16×16 to 28×28, 100 random instances for each problem size. Then, for each instance, we ran a series of experiments with dependencies restricted by the maximum distance, which was varied from 1 to half the maximum achievable distance (for example, for 20×20 spin glasses, we ran experiments only allowing dependencies between spins of a maximum distance from 1 to 10). When the restrictions on model structure were too strong, some of the instances would not converge even for extremely large population sizes ($N = 512000$); the results in these cases were omitted.

Figure 2a shows the execution-time speedup with model complexity restricted by the maximum distance. The horizontal axis is the ratio of the number of dependencies in the original runs that matched that restriction compared to the overall dependencies. The maximum distance allowed is shown as a label for some selected points in the graph.

The results show that restricting models by maximum distance results in significant speedups of about 4.3–5.2; in fact, the optimal speedups obtained are better than those obtained with the PCM-based model bias. In agreement with results in ref. [11], we also see that as the problem size increases, dependencies at larger distance should be allowed for maximum speedup. Nonetheless, the speedups obtained seem to be again nearly independent of problem size, just like for the results obtained with PCM.

Just as we did for the PCM-based approach, we also examined the effects of the distance-based model restriction on the number of bits that had to be examined during the entire model-building procedure. Figure 2b shows the factor by which this quantity decreases with various thresholds on the maximum distance. The results show a dramatic drop in the number of bits that must be examined with small values of the maximum distance, by as much as a factor of 30. They also show that this maximum decrease is maintained as problem size increases.

Table 2 shows the best speedups, the corresponding maximum distance threshold, and the percentage of total possible dependencies that were considered by hBOA for all problem sizes. We can see that hBOA is only considering a small percentage of the possible dependencies for these cutoffs. We can also clearly see that as problem size increases, the maximum speedup stays nearly the same, indicating that our speedups will scale to larger problems.

Size	Exec. speedup	q_{min}	% Total Dep.
256 (16×16)	4.29	2	4.7%
400 (20×20)	4.93	3	6.0%
576 (24×24)	5.22	3	4.1%
784 (28×28)	4.91	5	7.6%

Table 2: Distance cutoff runs with their best speedups by distance as well as the percentage of total possible dependencies that were considered for 2D Ising spin glass

A comparison of tables 1 and 2 reveals that while the speedups in the two cases are similar, the speedups obtained with the distance-based model restriction are slightly better than with the PCM-based approach. We also see the same pattern of hBOA considering approximately the same percentage of total dependencies using each of the methods.

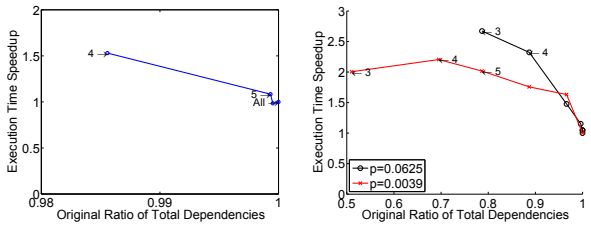
5.4 Distance-Based Bias on MAXSAT

In section 5.3 we saw that restricting model structure by distance leads to significant speedups of hBOA on 2D Ising spin glasses. Can this approach be applied to other problems? In this section we will attempt to answer this question by looking at combined-graph coloring problems encoded as instances of the MAXSAT problem.

To restrict models by distance in MAXSAT we first define a distance metrix. The distance between any two Boolean variables is defined as 1 for any two variables in the same clause. For any other pair of variables their distance is recursively defined as the minimum distance between these variables passing through any other variable. If there is no path between two propositions their distance is defined as the number of propositions.

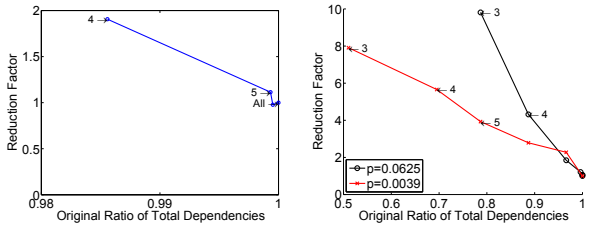
While this distance metric is relatively straightforward to implement, setting an adequate threshold on distances to maximize the speedups is not. We would certainly expect that many dependencies would be between propositions that share a clause but restricting hBOA to only consider dependencies between propositions in the same clauses would almost certainly be too severe of a restriction. Yet just as with spin glasses, if we do not restrict model structure substantially then the gains will be negligible.

To examine this tradeoff, we looked at instances of MAXSAT for graph coloring of combined graphs [7] with $p = 0$, $p = 2^{-4}$ and $p = 2^{-8}$. For each value of p, we considered 100 random instances where all 100 instances were 5-colorable, and contained 500 propositions and 3100 clauses. Then, for each of these instances we ran experiments with dependencies restricted by the maximum distance, which was varied from 1 to the maximum distance found between any two propositions (for example, for $p = 2^{-4}$ we ran experiments using a maximum distance from 1 to 9). For some instances with $p = 1$ the maximum distance was 500, indicating that there was no path between some pairs of propositions. On the tested problems, small distance restrictions (restricting to only distance 1 or 2) were sometimes too restrictive and some instances would not be solved even with



(a) $p = 1$ (b) $p = 2^{-4}, 2^{-8}$

Figure 3: Execution-time speedup with model restrictions based on the maximum distance on MAXSAT for different values of p



(a) $p = 1$ (b) $p = 2^{-4}, 2^{-8}$

Figure 4: Factor by which the number of bits in model building decreases with the model restrictions based on maximum distance on MAXSAT for different values of p.

extremely large population sizes ($N = 512000$); in these cases the results were omitted.

Figure 3 shows the execution-time speedup of hBOA on MAXSAT with model complexity restricted by the maximum distance. The horizontal axis stands for the ratio of the number of dependencies with a specific distance bound and the total number of dependencies in the original, unrestricted runs. The maximum distance allowed is shown as a label for selected points in the graph.

The results show that the speedups obtained by restricting model structures by distance vary with the amount of structure in the considered problem instances. For $p = 1$, problem instances have very little structure and we see that only one distance threshold led to a significant speedup (of about 1.5). For $p = 2^{-4}$ we see a maximum speedup of about 2.5 and for the most structured problems with $p = 2^{-8}$ we see a maximum speedup of about 2.2. We also see that as the amount of structure increases that a wider variety of distance thresholds lead to speedups. For $p = 1$ only one threshold led to a substantial improvement. However, as p decreases, we get a wider band of thresholds that lead to noticeable improvements. We also see that as p increases the number of dependencies at small distances increases rapidly—for example, for $p = 1$ we see that over 98% dependencies were of distance 4 or less, while for $p = 2^{-8}$, only 72% dependencies were at distance 4 or less.

We also examined the effects of the MAXSAT distance-based restrictions on the number of bits that had to be examined during the entire model-building procedure. Figure 4 shows the factor by which this quantity decreases with various thresholds on the maximum distance. As in the previous figure, the distance restrictions are labeled on the graph with arrows. As in the execution time results, we see that our gains are smaller for $p = 1$ but are much higher for the other two values of p.

p	Exec. speedup	q_{min}	% Total Dep.
$p = 1$	1.53	4	97.7%
$p = 2^{-4}$	2.67	3	29.6%
$p = 2^{-8}$	2.20	4	28.4%

Table 3: Distance cutoff runs with their best speedups by distance as well as the percentage of total possible dependencies that were considered for MAXSAT

Table 3 shows the best speedups, the corresponding maximum distance threshold and the percentage of possible dependencies that were considered by hBOA for all values of p. While the results are less impressive than for the 2D Ising spin glasses, speedups are obtained for all values of p examined, indicating that even for MAXSAT problems with some structure, distance restrictions can improve the performance of hBOA. In contrast to the results on 2D spin glasses, we see that hBOA needs to consider a much larger proportion of possible dependencies. In fact, in the $p = 1$ case, almost all dependencies were considered, and yet the results still showed an almost 50% improvement in efficiency. In the remaining two cases, the results were very similar, with approximately 30% of all possible dependencies considered.

6. SUMMARY AND CONCLUSIONS

Besides providing the user with the global optimum or at least its good approximation, EDAs also provide a series of probabilistic models that hold a great deal of information about the problem. If one wants to solve more problem instances of similar type, using this readily available source of problem-specific knowledge should allow the user to improve performance of EDAs on future problem instances of this type. This study takes the first step in exploiting this information and shows that restricting model building in hBOA based on the results from previous runs on similar problems leads to substantial speedups. Two approaches to restricting probabilistic models based on previous runs were proposed and tested, yielding the speedup of about 4 to 5 on the 2D Ising spin glass and the speedup of about 1.5 to 2.5 on MAXSAT depending on the amount of structure in the problem.

While this study considered only two specific classes of problems—2D Ising spin glasses and MAXSAT—the proposed approaches can be adapted to any problem where either (1) problem structure does not vary significantly between different problem instances or (2) one can impose a distance metric on problem variables so that variables located at greater distances are less likely to be connected in the probabilistic model. Some examples of such problems classes include the 3D spin glass and various problems from graph theory (such as the minimum vertex cover and graph bipartitioning). The proposed approaches then provide a principled way to control model bias based on previously discovered probabilistic models without requiring the user to manually design such a bias. The proposed techniques can also be adapted to other EDAs based on multivariate probabilistic models, such as ECGA.

While any efficiency enhancement technique is useful on its own right, combining multiple efficiency enhancement techniques often yields multiplicative speedups [8, 26]. For example, sporadic model building or parallel model building can be used in combination with the techniques proposed

in this paper, further improving hBOA performance. This should allow the practitioners to further increase the size of problems feasible with EDAs and solve problems unsolvable with the current methods.

There are three key areas for future research on this topic. First of all, the proposed techniques should be tested in other classes of important problems, such as the graph bipartitioning or the minimum vertex cover problems. Second, the process of choosing appopriate thresholds to achieve maximum speedups should be made more automatic so that the user does not have to rely on the trial-and-error approach to setting these parameters. Finally, the proposed approaches should be improved by exploring other techniques to bias model building and extended to deal with other problem types.

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 Material Command, USAF, under grant FA9550-06-1-0096, 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. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the Air Force Office of Scientific Research, or the U.S. Government.

7. REFERENCES

[1] S. Baluja. Incorporating a priori knowledge in probabilistic-model based optimization. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable optimization via probabilistic modeling: From algorithms to applications*, pages 205–219. Springer, 2006.

[2] D. Boughaci and H. Drias. A performance comparison of evolutionary meta-heuristics and solving MAX-SAT problems. In A. Okatan, editor, *International Conference on Computational Intelligence*, pages 379–383. International Computational Intelligence Society, 2004.

[3] A. E. I. Brownlee, J. A. W. McCall, and D. F. Brown. Solving the MAXSAT problem using a multivariate EDA based on markov networks. In P. A. N. Bosman, editor, *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2007)*, pages 2423–2428, London, United Kingdom, 7-11 July 2007. ACM Press.

[4] D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. Technical Report MSR-TR-97-07, Microsoft Research, Redmond, WA, 1997.

[5] P. Dayal, S. Trebst, S. Wessel, D. Würtz, M. Troyer, S. Sabhapandit, and S. Coppersmith. Performance limitations of flat histogram methods and optimality of Wang-Langdau sampling. *Physical Review Letters*, 92(9):097201, 2004.

[6] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In M. I. Jordan, editor, *Graphical models*, pages 421–459. MIT Press, 1999.

[7] I. Gent, H. H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. *Proceedings of the American Association of Artificial Intelligence (AAAI-99)*, pages 654–660, 1999.

[8] D. E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer, 2002.

[9] J. Gottlieb, E. Marchiori, and C. Rossi. Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation*, 10(1):35–50, 2002.

[10] G. R. Harik. Finding multimodal solutions using restricted tournament selection. *International Conference on Genetic Algorithms (ICGA-95)*, pages 24–31, 1995.

[11] M. Hauschild, M. Pelikan, C. Lima, and K. Sastry. Analyzing probabilistic models in hierarchical boa on traps and spin glasses. *Genetic and Evolutionary Computation Conference (GECCO-2007)*, I:523–530, 2007.

[12] H. H. Hoos and T. Stutzle. Satlib: An online resource for research on sat. *SAT 2000*, pages 283–292, 2000. SATLIB is available online at www.satlib.org.

[13] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the principles and applications of decision analysis*, volume II, pages 721–762. Strategic Decisions Group, Menlo Park, CA, 1981.

[14] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.

[15] H. Mühlenbein and T. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal on Approximate Reasoning*, 31(3):157–192, 2002.

[16] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, San Mateo, CA, 1988.

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

[18] M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. *Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 511–518, 2001.

[19] M. Pelikan and D. E. Goldberg. Hierarchical BOA solves Ising spin glasses and MAXSAT. *Genetic and Evolutionary Computation Conference (GECCO-2003)*, II:1275–1286, 2003.

[20] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[21] M. Pelikan, K. Sastry, and E. Cantú-Paz, editors. *Scalable optimization via probabilistic modeling: From algorithms to applications*. Springer-Verlag, 2006.

[22] M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258, 2002.

[23] S. Rana and D. L. Whitley. Genetic algorithm behavior in the MAXSAT domain. *Parallel Problem Solving from Nature*, pages 785–794, 1998.

[24] K. Sastry. Efficient atomic cluster optimization using a hybrid extended compact genetic algorithm with seeded population. IlliGAL Report No. 2001018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2001.

[25] 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.

[26] K. Sastry, M. Pelikan, and D. E. Goldberg. Efficiency enhancement of estimation of distribution algorithms. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, pages 161–185. Springer, 2006.

[27] J. Schwarz and J. Ocenasek. A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning, 2000. Personal communication.

[28] Spin Glass Ground State Server. `http://www.informatik.uni-koeln.de/ls\_juenger/research/sgs/sgs.html`, 2004. University of Köln, Germany.