Missouri Estimation of Distribution Algorithms Laboratory

Using previous models to bias structural learning in the hierarchical BOA

Mark Hauschild, Martin Pelikan, Kumara Sastry, and David E. Goldberg

MEDAL Report No. 2008003

January 2008

Abstract

Estimation of distribution algorithms (EDAs) are stochastic optimization techniques that explore the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions. While the primary goal of applying EDAs is to discover the global optimum or at least its accurate approximation, besides this, any EDA provides us with a sequence of probabilistic models, which in most cases 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 problem-specific information has been practically 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 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 hBOA runs on similar problems. We show that the proposed methods lead to substantial speedups and argue that the methods should work well in other applications that require solving a large number of problems with similar structure.

Keywords

Hierarchical BOA, efficiency enhancement, learning from experience, probabilistic model, model structure, model complexity, estimation of distribution algorithms.

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/

Using previous models to bias structural learning in the hierarchical BOA

Mark Hauschild
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Math and Computer Science, 320 CCB
University of Missouri at St. Louis
One University Blvd., St. Louis, MO 63121
mwh308@umsl.edu

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (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

Kumara Sastry
Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign, Urbana IL 61801
ksastry@uiuc.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign, Urbana IL 61801
deg@uiuc.edu

Abstract

Estimation of distribution algorithms (EDAs) are stochastic optimization techniques that explore the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions. While the primary goal of applying EDAs is to discover the global optimum or at least its accurate approximation, besides this, any EDA provides us with a sequence of probabilistic models, which in most cases 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 problem-specific information has been practically 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 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 hBOA runs on similar problems. We show that the proposed methods lead to substantial speedups and argue that the methods should work well in other applications that require solving a large number of problems with similar structure.

Keywords: Hierarchical BOA, efficiency enhancement, learning from experience, probabilistic

model, model structure, model complexity, estimation of distribution algorithms.

1 Introduction

One of the key goals in genetic and evolutionary computation is the design of competent genetic algorithms (GAs) that can solve hard problems scalably and reliably (Goldberg, 2002). Much work has been done in this regard, resulting in algorithms that can solve broad classes of problems in a robust and scalable manner. Yet this is not the complete story because even if we can solve a problem in low-order polynomial time, the problem may still be intractable. This can happen for many reasons, among them extremely complex fitness functions and an enormously large number of variables to optimize. To solve such challenging problems, practitioners are often forced to use additional efficiency enhancement techniques (Goldberg, 2002; Sastry, Pelikan, & Goldberg, 2006)—such as parallelization and hybridization—to speed up GAs, and a number of efficiency enhancement techniques have been proposed for this purpose in the past.

Estimation of distribution algorithms (EDAs) (Baluja, 1994; Larrañaga & Lozano, 2002; Pelikan, Goldberg, & Lobo, 2002; Pelikan, Sastry, & Cantú-Paz, 2006) are among the most powerful and generally applicable genetic and evolutionary algorithms, in which traditional variation operators of genetic algorithms—such as crossover and mutation—are replaced 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 (Larrañaga & Lozano, 2002; Pelikan, Sastry, & Cantú-Paz, 2006), 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 (Schwarz & Ocenasek, 2000; Baluja, 2006), 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 the hierarchical Bayesian optimization algorithm (hBOA). Section 3 discusses efficiency enhancement techniques for genetic and evolutionary algorithms with a particular emphasis on efficiency enhancement of 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)

Estimation of distribution algorithms (EDAs) (Baluja, 1994; Mühlenbein & Paaß, 1996; Larrañaga & Lozano, 2002; Pelikan, Goldberg, & Lobo, 2002)—also called probabilistic model-building genetic algorithms (PMBGAs) (Pelikan, Goldberg, & Lobo, 2002; Pelikan, 2005) and iterated density estimation algorithms (IDEAs) (Bosman & Thierens, 2000)—replace standard crossover and mutation

operators of genetic and evolutionary algorithms by building a probabilistic model of selected solutions and sampling the built model to generate new candidate solutions. The hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan & Goldberg, 2003b; Pelikan, 2005) is an EDA that uses Bayesian networks to represent the probabilistic model and incorporates restricted tournament replacement (Harik, 1995) for effective diversity maintenance. This section outlines hBOA and briefly discusses Bayesian networks, which are used to guide the exploration of the search space in hBOA.

2.1 Basic hBOA Procedure

hBOA 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 the threshold $\tau = 50\%$, which selects the best half of the current population based on the objective function.

After selecting the promising solutions, hBOA builds a Bayesian network (Howard & Matheson, 1981; Pearl, 1988) with local structures as a model for these solutions. New solutions are generated by sampling the built network. The new solutions are then incorporated into the original population using restricted tournament replacement (RTR) (Harik, 1995), which ensures effective diversity maintenance. RTR with window size $w > 1$ incorporates each new candidate solution X into the original population using the following three steps: (1) Randomly select a subset W of w candidate solutions from the original population. (2) Let Y be a solution from W that is most similar to X (based on genotypic distance). (3) Replace Y with X if X is better; otherwise, discard X. A robust rule of thumb is to set $w = \min\{n, N/20\}$, where n is the number of decision variables in the problem and N is the population size (Pelikan, 2005).

The next iteration is executed unless some predefined termination criteria are met. For example, the run can be terminated when the maximum number of generations is reached or the entire population consists of copies of the same candidate solution. For more details about the basic hBOA procedure, see Pelikan and Goldberg (2001) or Pelikan (2005).

2.2 Bayesian Networks

Bayesian networks (Pearl, 1988; Howard & Matheson, 1981) combine graph theory, probability theory and statistics to provide a flexible and practical tool for probabilistic modeling and inference. BOA and hBOA use Bayesian networks to model promising solutions found so far and sample new candidate solutions. A Bayesian network 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.

Mathematically, a Bayesian network with n nodes encodes a joint probability distribution of n random variables $X_1, X_2, \ldots, X_n$:

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

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).

In addition to encoding direct conditional dependencies, a Bayesian network may also encode a number of conditional independence assumptions. More specifically, a Bayesian network encodes the assumption that each variable X_i is conditionally independent of its predecessors in an ancestral ordering of variables given Π_i, where the ancestral ordering orders the variables so that for all i the variables in Π_i precede X_i. Since Bayesian networks are acyclic, there always exists at least one ancestral ordering of the variables.

hBOA uses Bayesian networks with local structures in the form of dependency trees (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999). That means that the conditional probabilities for each variable are stored in a decision tree, allowing a more efficient representation of conditional dependencies and a more powerful model-building procedure. For more details on learning and sampling Bayesian networks with local structures, see Chickering, Heckerman, and Meek (1997) and Pelikan (2005).

3 Efficiency Enhancement Techniques

Over the last two decades or so, there has been much work in designing *competent* genetic algorithms (GAs) (Goldberg, 2002), that is, GAs that can solve hard problems quickly, reliably and accurately. A number of such competent GAs have been proposed and shown to solve broad classes of challenging problems of bounded difficulty reliably with only low-order polynomial time complexity (Goldberg, Korb, & Deb, 1989; Harik, 1999; Pelikan, Goldberg, & Cantú-Paz, 1999; Etxeberria & Larrañaga, 1999; Mühlenbein & Mahnig, 1999; Goldberg, 2002). However, solving broad classes of difficult problems in low-order polynomial time is not the end of the story—even when we can solve a problem in low-order polynomial time, the number of variables may be too large or the evaluations may take too long for the computation to be feasible with the available computational resources. One of the approaches to alleviating this problem is to incorporate various efficiency enhancement techniques (Goldberg, 2002; Sastry, Pelikan, & Goldberg, 2006; Pelikan, 2005)—such as parallelization and hybridization—into GAs and EDAs with the goal of speeding up these algorithms. In EDAs, there are two potential bottlenecks to address with efficiency enhancement techniques: (1) fitness evaluation and (2) model building.

Efficiency enhancement techniques for EDAs can be divided into the following categories (Pelikan, 2005):

1. Parallelization.
2. Evaluation relaxation.
3. Hybridization.
4. Time continuation.
5. Sporadic and incremental model building.
6. Incorporating problem-specific knowledge and learning from experience.

In the remainder of this section we briefly review each of these approaches, with the emphasis on efficiency enhancement techniques for EDAs. For a more detailed discussion of efficiency enhancement techniques for EDAs, please see Sastry, Pelikan, and Goldberg (2006).

3.1 Parallelization

One of the most straightforward approaches to speed up genetic and evolutionary algorithms (GEAs) is to distribute the computation among multiple processors. While this can be done in different ways (Cantú-Paz, 2000), it is clear that the very nature of most GEAs makes them efficiently parallelizable. First of all, fitness evaluations in any GEA are usually independent and they can be thus distributed among different processors in a straightforward manner using the master-slave architecture, leading to substantial speedups for problems with costly fitness evaluation (Cantú-Paz, 2000).

In EDAs, it may also be important to parallelize model building itself, especially when complex probabilistic models with multivariate interactions—such as Bayesian networks of hBOA—are used. Nonetheless, model building may also be parallelized in simpler scenarios, such as in the compact genetic algorithm; in fact, one of the most impressive results in parallelization of EDAs is the efficient parallel implementation of the compact genetic algorithm, which was successfully applied to a noisy optimization problem with over one billion decision variables (Sastry, Goldberg, & Llorà, 2007). Several approaches to parallelizing model building in advanced EDAs with multivariate models have also been proposed (Ocenasek, 2002; Ocenasek, Cantú-Paz, Pelikan, & Schwarz, 2006; Mendiburu, Miguel-Alonso, & Lozano, 2006).

3.2 Evaluation Relaxation

In some problems, fitness evaluation of each candidate solution takes considerable time. For example, each fitness evaluation may depend on a complex finite element analysis or it may require a time-consuming physical simulation. While parallelization may be used to reduce the burden of expensive fitness evaluation, sometimes parallelization of fitness evaluations is insufficient or a sufficiently powerful parallel computer is just not available. Nonetheless, to further improve performance of GEAs with expensive fitness evaluation, it is sometimes possible to eliminate some of the fitness evaluations by using approximate models of the fitness function, which can be evaluated much faster than the actual fitness function. Efficiency enhancement techniques based on this principle are called evaluation relaxation techniques (Goldberg, 2002; Smith et al., 1995; Sastry et al., 2001a; Pelikan & Sastry, 2004; Sastry et al., 2004).

There are two basic approaches to evaluation relaxation: (1) endogenous models (Smith, Dike, & Stegmann, 1995; Sastry, Goldberg, & Pelikan, 2001a; Pelikan & Sastry, 2004; Sastry, Pelikan, & Goldberg, 2004) and (2) exogenous models (Sastry, 2001b; Albert, 2001). With endogenous models, the fitness values for some of the new candidate solutions are estimated based on the fitness values of the previously generated and evaluated solutions. With exogenous models, a faster but less accurate surrogate model is used for some of the evaluations, especially for those early in the run. Of course, the two approaches can be combined to maximize the benefits, although doing this may not always be straightforward.

Surrogate fitness models can be used for evaluation relaxation in any GEA regardless of the operators and representation used. Therefore, there is not much difference between using surrogate fitness models in standard GAs and EDAs. On the other hand, building accurate endogenous models becomes more effective in EDAs than in standard GAs due to the use of an explicit probabilistic model of promising candidate solutions, which can be extended to provide a quick and relatively accurate approximation of the fitness function.

The first study (Sastry, Goldberg, & Pelikan, 2001b) that incorporated endogenous models into EDAs considered the univariate marginal distribution algorithm (UMDA) (Mühlenbein & Paaß, 1996). In UMDA, the distribution of selected solutions is modeled with the probability vector,

which stores the probability of a 1 at any given position. To estimate fitness, the probability vector was extended to also store statistics on the average fitness of all solutions with a 0 or a 1 in any string position. These data were then used to estimate fitness of new solutions. Sastry et al. (2001b) showed that optimally the number of evaluations of the actual fitness function can be reduced by about 20% if approximately half of the new candidate solutions are evaluated with the endogenous model whereas the remaining ones are evaluated using the actual fitness function.

The endogenous-model approach used in UMDA was later extended to the extended compact genetic algorithm (ECGA) (Harik, 1999; Sastry, Pelikan, & Goldberg, 2004) and the Bayesian optimization algorithm (Pelikan & Sastry, 2004). Similarly as in UMDA, in both ECGA as well as BOA the probabilistic models were extended with additional data for estimating fitness. As a result, in some cases, speedups of 30 or more were obtained (Pelikan & Sastry, 2004).

3.3 Hybridization

In many real-world applications, GEAs are combined with other optimization algorithms. Typically, simple and fast local search techniques—which can quickly locate the closest local optimum—are incorporated into a GEA, reducing the problem of finding the global optimum to that of finding only the basin of attraction of the global optimum. As an example, consider the simple deterministic hill climber (DHC), which takes a candidate solution represented by a binary string and keeps performing single-bit flips on the solution that lead to the greatest improvement in fitness. Using DHC to improve every solution generated by hBOA was shown to significantly improve hBOA performance on the problem of finding ground states of Ising spin glasses (both 2D and 3D) as well as on MAXSAT (Pelikan & Goldberg, 2003a).

While even incorporating simple local search techniques can lead to significant improvements in time complexity of various GEAs, sometimes more advanced optimization techniques are available that are tailored to the problem being solved. As an example, consider cluster exact approximation (CEA) (Hartmann, 1996), which can incorporated into hBOA or any other GEA (Pelikan & Hartmann, 2006) when solving the problem of finding ground states of Ising spin-glass instances arranged on finite-dimensional lattices. Unlike DHC, CEA can flip many bits at once, often yielding solutions very close to the global optimum after only a few iterations. With the use of CEA, the size of instances solvable with hBOA and other GEAs can be significantly increased (Pelikan & Hartmann, 2006).

The examples discussed above represented two extremes—while DHC performs very simple local search but it can be applied to any problem, CEA is much more effective than DHC but it is a specialized technique applicable to only a highly restricted class of problems (finite-dimensional spin glass models). Yet there is a middle ground between these extremes. Specifically, one can use the probabilistic models developed by advanced EDAs such as ECGA and BOA to design specialized local search techniques, which can significantly improve performance of these EDAs by exploiting problem structure encoded in the probabilistic model (Sastry & Goldberg, 2004a; Lima, Pelikan, Sastry, Butz, Goldberg, & Lobo, 2006).

3.4 Time Continuation

In time continuation, the goal is to maximize performance of GEAs by exploiting the trade-off between making more runs with a small population size and making fewer runs (or even only a single run) with a larger population size (Goldberg, 1999; Srivastava & Goldberg, 2001; Goldberg, 2002). To reinitialize the algorithm at the beginning of each run, the population from the previous run can be randomized using a continuation operator. One of the important results in this line of

research indicates that for ECGA on separable problems of bounded difficulty, if the population size is large enough for a sufficiently accurate model of the underlying problem decomposition, using a single iteration with a local searcher based on the probabilistic model outperforms standard ECGA (Sastry & Goldberg, 2004b).

3.5 Sporadic and Incremental Model-building

As we mentioned earlier, while in standard GEAs fitness evaluation is typically the computational bottleneck, in EDAs model building may be an even more important component to tackle with efficiency enhancement. This is especially important for EDAs that build complex multivariate probabilistic models, such as ECGA and hBOA.

Model building in ECGA, hBOA and other similar EDAs usually consists of two parts: (1) learning the structure and (2) learning the parameters of the identified structure. Typically, learning the model structure is much more complex than learning the parameters (Ocenasek & Schwarz, 2000; Pelikan, 2005). However, since the model structure is expected to not change much between consequent iterations, one way to speed up model building is to use sporadic model building, in which the structure is updated only once once in a while (Pelikan, Sastry, & Goldberg, 2006). This can lead to significant speedups even in problems that require the probabilistic model to change throughout the run.

Since the model structure is expected to not change much over time and making incremental changes to model structure is usually much simpler than building the structure from scratch, it may also be advantageous to change the model structure only incrementally without rebuilding the model from scratch in every iteration. This is the basic idea of incremental model building (Etxeberria & Larrañaga, 1999).

3.6 Incorporating Problem-Specific Knowledge and 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 (Schwarz & Ocenasek, 2000; Sastry, 2001a; Pelikan & Goldberg, 2003a) and (2) bias or restrict the model building procedure (Schwarz & Ocenasek, 2000; Mühlenbein & Mahnig, 2002; Baluja, 2006). 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 (Schwarz & Ocenasek, 2000; Pelikan & Goldberg, 2003a), or created in some way from high-quality solutions of smaller instances of the same problem (Sastry, 2001a).

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 (2001a) dramatically lowered the asymptotic complexity of ECGA by seeding the population with atomic configurations with one fewer atom. For example, before solving the problem with 10 atoms, the 9-atom problem

was first solved and then the initial population for the 10-atom problem was seeded with configurations obtained in the 9-atom case, each extended with a single atom, which was positioned randomly.

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 and Ocenasek (2000). 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, Schwarz and Ocenasek (2000) 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 scoring 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 and Mahnig (2002) 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 procedure.

Finally, Baluja (2006) 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 and Mahnig (2002), 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 and Mahnig (2002), 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. The latter two difficulties are less important with soft model restrictions based on specifying a prior network structure than with hard model restrictions which strictly disallow some dependencies.

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. If we want to solve more problems of similar type, information contained in probabilistic models may be used to design effective variation operators and efficiency enhancement techniques tailored to the specific problem type. 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 three optimization problems: (1) Concatenated traps of order 5, (2) 2D Ising spin glass with $\pm J$ couplings and periodic boundary conditions, and (3) 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 the test problems considered in this paper.

4.1.1 Trap-5: Concatenated 5-bit trap

In trap-5 (Ackley, 1987; Deb & Goldberg, 1991), the input string is first partitioned into independent groups of 5 bits each. This partitioning is unknown to the algorithm and it does not change during the run. A 5-bit fully deceptive trap function is applied to each group of 5 bits and the contributions of all trap functions are added together to form the fitness. The contribution of each group of 5 bits is computed as

$$trap_5(u) = \begin{cases} 5 & \text{if } u = 5 \\ 4 - u & \text{otherwise} \end{cases}, \tag{2}$$

where u is the number of 1s in the input string of 5 bits. The task is to maximize the function. An n-bit trap5 function has one global optimum in the string of all 1s and $(2^{n/5} - 1)$ other local optima. Traps of order 5 necessitate that all bits in each group are treated together, because statistics of lower order are misleading. Since hBOA performance is invariant with respect to the ordering of string positions (Pelikan, 2005), it does not matter how the partitioning into 5-bit groups is done, and thus, to make some of the results easier to understand, we assume that trap partitions are located in contiguous blocks of bits.

4.1.2 2D Ising Spin Glass with $\pm J$ couplings and periodic boundary conditions

Ising spin glasses are prototypical models for disordered systems and have played a central role in statistical physics during the last three decades (Binder & Young, 1986; Mezard, Parisi, & Virasoro, 1987; Fischer & Hertz, 1991; Young, 1998). 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

$$E(C) = \sum_{\langle i,j \rangle} s_i J_{i,j} s_j \ , \tag{3}$$

where the sum runs over all couplings $\langle i, j \rangle$. For a given spin configuration, couplings which have $s_i J_{i,j} s_j$ are called *satisfied*, unsatisfied else.

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).

In order to obtain a quantitative understanding of the disorder in a spin glass system introduced by the random spin-spin couplings, 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 (Dayal et al., 2004). The ground states of the remaining instances were obtained from the Spin Glass Ground State Server at the University of Cologne (Spin Glass Ground State Server, 2004).

To improve the performance of hBOA on 2D spin glass, we incorporate a deterministic hill climber (DHC) (see section 3) to improve quality of each evaluated solution. The local searcher is applied to every solution before it is evaluated.

4.1.3 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 (Rana & Whitley, 1998; Gottlieb, Marchiori, & Rossi, 2002; Pelikan & Goldberg, 2003a; Boughaci & Drias, 2004; Brownlee, McCall, & Brown, 2007).

MAXSAT problems are often expressed with logic formulae in conjunctive normal form with clauses of length at most k; formulae in this form are called k-CNF formulae. A CNF formula is

a *logical and* of clauses, where each clause is a *logical or* of literals. Each literal can either be a proposition or a negation of a proposition. An example of a 3-CNF formula with 4 propositions X_1, X_2, X_3, X_4 is

$$(X_4 \vee X_3) \wedge (X_1 \vee \neg X_2) \wedge (\neg X_4 \vee X_2 \vee X_3) \tag{4}$$

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. In the example from equation 4, the assignment setting all literals to true would satisfy all the clauses in the formula and therefore it would be one of the global optima of the corresponding MAXSAT problem. hBOA encodes the interpretations with binary strings where each bit specifies an assignment of one proposition (0 for false, 1 for true). Of note is that MAXSAT is NP complete for k-CNF if $k \geq 2$.

In this paper we will consider instances of combined-graph coloring translated into MAXSAT (Gent, Hoos, Prosser, & Walsh, 1999), which are especially interesting because they are often difficult for standard MAXSAT heuristics such as WalkSAT (Pelikan, 2005) and because they represent a class of problems where randomness is combined with structure (Gent, Hoos, Prosser, & Walsh, 1999). In graph coloring, the task is to color the vertices's of a given graph so that no connected vertices's share the same color, with the number of colors bounded by a constant. Any given graph-coloring instance can be mapped to a MAXSAT instance by having one proposition for each pair (color, vertex) and creating a formula that is satisfiable if and only if exactly one color is chosen for each vertex and the colors of vertices's corresponding to each edge are different.

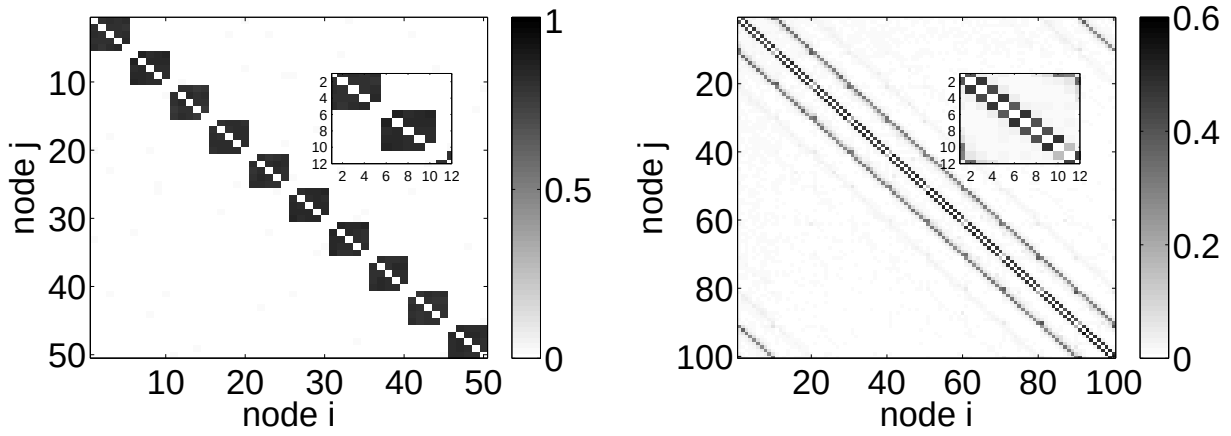
The graph-coloring problem instances were created by combining regular ring lattices and random graphs with a specified number of neighbors (Gent, Hoos, Prosser, & Walsh, 1999). Combining two graphs consists of selecting (1) all edges that overlap in the two graphs, (2) a random fraction (1-p) of the remaining edges from the first graph, and (3) a random fraction p of the remaining edges from the second graph. By combining regular graphs in this way, the amount of structure and randomness in the resulting graph can be controlled, since as p decreases the graphs become more regular (for $p = 0$, we are left with a regular ring lattice). This allows us to test methods against MAXSAT instances with varying amounts of structure. All tested instances of combined-graph coloring were downloaded from the Satisfiability Library SATLIB (Hoos & Stutzle, 2000).

As with the experiments on 2D spin glasses, DHC is applied to every solution before it is evaluated to improve its quality.

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 is 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 $P_{ij} = 0.25$.

While we do not consider time t at which each model was discovered, for some problems it may be necessary to incorporate time as well and compute a sequence of matrices P_{ij}^t where t denotes the iteration (generation) at which the probabilistic model was built. This may be beneficial if the



(a) Trap5 of size $n = 50$

(b) Ising Spin Glass of size $n = 100$

Figure 1: Probability Coincidence Matrix for 2 different problems.

models are expected to change significantly over time; otherwise, it should be advantageous to use all models available to use a more accurate estimate of the structures seen in past hBOA runs.

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.

To gain a better understanding of the information provided by PCM, we computed PCM for probabilistic models obtained by hBOA for trap-5 of $n = 50$ bits. The bisection method (Sastry, 2001b; Pelikan, 2005) was first used to compute an adequate population size to obtain the global optimum in all out of 30 independent runs. Then, the probabilistic models from all 30 successful runs were processed to compute the resulting PCM. Since trap-5 consists of independent subproblems of 5 bits each, for each trap partition, the probabilistic models should contain dependencies between all pairs of string positions in this partition. This information should be apparent in the obtained PCM matrix.

The resulting PCM matrix for trap-5 of $n = 50$ bits is visualized in figure 1a. We can clearly see that the majority of the models contain an edge between any pair of positions in the same trap partition, whereas the percentages of edges between pairs of positions that are not located in the same trap partition are extremely small. This agrees with the intuition based on the deceptiveness of trap-5 and the necessity of discovering edges between positions in the same trap partition.

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 $P_{ij} < p_{min} = 0.2$.

Revisiting the example PCM for trap-5 of 50 bits (see figure 1a), it is easy to notice that for all but extremely small values of p_{min}, the probabilistic models would be restricted to contain only the edges between pairs of positions located concurrently in any trap partition.

Another example of PCM is shown in figure 1b, which shows a PCM for 100 random instances of the 10×10 Ising spin glass. For each instance, we first used the bisection method (Sastry, 2001b; Pelikan, 2005) to ensure that hBOA finds a true ground state in 5 out of 5 independent runs. Then, models from all 500 successful runs (5 for each of the 100 instances) were processed to provide the PCM.

The resulting PCM for the 10×10 spin glass indicates that dependencies are most frequent between connected spins and the percentage of dependencies decreases with spin distance, as is expected based on previous analyses of probabilistic models for 2D spin glasses (Hauschild, Pelikan, Lima, & Sastry, 2007). Unlike in trap-5, the influence of p_{min} on the resulting model restriction is much stronger. Nonetheless, even for small p_{min}, the number of possible edges shrinks dramatically. Experimental results with various values of p_{min} are shown in section 5.

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}.

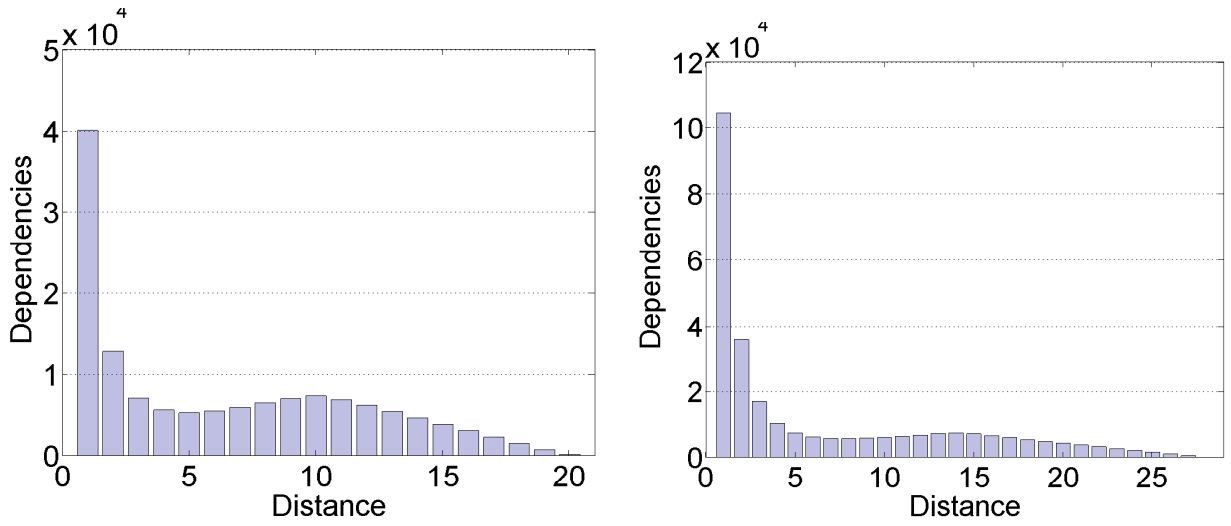
4.3 Biasing Models Using Distance

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 both trap-5 (with fixed trap partitions) and 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 (Hauschild, Pelikan, Lima, & Sastry, 2007); 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 0 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. Similarly as for the Ising spin glass, variables located in the same clause can be expected to be related much more frequently than other variables.

Given the distance metric, which defines distance between any pair of string positions, we can 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. Of course, other statistics of similar type may be computed as well. It is easy to see that the values in Q may never decrease with distance and the entry corresponding to the maximum achievable distance is equal to 1.

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.

To visualize the relationship between the distance and the number of dependencies of this



(a) $n = 400$ (b) $n = 784$

Figure 2: Total dependencies found at each distance during 5 runs of two different sized spin glass instances.

distance, figure 2 shows histograms of the number of dependencies of a given distance in 5 hBOA runs on one instance of the 20×20 spin glass and one instance of the 28×28 spin glass. As we can see, most of the dependencies found correspond to shorter distances (Hauschild, Pelikan, Lima, & Sastry, 2007). However, to use the aforementioned approach to model restriction, we must still decide on the threshold q_{min}. Typically, we would expect the maximum distance of dependencies to consider to grow with problem size; this would most likely result in a decreasing value of q_{min} with problem size.

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

Binary tournament selection was used in all experiments to select promising solutions. To build Bayesian networks, the greedy algorithm was used with the Bayesian-Dirichlet scoring metric for Bayesian networks with local structures (Cooper & Herskovits, 1992; Chickering, Heckerman, & Meek, 1997) with an additional penalty for model complexity (Friedman & Goldszmidt, 1999; Pelikan, 2005). In RTR, the window size was set as $w = \min\{n, N/20\}$ (Pelikan, 2005).

For all problem instances, bisection (Sastry, 2001b; Pelikan, 2005) 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 experiments and hBOA scalability theory (Pelikan, Sastry, & Goldberg, 2002) 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).

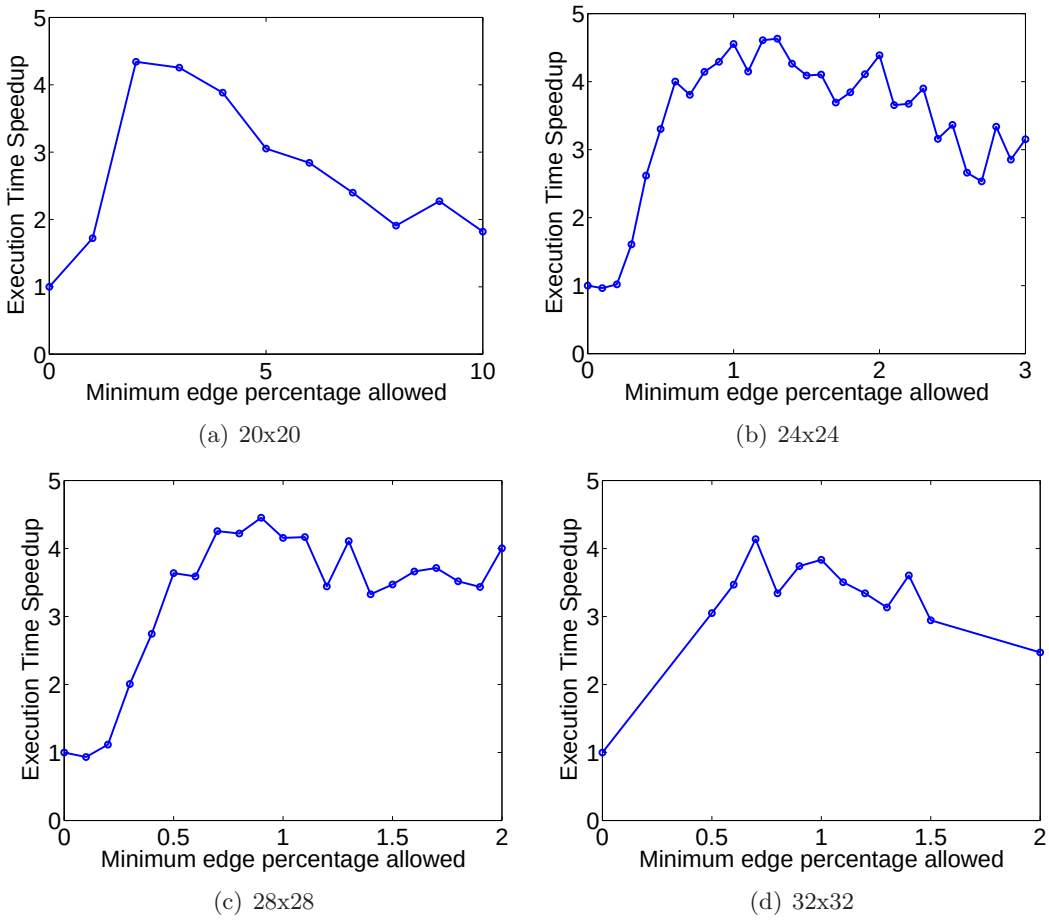
5.2 Experiments with PCM Model Bias on Ising Spin Glass

While building a PCM is relatively straightforward, determining a threshold p_{min} for cutting off edges is not. If the threshold is too small, models will not be restricted much and the benefits of using PCM to restrict model structures may be negligible. If the threshold is too large, the restrictions may be too severe and hBOA performance may suffer by using inadequate models.

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 cross-validation. For each problem size, the 100 instances were divided into 10 equally sized subsets and in each step of the cross-validation, 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 3 shows the average execution-time speedup obtained from the 10-fold cross-validation 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 (this would indicate too severe model restrictions). First of all, 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).

But exactly how are our restrictions affecting model-building efficiency? 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 of model building in hBOA. The fewer bits we must examine, the faster the model building should be. 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



(a) 20x20 (b) 24x24

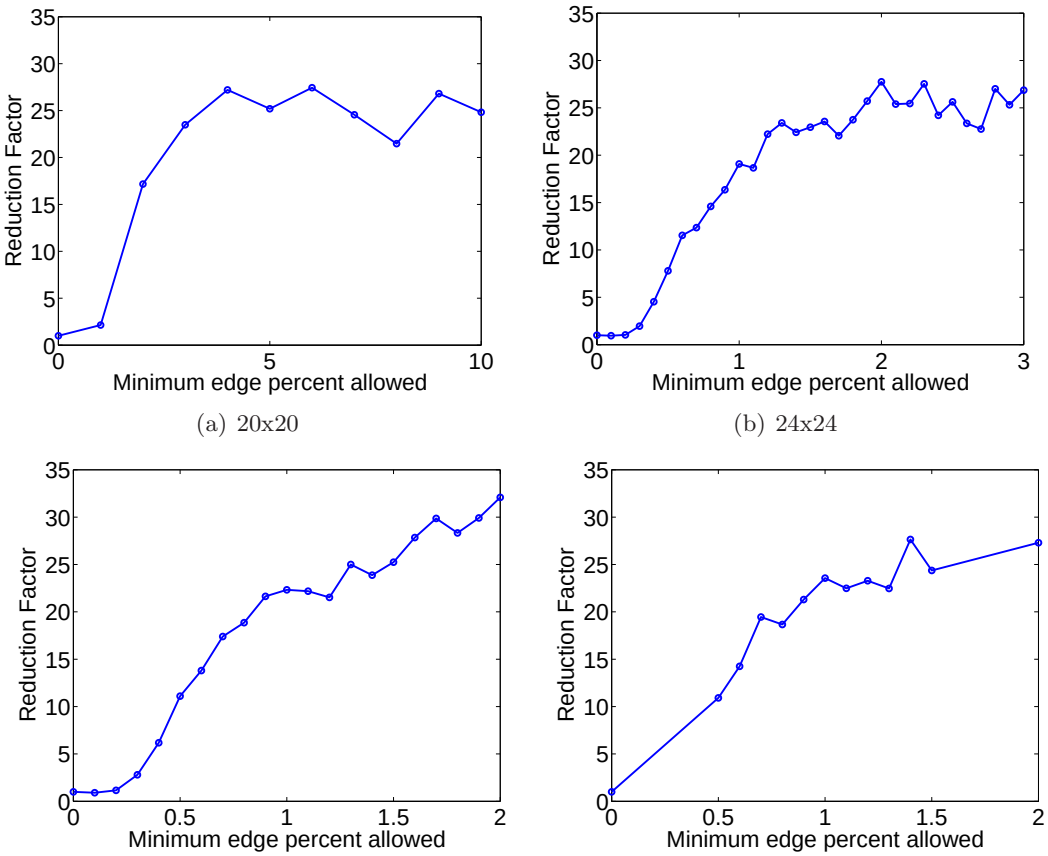
(c) 28x28 (d) 32x32

Figure 3: Execution time speedup as we increase the restrictions on model-building for 10 cross-checks among 100 different instances of various sizes

reduces by the same factor. For Bayesian local structures, limiting the model structure leads to a decrease in the number of potential splits.

Figure 4 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



(a) 20x20 (b) 24x24

(c) 28x28 (d) 32x32

Figure 4: Factor by which the number of bits in model building decreases with the model restriction based on PCM.

remarkably similar for the different problem sizes.

5.3 Experiments with Distance-Based Bias on 2D Ising Spin Glass

As was argued in section 4, restricting model structures should lead to significant speedups as was also supported by experiments obtained with the 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 get from one spin to the other one.

While it is clear that the probabilistic models discovered by hBOA contain mostly dependencies at shorter distances (see figure 2), 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 (Hauschild, Pelikan, Lima, & Sastry, 2007). On the other hand, if the distances are not restricted sufficiently, the benefits of using this approach may be negligible.

Size	Execution-time 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.

To explore this trade-off, 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 run experiments only allowing dependencies between spins of a maximum distance from 1 to 10). For the larger problems some of the instances would not converge even for extremely large population sizes ($N = 512000$) when the restrictions on model structure were too strong; the results in these cases are omitted.

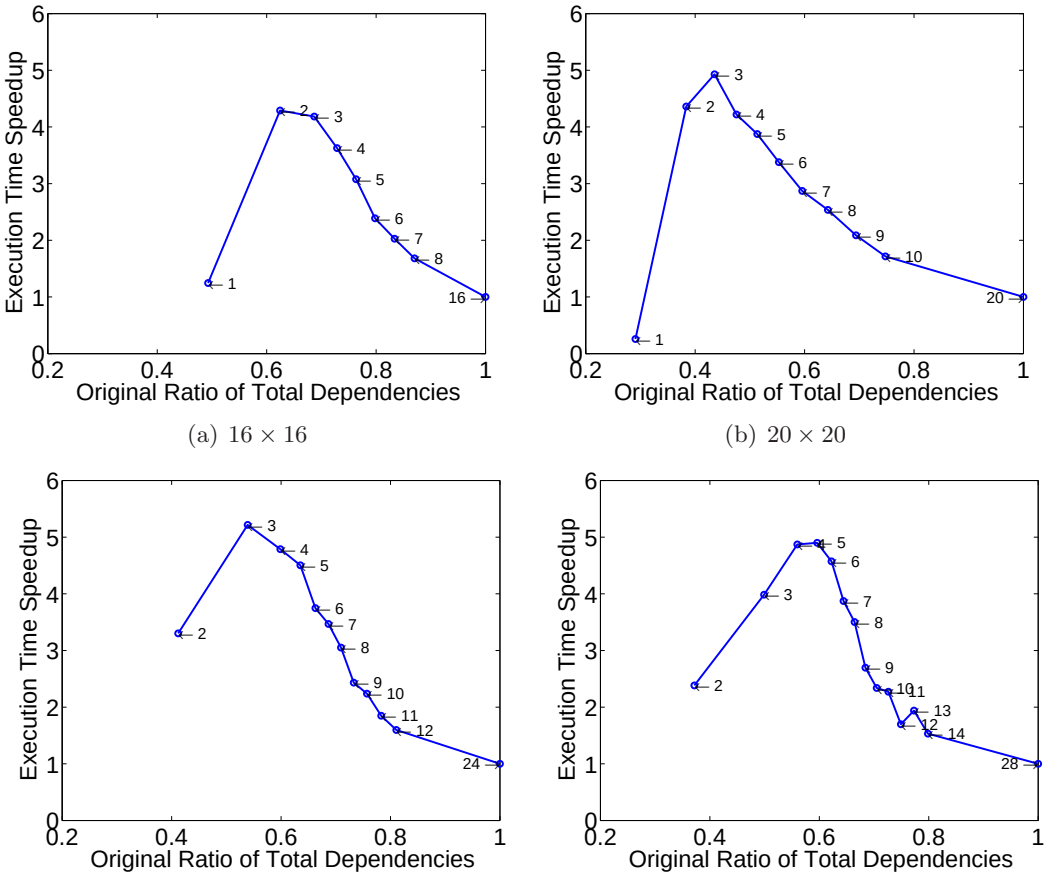
Figure 5 shows the execution-time speedup with model complexity restricted by the maximum distance. The horizontal axis is the ratio of the number of dependencies in the original runs that matched that restriction compared to the overall dependencies. The maximum distance allowed is shown as a label for each of the points in the graph. For example, in figure 5a we see that in the original runs of spin glasses of size 16x16, about 50% of the dependencies were neighbor dependencies and more than 80% were dependencies of distance 7 or less.

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 previous results (Hauschild, Pelikan, Lima, & Sastry, 2007), 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 6 shows the factor by which this quantity decreases with various thresholds on the maximum distance. Like in the previous figure, the distance restrictions are labeled on the graph with arrows. 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, 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. This is again great news, indicating that we can keep our speedups even as we scale to larger problems.

A comparison of tables 1 and 2 reveals that while the speedups in the two cases are similar,



(a) 16×16 (b) 20×20

(c) 24×24 (d) 28×28

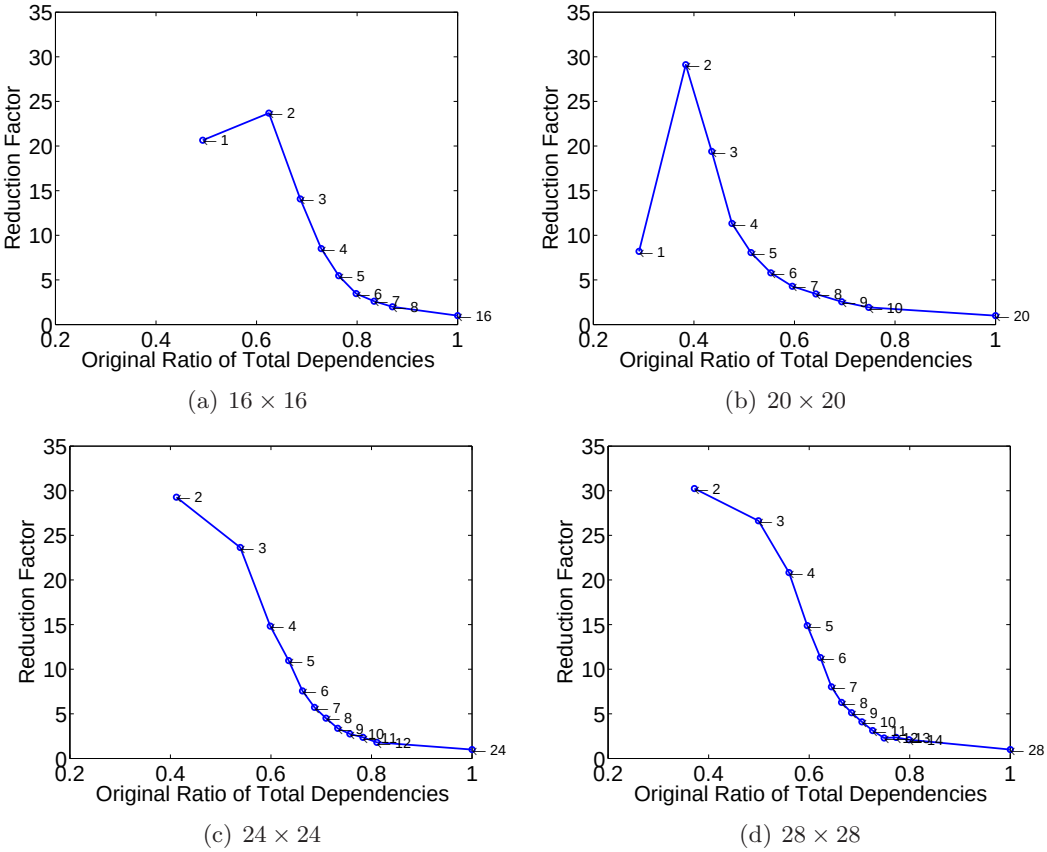
Figure 5: Execution-time speedup with model restriction based on the maximum distance on the 2D Ising spin glass.

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 Experiments with 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 maximum distance on MAXSAT we must first define a distance metric. Before defining the metric, we create a graph corresponding to the underlying MAXSAT instance by creating a special node for each proposition and connecting all propositions that appear in the same clause with an edge of length 1. Then, the distance between two propositions is defined as the shortest path between these propositions in the underlying graph. The distances can be computed using an all-pairs shortest path algorithm. For example, consider the following MAXSAT instance:



(a) 16×16 (b) 20×20

(c) 24×24 (d) 28×28

Figure 6: Factor by which the number of bits in model building decreases with the model restrictions based on maximum distance on 2D Ising spin glass.

$$(X_4 \vee X_3) \wedge (X_5 \vee \neg X_2) \wedge (\neg X_1 \vee X_3) \wedge (X_5 \vee X_1) \tag{5}$$

In the above formula, X_4 is 1 unit of distance from X_3 and 2 units from X_1 because the shortest path between X_4 and X_1 passes X_3, which is 1 unit of distance from both X_4 and X_1. Additionally, X_4 is 3 units of distance from X_5 because the shortest path between X_4 and X_5 goes through X_1, which is at distance 1 from X_5 and at distance 2 from X_4. It is possible that there is no path between two propositions and in this case the distance between them is defined as the number of propositions (this distance is unattainable for any two connected nodes).

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 trade-off, we looked at instances of MAXSAT for graph coloring of combined graphs (Gent, Hoos, Prosser, & Walsh, 1999) 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

Size	Execution-time speedup	q_{min}	% Total Dep.
256 (16×16)	4.2901	2	4.7%
400 (20×20)	4.9288	3	6.0%
576 (24×24)	5.2156	3	4.1%
784 (28×28)	4.9007	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

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 extremely large population sizes ($N = 512000$); in these cases the results were omitted (such restrictions were not used). For $p = 1$, the maximum distance of 3 was also too restrictive and these results were also omitted.

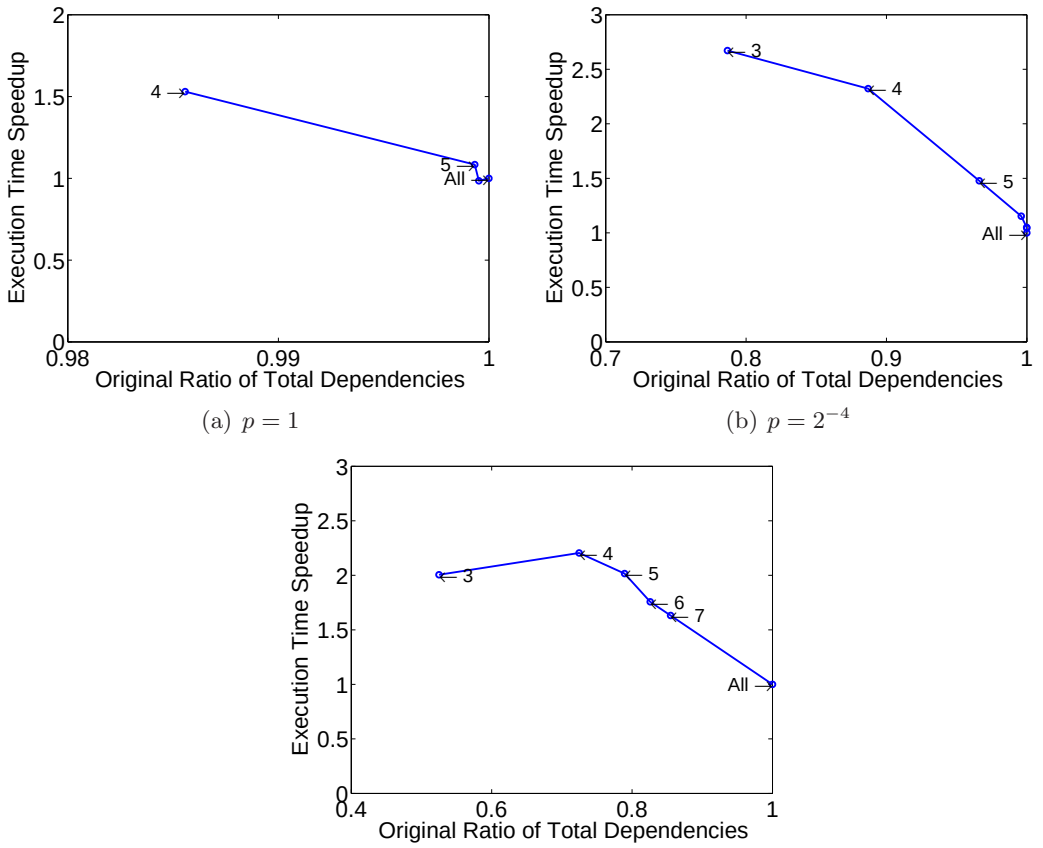
Figure 7 shows the execution-time speedup of hBOA on MAXSAT with model complexity restricted by the maximum distance. As in the results with distance restrictions on 2D spin glasses, the horizontal axis stands for the ratio of the number of dependencies with a specific distance bound and the total number of dependencies in the original, unrestricted runs. The maximum distance allowed is shown as a label for each of the 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 (represented by parameter p). For $p = 1$, problem instances have very little structure and we see that only one distance threshold led to a speedup of about 1.5; the speedups obtained in the remaining cases were negligible. 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.

As the amount of structure in the problem increases (by decreasing p), we also see 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. One thing to note about the graphs is 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 8 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.

Table 3 shows the best speedups, the corresponding maximum distance threshold and the percentage of total 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 work to 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



(a) $p = 1$

(b) $p = 2^{-4}$

(c) $p = 2^{-4}$

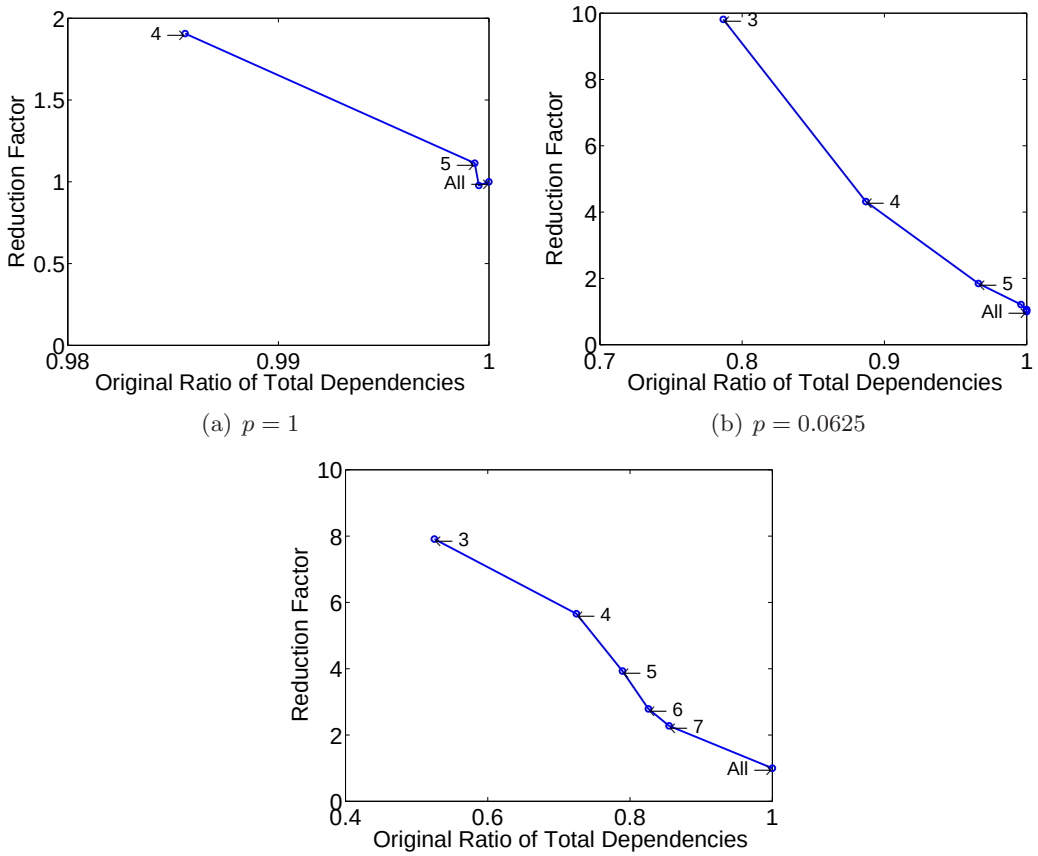
Figure 7: Execution-time speedup with model restrictions based on the maximum distance on MAXSAT for different values of p

almost 50% improvement in efficiency. In the remaining two cases, the results were very similar, with approximately 30% of the original 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



(a) $p = 1$

(b) $p = 0.0625$

(c) $p = 0.0039$

Figure 8: 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.

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 tpractitionershe 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 coloring). 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 techniqueappropriates 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 (Goldberg, 2002; Sastry, Pelikan, & Goldberg, 2006). 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 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 coloring or the minimum vertex cover. Second, the process of choosing thresholds to achieve maximum speedups should be

p	Execution-time 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

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 Materiel 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.

The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the Air Force Office of Scientific Research, or the U.S. Government. Some experiments were done using the hBOA software developed by Martin Pelikan and David E. Goldberg at the University of Illinois at Urbana-Champaign and most experiments were performed on the Beowulf cluster maintained by ITS at the University of Missouri in St. Louis.

References

Ackley, D. H. (1987). An empirical study of bit vector function optimization. *Genetic Algorithms and Simulated Annealing*, 170–204.

Albert, L. A. (2001). *Efficient genetic algorithms using discretization scheduling.* Master's thesis, University of Illinois at Urbana-Champaign, Department of General Engineering, Urbana, IL.

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.

Baluja, S. (2006). Incorporating a priori knowledge in probabilistic-model based optimization. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable optimization via probabilistic modeling: From algorithms to applications* (pp. 205–219). Springer.

Binder, K., & Young, A. (1986). Spin-glasses: Experimental facts, theoretical concepts and open questions. *Rev. Mod. Phys.*, *58*, 801.

Bosman, P. A. N., & Thierens, D. (2000). Continuous iterated density estimation evolutionary algorithms within the IDEA framework. *Workshop Proceedings of the Genetic and Evolutionary*

Computation Conference (GECCO-2000), 197–200.

Boughaci, D., & Drias, H. (2004). A performance comparison of evolutionary meta-heuristics and solving MAX-SAT problems. In Okatan, A. (Ed.), *International Conference on Computational Intelligence* (pp. 379–383). International Computational Intelligence Society.

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

Cantú-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*. Boston, MA: Kluwer.

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.

Cooper, G. F., & Herskovits, E. H. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

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

Deb, K., & Goldberg, D. E. (1991). *Analyzing deception in trap functions* (IlliGAL Report No. 91009). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Etxeberria, R., & Larrañaga, P. (1999). Global optimization using Bayesian networks. In Rodriguez, A. A. O., Ortiz, M. R. S., & Hermida, R. S. (Eds.), *Second Symposium on Artificial Intelligence (CIMAF-99)* (pp. 332–339). Habana, Cuba: Institude of Cybernetics, Mathematics, and Physics and Ministry of Science, Technology and Environment.

Fischer, K., & Hertz, J. (1991). *Spin glasses*. Cambridge: Cambridge University Press.

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

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

Goldberg, D. E. (1999). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. *Genetic and Evolutionary Computation Conference (GECCO-99)*, 212–219. Also IlliGAL Report No. 99002.

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

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, *3*(5), 493–530.

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

Harik, G. (1999). *Linkage learning via probabilistic modeling in the ECGA* (IlliGAL Report No. 99010). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

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

Hartmann, A. K. (1996). Cluster-exact approximation of spin glass ground states. *Physica A*, *224*, 480.

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

Hoos, H. H., & Stutzle, T. (2000). Satlib: An online resource for research on sat. *SAT 2000*, 283–292. SATLIB is available online at www.satlib.org.

Howard, R. A., & Matheson, J. E. (1981). Influence diagrams. In Howard, R. A., & Matheson, J. E. (Eds.), *Readings on the principles and applications of decision analysis*, Volume II (pp. 721–762). Menlo Park, CA: Strategic Decisions Group.

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

Lima, C. F., Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E., & Lobo, F. G. (2006). Substructural neighborhoods for local search in the Bayesian optimization algorithm. *Parallel Problem Solving from Nature*, 232–241.

Mendiburu, A., Miguel-Alonso, J., & Lozano, J. A. (2006). Implementation and performance evaluation of a parallelization of estimation of bayesian network algorithms. *Parallel Processing Letters*, *16*(1), 133–148.

Mezard, M., Parisi, G., & Virasoro, M. (1987). *Spin glass theory and beyond*. Singapore: World Scientific.

Mühlenbein, H., & Mahnig, T. (1999). FDA – A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, *7*(4), 353–376.

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

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature*, 178–187.

Ocenasek, J. (2002). *Parallel estimation of distribution algorithms*. Doctoral dissertation, Faculty of Information Technology, Brno University of Technology, Brno.

Ocenasek, J., Cantú-Paz, E., Pelikan, M., & Schwarz, J. (2006). Design of parallel estimation of distribution algorithms. Springer.

Ocenasek, J., & Schwarz, J. (2000). The parallel Bayesian optimization algorithm. In *Proceedings of the European Symposium on Computational Inteligence* (pp. 61–67). Physica-Verlag.

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

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. *Genetic and Evolutionary Computation Conference (GECCO-2001)*, 511–518.

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

Pelikan, M., & Goldberg, D. E. (2003b). A hierarchy machine: Learning to optimize from nature and humans. *Complexity*, *8*(5), 36–45.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. *Genetic and Evolutionary Computation Conference (GECCO-99)*, *I*, 525–532.

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.

Pelikan, M., & Hartmann, A. K. (2006). Searching for ground states of Ising spin glasses with hierarchical BOA and cluster exact approximation. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable optimization via probabilistic modeling: From algorithms to applications* (pp. 333–349). Springer.

Pelikan, M., & Sastry, K. (2004). Fitness inheritance in the Bayesian optimization algorithm. *Genetic and Evolutionary Computation Conference (GECCO-2004)*, *2*, 48–59.

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

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

Pelikan, M., Sastry, K., & Goldberg, D. E. (2006). Sporadic model building for efficiency enhancement of hierarchical BOA. *Genetic and Evolutionary Computation Conference (GECCO-2006)*, 405–412.

Rana, S., & Whitley, D. L. (1998). Genetic algorithm behavior in the MAXSAT domain. *Parallel Problem Solving from Nature*, 785–794.

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

Sastry, K. (2001b). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, Department of General Engineering, Urbana, IL.

Sastry, K., & Goldberg, D. E. (2004a, 26-30). Designing competent mutation operators via probabilistic model building of neighborhoods. *Genetic and Evolutionary Computation Conference (GECCO-2004)*, 114–125.

Sastry, K., & Goldberg, D. E. (2004b). Let's get ready to rumble: Crossover versus mutation head to head.

Sastry, K., Goldberg, D. E., & Llorà, X. (2007). Towards billion bit optimization via parallel estimation of distribution algorithm. *Genetic and Evolutionary Computation Conference (GECCO-2007)*, 577–584.

Sastry, K., Goldberg, D. E., & Pelikan, M. (2001a). Don't evaluate, inherit. *Genetic and Evolutionary Computation Conference (GECCO-2001)*, 551–558.

Sastry, K., Goldberg, D. E., & Pelikan, M. (2001b). Don't evaluate, inherit. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., & Burke, E. (Eds.), *Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 551–558). San Francisco, CA: Morgan Kaufmann.

Sastry, K., Pelikan, M., & Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 720–727.

Sastry, K., Pelikan, M., & Goldberg, D. E. (2006). Efficiency enhancement of estimation of distribution algorithms. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications* (pp. 161–185). Springer.

Schwarz, J., & Ocenasek, J. (2000). A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning. Personal communication.

Smith, R. E., Dike, B. A., & Stegmann, S. A. (1995). Fitness inheritance in genetic algorithms. *Proceedings of the ACM Symposium on Applied Computing*, 345–350.

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

Srivastava, R., & Goldberg, D. E. (2001). Verification of the theory of genetic and evolutionary continuation. *Genetic and Evolutionary Computation Conference (GECCO-2001)*, 551–558. Also IlliGAL Report No. 2001007.

Young, A. (Ed.) (1998). *Spin glasses and random fields*. Singapore: World Scientific.