Missouri Estimation of Distribution Algorithms Laboratory

Intelligent Bias of Network Structures in the Hierarchical BOA

Mark Hauschild and Martin Pelikan

MEDAL Report No. 2009005

January 2009

Abstract

One of the primary advantages of estimation of distribution algorithms (EDAs) over many other stochastic optimization techniques is that they supply us with a roadmap of how they solve a problem. This roadmap consists of a sequence of probabilistic models of candidate solutions of increasing quality. The first model in this sequence would typically encode the uniform distribution over all admissible solutions whereas the last model would encode a distribution that generates at least one global optimum with high probability. It has been argued that exploiting this knowledge should improve EDA performance when solving similar problems. This paper presents an approach to bias the building of Bayesian network models in the hierarchical Bayesian optimization algorithm (hBOA) using information gathered from models generated during previous hBOA runs on similar problems. The approach is evaluated on trap-5 and 2D spin glass problems.

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/

Intelligent Bias of Network Structures 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

Abstract

One of the primary advantages of estimation of distribution algorithms (EDAs) over many other stochastic optimization techniques is that they supply us with a roadmap of how they solve a problem. This roadmap consists of a sequence of probabilistic models of candidate solutions of increasing quality. The first model in this sequence would typically encode the uniform distribution over all admissible solutions whereas the last model would encode a distribution that generates at least one global optimum with high probability. It has been argued that exploiting this knowledge should improve EDA performance when solving similar problems. This paper presents an approach to bias the building of Bayesian network models in the hierarchical Bayesian optimization algorithm (hBOA) using information gathered from models generated during previous hBOA runs on similar problems. The approach is evaluated on trap-5 and 2D spin glass problems.

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

1 Introduction

The hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan, 2005; Pelikan & Goldberg, 2006) is an advanced estimation of distribution algorithm (EDA) (Baluja, 1994; Larrañaga & Lozano, 2002; Pelikan, Goldberg, & Lobo, 2002; Pelikan, Sastry, & Cantú-Paz, 2006), capable of solving a broad class of nearly decomposable and hierarchical problems scalably and reliably. Like all EDAs, hBOA operates by replacing standard variation operators of genetic algorithms, such as crossover and mutation, with building a probabilistic model of promising solutions and sampling new candidate solutions from the built model. Though EDAs have many advantages over standard genetic algorithms (Larrañaga & Lozano, 2002; Pelikan, Sastry, & Cantú-Paz, 2006), one of their biggest advantages is that after each run of an EDA, we are left with a series of probabilistic models that reveal a significant amount of information about the problem. These

models can in many ways be considered a roadmap to the final population of solutions, in that they represent exactly how the EDA focused its exploration on specific regions of the search space. While it has been argued that it should be possible to exploit this information to improve EDA performance (Schwarz & Ocenasek, 2000; Baluja, 2006), this has been rather difficult in practice. Most attempts to use this information so far have relied on hand-inspection of previous models and required parameters to be tuned to ensure efficient performance (Hauschild, Pelikan, Sastry, & Goldberg, 2008; Baluja, 2006; Schwarz & Ocenasek, 2000).

This paper proposes an automated technique for exploiting information gathered in previous runs to speed up subsequent runs of an EDA on similar problems. This is accomplished by building a split probability matrix (SPM) based on models acquired in previous runs on similar problems and using the built SPM to bias the structural priors used in model building. The proposed method requires only one parameter and can be used without any hand-inspection of previous models. Besides providing an approach to bias probabilistic models based on previous runs, the proposed technique can also be used to replace more conventional penalty terms for controlling model complexity in multivariate EDAs. While in this paper we apply the proposed technique only to hBOA, it should be straightforward to adapt this method to other multivariate EDAs.

The paper is organized as follows. Section 2 outlines hBOA. Section 3 discusses learning from experience in EDAs, with a particular emphasis on previous techniques used to bias model building. Section 4 starts by explaining structural priors in Bayesian metrics and then outlines the proposed approach to biasing model building in hBOA using the split probability matrix (SPM). Section 5 presents the test problems and experimental results. Finally, section 6 summarizes and concludes the paper.

2 Hierarchical BOA (hBOA)

The hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan, 2005) 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. In this paper we use truncation selection with threshold $\tau = 50\%$. After selecting the promising solutions, hBOA builds a Bayesian network (Howard & Matheson, 1981; Pearl, 1988) with local structures in the form of decision trees (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999) as a model of these solutions. New solutions are generated by sampling the built network. These are then incorporated into the original population using restricted tournament replacement (RTR) (Harik, 1995), 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. (Pelikan, 2005). The next iteration is then executed unless some predefined termination criteria are met.

A Bayesian network (BN) (Pearl, 1988; Howard & Matheson, 1981) 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 $X_1, X_2, \ldots, X_n$:

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

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), and $p(X_i|\Pi_i)$ is the conditional probability of X_i given Π_i.

In BNs with decision trees, the conditional probabilities $p(X_i|\Pi_i)$ are encoded by a decision tree T_i; for n variables, there are n decision trees. Each internal node of the decision tree T_i is labeled by a variable $X_j \in \Pi_i$ where $j \neq i$. Children of a node labeled by X_j correspond to disjoint subsets of the potential values of X_j; for each value of X_j, there is one child corresponding to this value. Each traversal of a decision tree T_i for X_i thus corresponds to a constraint on the values of other variables from Π_i. Each leaf of T_i then stores the probabilities of X_i given the constraint defined by the traversal of T_i ending in this leaf.

To learn the structure of a Bayesian network with decision trees, a simple greedy algorithm is used (Heckerman, Geiger, & Chickering, 1994; Chickering, Heckerman, & Meek, 1997). The greedy algorithm starts with an empty network represented by single-node decision trees. Each iteration splits one leaf of any decision tree that improves the score of the network most until no more improvement is possible. For more details on learning BNs with decision trees, see (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999; Pelikan, 2005).

Learning the structure (decision trees) is the most challenging task in model building (Pelikan, 2005). That is why when enhancing efficiency of model building in hBOA, the primary focus is typically on learning the structure.

3 Learning from Experience

When solving a problem with genetic and evolutionary algorithms (GEAs) or estimation of distribution algorithms (EDAs), the only information we usually require is the representation of candidate solutions and the fitness function. However, when problem-specific information is available, it is often possible to use this information to improve algorithm performance. In general, there are two approaches to speeding up EDAs by incorporating problem-specific knowledge: (1) bias the procedure for generating the initial population (Schwarz & Ocenasek, 2000; Sastry, 2001a; Pelikan & Goldberg, 2003) 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). In this section we will discuss approaches most closely related to that proposed in this paper, that is, approaches that bias model building to prefer specific network structures.

The first attempt to bias model building in EDAs using problem-specific knowledge was made by Schwarz & Ocenasek (Schwarz & Ocenasek, 2000). In their study, BOA was applied to the graph bipartitioning problem, 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 should be more likely connected in the probabilistic model, the underlying graph was used as a prior network and models were biased to give preference to edges that can also be found in the underlying graph. Other dependencies were allowed but were penalized.

Another study that looked at solving the graph bipartitioning problem by using prior knowledge was done by Mühlenbein & Mahnig (Mühlenbein & Mahnig, 2002). However, unlike in ref. (Schwarz & Ocenasek, 2000), which provided a soft bias towards models that corresponded to the underlying graph, here the authors restricted the models to only allow edges between nodes connected in the

underlying graph. This led to both a reduction in model complexity as well as a speedup of the model building.

Baluja (Baluja, 2006) discussed how to bias EDA model building on the graph coloring problem, in which the task is to assign a color to each node out of the set of available colors such that no connected nodes have the same color. Much like the study of Mühlenbein & Mahnig (Mühlenbein & Mahnig, 2002), the probabilistic models were restricted to only allow edges that were consistent with the underlying graph structure. In this study though, dependency-tree models were used, whereas in the study of Mühlenbein & Mahnig (Mühlenbein & Mahnig, 2002), Bayesian networks were used. More recently, Santana (Santana, Larrañaga, & Lozano, 2007) used hard restrictions to help speed up model-building of a dependency-tree model when working on the protein design problem.

More recently, Hauschild et al. (Hauschild, Pelikan, Sastry, & Goldberg, 2008) examined two different approaches to biasing model building of Bayesian networks based on models gathered in previous runs; as test problems, the 2D Ising spin glass and MAXSAT were used in this study. In the first approach, statistics were first gathered on the probability of an edge between any pair of nodes in a number of trial runs. Then, for problems with similar structure, these statistics were used to strictly disallow edges that were below a cutoff threshold supplied by the user. In the second approach, statistics were gathered on the likelihood of edges spanning certain distances given a problem-specific distance metric on the variables. A user-supplied cutoff threshold was then used to disallow least likely edges. Both these approaches resulted in a substantial speedup of model building if adequate thresholds were used.

Lastly, while not directly related to efficiency enhancement, the work of Claudio et al. (Lima, Lobo, & Pelikan, 2008) is closely related to the proposed method of modifying the complexity penalty of hBOA. In previous work they had observed that truncation selection led to improved model quality when compared to using tournament selection. They showed that by taking into account the non-uniform distribution of the population under tournament selection, it was possible to modify the complexity penalty based on tournament size so that model quality was greatly improved.

Most of these results showed that using prior knowledge to bias model building in EDAs can lead to significant speedups, allowing EDAs to solve more complex and larger problems. However, there are several problems with the currently available methods. First of all, it is often very difficult to process prior information about the problem structure to define adequate restrictions on model structure. Furthermore, 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. Finally, in several of the aforementioned methods, a parameter is used to specify the level of bias and setting this parameter adequately is not always an easy task; with inadequate settings, a slowdown or a complete inability to solve the problem may result. While the approach proposed in this paper can also be tuned via a user-specified parameter, the default value of this parameter appears to work quite well and substantial speedups are obtained for a broad range of values of this parameter.

4 Biasing the Model Building

Before describing the proposed approach to biasing model building in hBOA, it is important to understand main goals of introducing such a bias. First of all, by discouraging dependencies that are unlikely to improve model quality based on information obtained from models on similar problems, the search for model structure becomes more efficient due to the reduction of the space of likely

structures; additionally, the models may become more accurate because of adding fewer spurious dependencies. On the other hand, by encouraging dependencies that seem important based on promising regions explored during prior runs on similar problems, the bias makes it more likely that important dependencies will be covered even with smaller samples (population sizes).

The remainder of this section describes details of the structural bias proposed in this paper.

4.1 Structural Priors in Bayesian Metrics

As the main mechanism for introducing structural bias, we use the structural priors included in Bayesian-Dirichlet metric and other Bayesian metrics. The basic form of Bayesian-Dirichlet metric for network B and data set D is

$$p(B|D,\xi) = \frac{p(B|\xi)p(D|B,\xi)}{p(D|\xi)}. \tag{2}$$

where ξ denotes the background knowledge. In the above equation, the prior probability of a network structure is represented by the term $p(B|\xi)$. The prior probabilities are often set to penalize complex networks. In hBOA, the bias toward simple models is typically introduced as (Friedman & Goldszmidt, 1999; Pelikan, 2005)

$$p(B|\xi) = c2^{-0.5(\sum_i |L_i|)log_2 N}, \tag{3}$$

where N is the size of the population, $\sum_i |L_i|$ is the total number of leaves in all decision trees representing parameters of the local conditional distributions, and c is a normalization constant so that the prior probabilities of all network structures sum to 1.

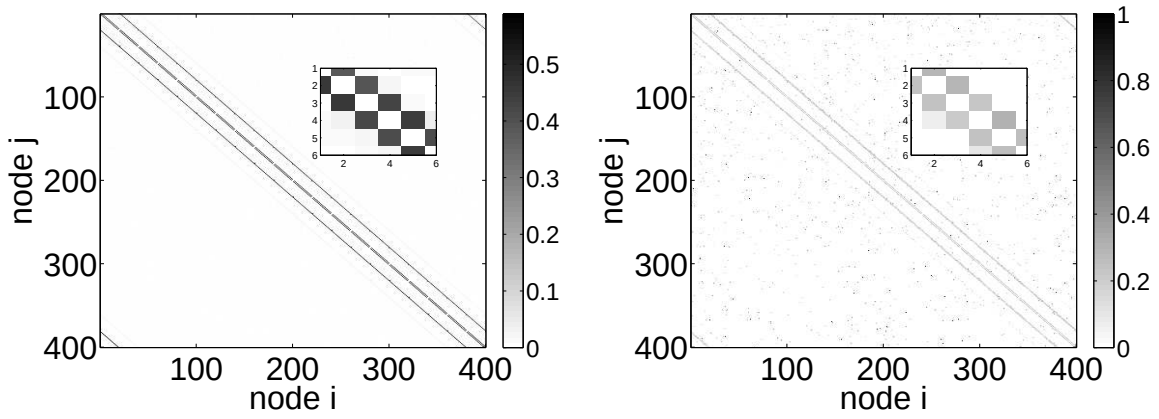
The bias toward probabilistic models resembling those discovered in previous runs on similar problems will be incorporated by modifying prior probabilities of network structures as will be described shortly.

4.2 Split Probability Matrix

The key to the proposed method to biasing probabilistic models in hBOA is the *split probability matrix* (SPM), which can be built from models gathered from previous runs of hBOA on similar problems. The SPM stores observed probabilities of certain properties of network structures, which can consequently be used to bias new models.

More formally, the SPM is a four-dimensional matrix of size $n \times n \times d \times e$ where n is the number of variables (problem size), d is the maximum number of splits in any decision graph for which we gather the data for and e is the number of generations from which we collect data. Let us denote the matrix by S and the elements of S by $S_{i,j,k,g}$ where $i, j \in \{1 \dots n\}$, $k \in \{1 \dots d\}$ and $g \in \{1 \dots e\}$. The value $S_{i,j,k,g}$ is defined as, during the generation g of the sample data, the conditional probability of a kth split on the variable X_j in the decision graph for X_i given that there were $k-1$ such splits performed already; if there are no models with at least $k-1$ such splits, we define $S_{i,j,k,g} = 0$.

To ensure enough data for the SPM-based bias in each generation, before computing S, we first examine all models collected and set an upper threshold e for the number of generations to store in SPM based on the generation reached by at least 90% of the successful hBOA runs on similar problems. Of course, the 90% threshold can be modified, but it should remain a rather large value so that the sample for creating SPM is large enough for every generation under consideration. Then, we parse all these probabilistic models, incrementing the element $S_{i,j,k,g}$ if there were at least k



(a) No splits (b) One split

Figure 1: First two levels of the first generation SPM generated from 90 instances of 20×20 2D Ising spin glasses.

splits on X_j in the decision tree for X_i during generation g. Once this is complete, for all $k \neq 1$, we divide all elements $S_{i,j,k}$ by $S_{i,j,k-1}$. Lastly, we divide all elements S_{ij1} by the total number of probabilistic models that were parsed. If the number of generations in a specific run is greater than the upper bound e, the priors based on the generation e are used for the rest of the run.

For example, suppose that during generation g for a particular $i \neq j$, 40 of the 100 available models had at least one split on variable X_j in the decision tree for X_i, and 10 of those models had at least two such splits. Then $S_{i,j,1,g} = 40/100 = 0.4$ and $S_{i,j,2,g} = 10/40 = 0.25$.

Figure 1 shows an example SPM gathered from the first generation of 90 instances of 20×20 2D Ising spin glasses. Figure 1a represents the probability that there was at least one split between two variables in a decision tree during the first generation. Figure 1b shows the conditional probability of the second split between two variables in a decision tree given that there is already a split between them in the first generation.

4.3 SPM-Based Model-Building Bias

The prior probability of any network structure is set to the product of prior probabilities for all decision trees:

$$p(B|\xi) = \prod_{i=1}^{n} p(T_i). \tag{4}$$

For decision tree T_i, $p(T_i)$ is set to the following product:

$$p(T_i) = \prod_{j \neq i} q^{\kappa}_{i,j,k(i,j)}, \tag{5}$$

where $q_{i,j,k(i,j)}$ denotes the probability that there are at least $k(i,j)$ splits on X_j in decision trees for X_i in the set of models used to bias model building, $k(i,j)$ denotes the number of splits on X_j in T_i, and κ is used to tune the effects of prior information on learning future models. For $\kappa = 0$, prior models have no effect on learning future models; the larger the value of κ, the stronger the effect of prior models are on learning of future models. Based on experimental results, we suggest to use $\kappa = 1$, although the optimal value of κ clearly depends on the problem under consideration.

Since the models are learned with the greedy algorithm described earlier in this paper, which performs one split at a time, the conditional probabilities stored in SPM can be directly used to compute the gains in the log-likelihood of the model after any admissible split. More specifically, let's consider evaluation of the split on X_j in the tree T_i during generation g, let's denote the network before performing this split by B and the network after performing this split by B', and let's assume that there were $k(i,j)-1$ splits on X_j in T_i before this split. Then, the gain in the log-likelihood corresponding to this split is defined as

$$\delta_{i,j} = \log_2 p(D|B',\xi) - \log_2 p(D|B,\xi) + \kappa \log_2 S_{i,j,k(i,j),g}.$$

For comparison, the gain without considering prior information about model structure and only considering the complexity penalty is defined as

$$\delta_{i,j} = \log_2 p(D|B',\xi) - \log_2 p(D|B,\xi) - 0.5 log_2 N.$$

Note that SPM is not defined for $g > e$; in this case, the entries for generation e are used instead.

4.4 Test Problems

4.4.1 Trap-5: Concatenated 5-bit trap

In trap-5 (Ackley, 1987; Deb & Goldberg, 1994), the input string is first partitioned into independent groups of 5 bits each. This partitioning is unknown to the algorithm. The contribution of each group of 5 bits is computed as

$$trap_5(u) = \begin{cases} 5 & \text{if } u = 5 \\ 4-u & \text{otherwise} \end{cases}, \quad (6)$$

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. Since $trap_5$ is fully deceptive, traps of order 5 necessitate that all bits in each group are treated together, because statistics of lower order are misleading (Thierens, 1999; Deb & Goldberg, 1994). 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.

4.4.2 2D Ising Spin Glass

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

$$E(C) = -\sum_{\langle i,j \rangle} s_i J_{i,j} s_j \ , \quad (7)$$

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

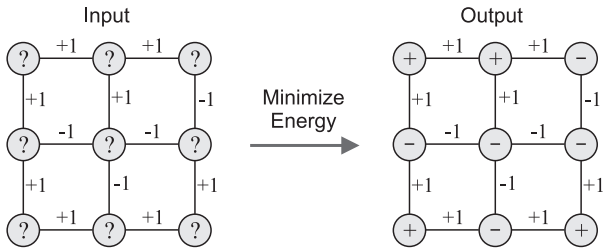


Figure 2: Finding ground states of Ising spin glasses in two dimensions. For simplicity, we omit periodic boundary conditions in this figure.

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). See figure 2 for an illustration.

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 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 Univ. of Wisconsin who identified the ground states using flat-histogram Markov chain Monte Carlo (Dayal et al., 2004). The ground states of the remaining instances were obtained from the Spin Glass Ground State Server at the Univ. of Cologne (Spin Glass Ground State Server, 2004).

To improve the performance of hBOA on the 2D spin glass, we incorporate a deterministic hill climber (DHC) based on single-bit flips (Pelikan, 2005) to improve the quality of each evaluated solution. The local searcher is applied to every solution before it is evaluated.

5 Experiments

This section covers our experiments using the SPM to bias hBOA model building on trap-5 and 2D Ising spin glasses. First the parameter settings and the experimental setup are discussed. The results are then presented.

5.1 Parameter Settings

For trap-5, bisection (Sastry, 2001b; Pelikan, 2005) was used to determine the minimum population size to ensure convergence to the global optimum in 10 out of 10 independent runs. For more reliable results, 10 independent bisection trials were run for each problem size, for a total of 100 runs for each problem size.

For each spin glass instance, bisection 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 all problem instances.

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. The results were obtained by averaging over all 100 runs. 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). In those problems using SPM bias, the maximum generational SPM data stored is set to the number of generations reached by at least 90% of the sample runs.

5.2 SPM Model Bias on Trap-5

For trap-5, 10 bisection runs of 10 runs each were first performed with standard model building bias toward simple networks used in the original hBOA. The models discovered in these 100 runs were then used to compute the SPM matrices, which were consequently used to bias model building in the next 10 rounds of bisection.

Figure 3a shows the average execution-time speedup obtained with the SPM bias over all runs using $\kappa = 1$. Note that the execution time is the time required for the entire execution of hBOA and not just that for model building. The maximum speedup achieved was by a factor of about 6; that means, hBOA with the SPM bias was able to solve the problem about 6 times faster than the original hBOA. However, as the problem size increased, speedups by factors of about 5 were more common, with the minimum speedup of 3.5. Figure 3b shows the speedup in terms of the number of evaluations. In most cases, speedups of almost 3 were achieved and, even in the worst case, the SPM bias halved the number of evaluations.

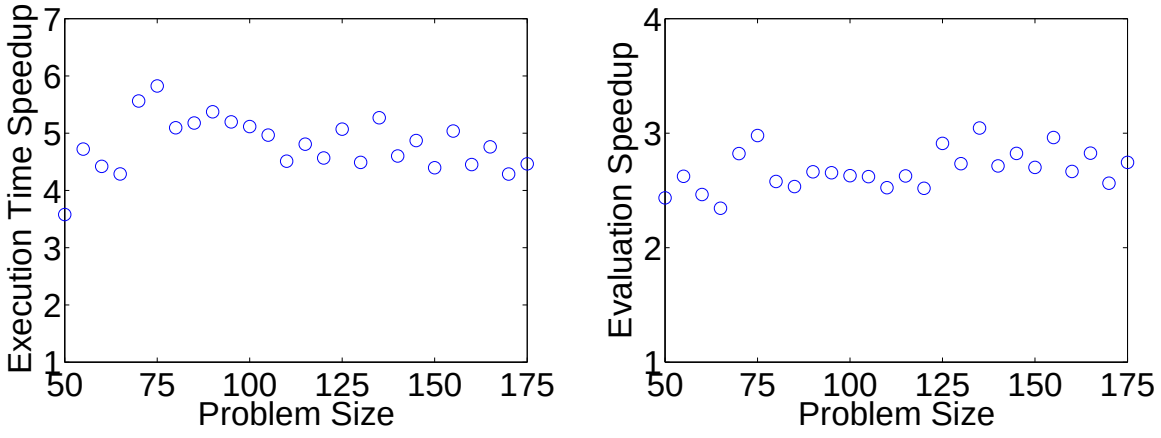
While it is clear that the overall execution time was improved using the SPM bias, it is also interesting to analyze the effects of the SPM bias on model building itself. To quantify the effects of the bias on model building complexity, we recorded the number of bits that must be checked to update model parameters during the entire model building procedure, as this is the primary factor affecting time complexity of model building in hBOA. Figure 3c shows the average factor of decrease in bits examined during model building with respect to problem size. We see that for the smallest problem size, $n = 50$, the number of bits examined during model building was reduced by a factor of about 14. The reduction factor continues to increase as the problem size increases; for the largest problem size considered, that is, $n = 175$, the reduction factor is as high as 67. This constantly increasing performance is in marked contrast to the speedup in evaluations. We see for trap-5 that the main contributor to the speedup in overall execution time is due to the reduction in work during model building.

Another interesting topic to discuss considers the effects of κ on the performance gains obtained with the SPM bias. If the effects of the SPM bias depend on the value of κ, κ provides us with a mechanism to tune the SPM bias. Figure 4 shows the effects of κ on hBOA performance in terms of the overall execution time in seconds, the number of evaluations and the number of bits examined on trap-5 of size $n = 100$. Figure 4a shows that in general, the execution time decreases as κ is decreased; however, the performance is only marginally improved when $\kappa < 1$ compared to $\kappa = 1$. The effects of κ in terms of the number of evaluations and the number of bits examined in model building are of similar nature as those measured in terms of the overall execution time (see Figure 4b and Figure 4c).

Note that the results for $\kappa = 0$ are omitted as they were very poor, corresponding to the case where no complexity penalty is used.

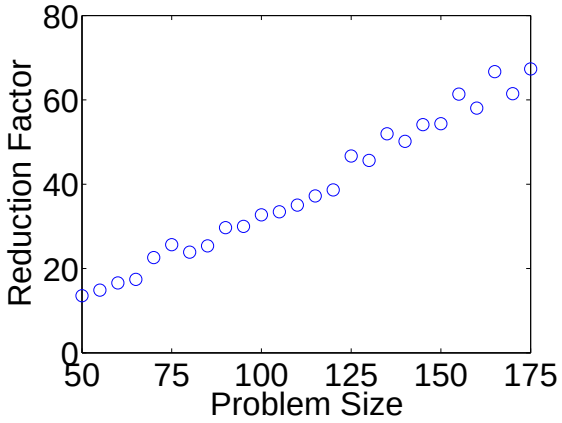
5.3 SPM Model Bias on 2D Ising Spin Glass

Thus far we saw that the SPM bias can substantially speed up hBOA on trap-5 by biasing the models towards promising structures. In this section we will examine the effects of the SPM bias on hBOA when solving 2D Ising spin glasses. Note that for trap-5 it is relatively easy to define a



(a) Execution Speedup

(b) Evaluation Speedup



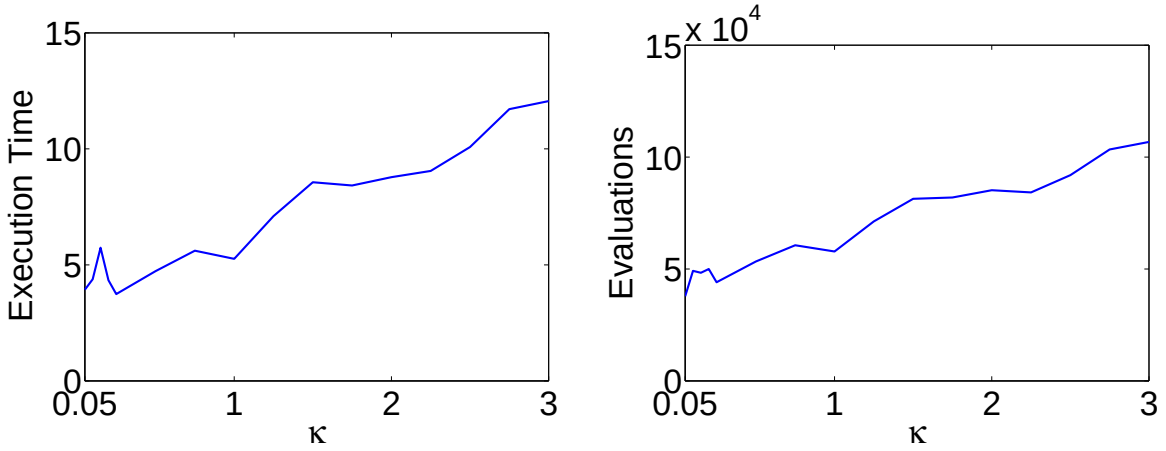
(c) Reduction in Bits Examined

Figure 3: Speedups obtained using SPM bias ($\kappa = 1$) on trap-5 in terms of the overall execution time, the number of evaluations, and the number of bits examined in model building.

perfect model; nonetheless, this is not the case with 2D Ising spin glasses, where even defining an adequate model can be extremely difficult (Mühlenbein, Mahnig, & Rodriguez, 1998; Hauschild, Pelikan, Lima, & Sastry, 2007).

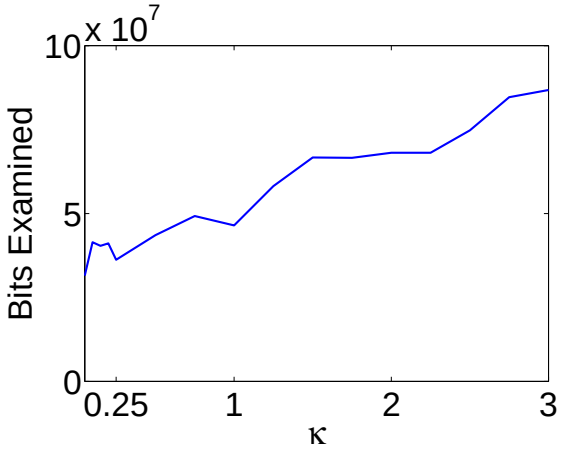
To examine the effects of the SPM bias on 2D Ising spin glasses, we considered three different problem sizes: 16×16 (256 spins), 20×20 (400 spins) and 24×24 (576 spins). For each problem size, we examined 100 random instances. In order to not use the same problem instances to both learn SPM and examine the resulting bias on hBOA performance, 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 the SPM 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 the SPM and each subset of instances was used for validation.

Table 1 shows the effects of the SPM bias on hBOA performance for $\kappa = 1$. For the smallest problem of size 16×16, the speedup of about 1.16 was obtained, but as the problem size increased, the speedup increased as well, reaching the value of about 1.56 for the problem of size 24×24.



(a) Execution Time

(b) Evaluations



(c) Bits Examined

Figure 4: The effects of κ on the execution time, the number of evaluations, and the number of bits examined in model building using SPM bias on trap-5 of size $n = 100$.

The number of bits examined in model building also decreases, and as the problem size grows, the benefits in terms of the number of bits examined grow as well. However, unlike for trap-5, we see a small increase in the number of evaluations, although the increase in the number of evaluations drops as the problem size increases. Therefore, while for trap-5 the decrease in the overall execution time was a result of the reduction in both the model building complexity as well as the number of evaluations, in 2D Ising spin glasses the speedups obtained using the SPM bias seem to almost solely be a result of reducing the time spent in model building.

Just like for trap-5, we also examined the effects of κ on the benefits obtained with the SPM bias. Figure 5 shows the execution times in seconds for hBOA with the SPM bias for a range of values of κ on the 2D Ising spin glass. Unlike for trap-5, where decreasing κ below 1 led to better performance, for 2D Ising spin glasses, $\kappa < 1$ yields dramatically worse performance than $\kappa = 1$. As κ starts to increase past 1, the execution times start to increase slowly. Table 2 shows the value of κ that led to the maximum speedup in execution time, as well as the accompanying speedups in terms of the number of evaluations and bits examined in model building.

size	Exec. speedup	Eval. Speedup	Bits Exam.
16×16	1.16	0.87	1.5
20×20	1.42	0.96	1.84
24×24	1.56	0.98	2.03

Table 1: Speedups obtained using SPM bias ($\kappa = 1$) on the 2D spin glass in terms of the overall execution time, the number of evaluations, and the number of bits examined in model building.

size	κ	Exec. speedup	Eval. Speedup	Bits Exam.
16×16	0.75	1.24	0.96	1.66
20×20	1.25	1.44	0.94	1.85
24×24	1	1.56	0.98	2.03

Table 2: The value of κ that led to the maximum execution-time speedup of hBOA using SPM bias on the 2D Ising spin glass of 3 sizes, and the accompanying speedups in the number of evaluations and the number of bits examined in model building.

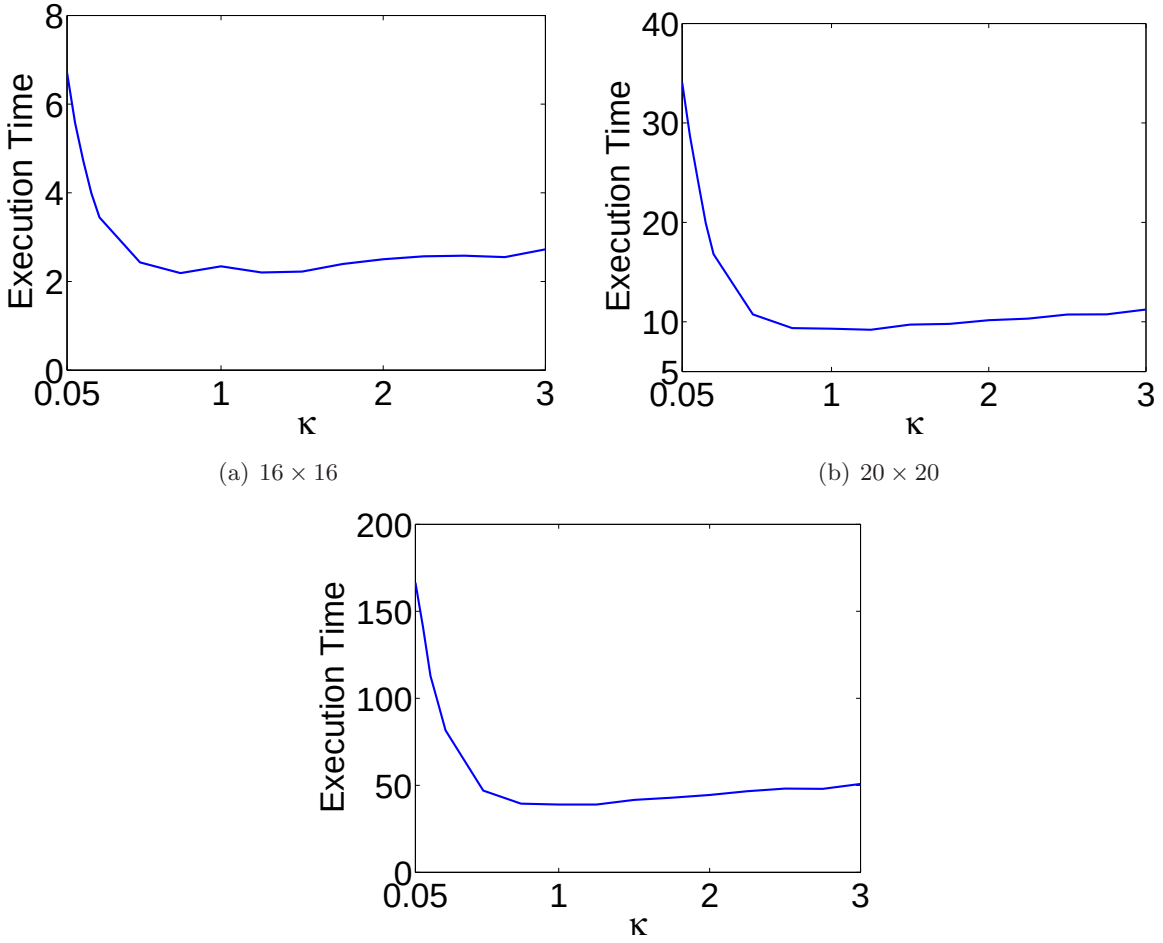
The effects of κ on the number of evaluations for the 2D spin glass are visualized in Figure 5. The results show that for all problem sizes the least evaluations are performed when $\kappa \approx 0.5$. Similarly as for the execution times, performance in terms of the number of evaluations gets dramatically worse when κ approaches 0, and it gets gradually worse as κ grows beyond its optimal value. The results for the number of evaluations are closely mirrored by the results for the number of bits examined showed in Figure 7. With respect to the number of bits examined, $\kappa \in [0.5, 1]$ yields best performance for all problem sizes.

6 Summary and Conclusions

EDAs do more than just solve hard optimization problems; they also provide a roadmap to the solution by leaving us with a series of probabilistic models that provide an overview of the fitness landscape and its properties. Since in optimization we often want to solve many problem instances of similar type, using this source of problem-specific knowledge should allow practitioners to improve EDA performance on these types of problems. This paper proposed an approach to biasing model building in EDAs based on probabilistic models from previous EDA runs on problems of similar type. The bias was introduced through structural priors of Bayesian metrics using the split probability matrix (SPM). The main focus was on the hierarchical BOA (hBOA), although the proposed approach can be adapted to other multivariate EDAs in a straightforward manner.

The proposed technique was tested on two important test problems: trap-5 and 2D spin glass. On trap-5, the SPM bias led to substantial speedups whether we measure time complexity of hBOA in terms of the number of evaluations, the number of bits examined in model building, or the overall execution time. With respect to the overall execution time, speedups of about 3.5 to 6 were obtained. While on 2D Ising spin glasses the SPM bias led to a slight increase in the number of evaluations, in terms of the number of bits examined in model building as well as the overall execution time substantial speedups were obtained. For example, the execution times reduced by a factor of 1.5 or more, with the speedups increasing with problem size.

While the proposed method does require one parameter, κ, the results show that $\kappa = 1$ leads to execution-time speedups in all tested problems, and that these speedups are close to those obtained with the optimal value of κ. This represents an important advantage over the previously proposed



(a) 16×16

(b) 20×20

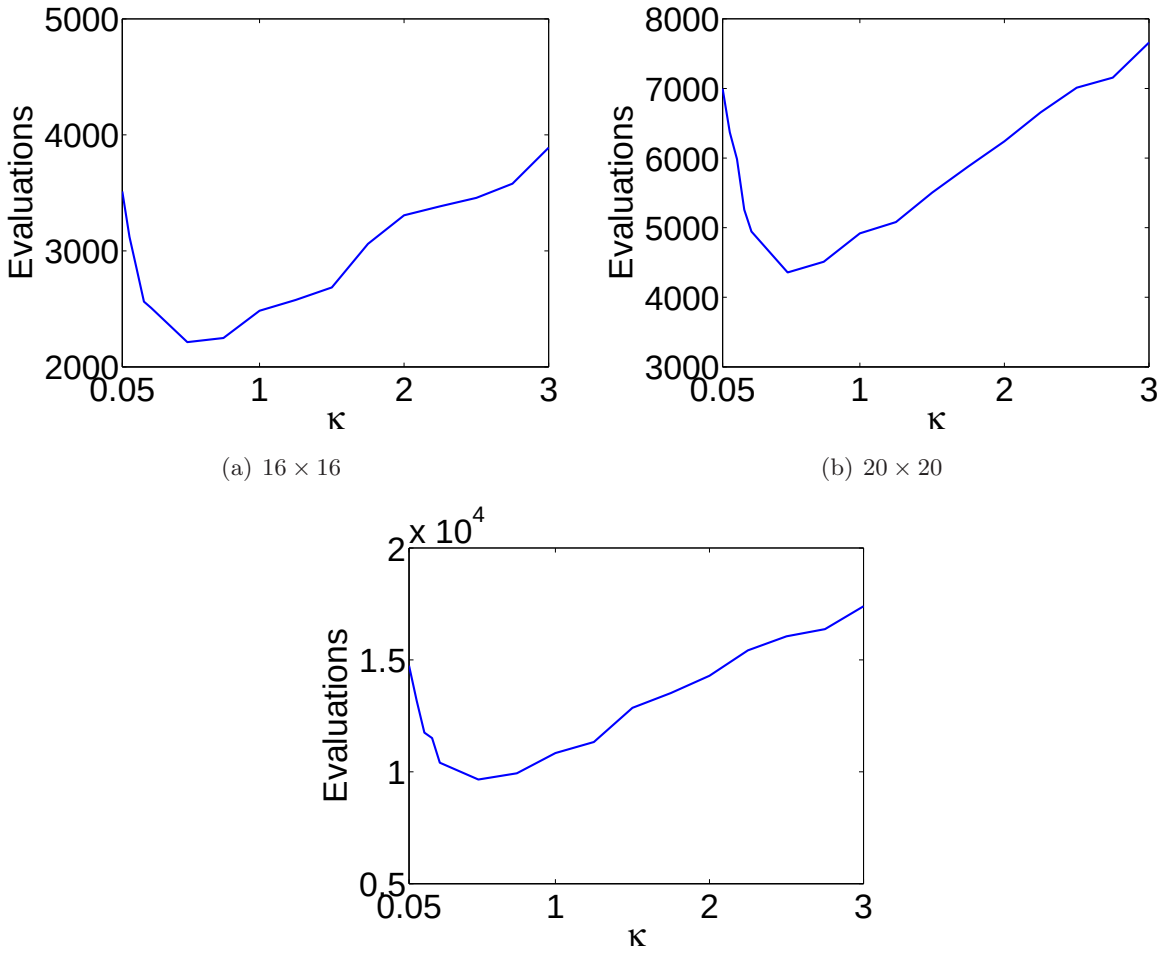
(c) 24×24

Figure 5: The effects of κ on the execution time of hBOA using SPM bias on the 2D Ising spin glass.

techniques to biasing model building, because these techniques required either substantial expert knowledge to design hand-crafted model restrictions, hand-inspection of the previously discovered models or they were very sensitive to one or more parameters.

While in this study we only considered two test problems—trap-5 and 2D Ising spin glasses—the proposed method should work on any problem that contains significant structural similarities between problem instances. These similarities may be defined in a more complex way, for example, via a problem-specific distance metric, as suggested in ref. (Hauschild, Pelikan, Sastry, & Goldberg, 2008). There are many examples of such problem classes, from graph theory (such as the minimum vertex cover and graph bipartioning) to MAXSAT and 3D spin glasses.

There are several key topics for future research in this area. First of all, the proposed technique should be tested on other important classes of problems. Second, it is important to explore the reasons for the different nature of the speedups obtained on trap-5 and the 2D Ising spin glass. Understanding the behavior of the proposed method on the spin glass problem and other difficult classes of problems will both provide inputs useful for the design of more effective approaches to biasing model building, as well as allow one to further improve the proposed method. Finally, other



(a) 16×16

(b) 20×20

(c) 24×24

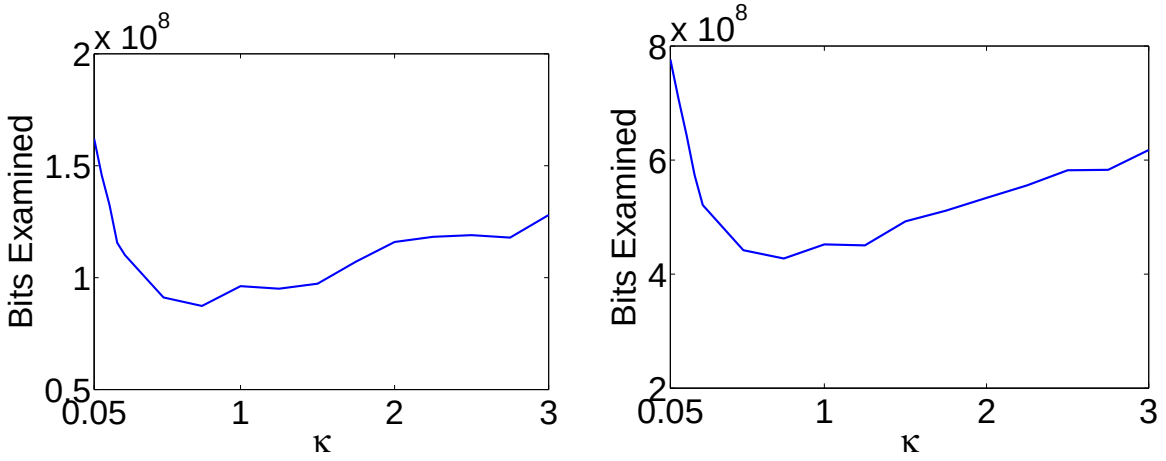
Figure 6: The effects of κ on the number of evaluations for hBOA using SPM bias on the 2D Ising spin glass.

approaches to gathering data from previous models should be examined in order to broaden the range of problems to which we can apply the proposed technique.

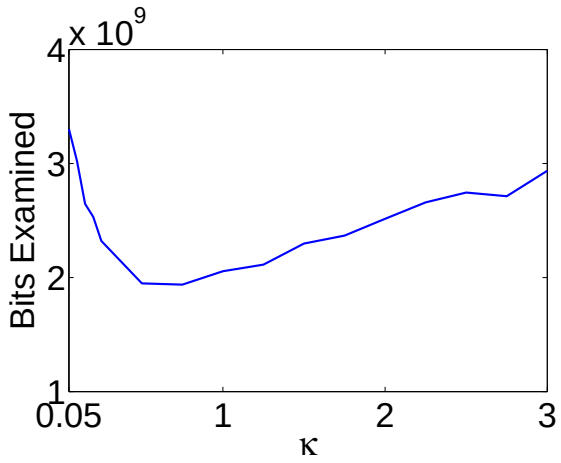
While EDAs provide optimization practitioners with a lot of information about the problem besides the high-quality solutions and while it is clear that using this information should substantially improve EDA performance when solving many problem instances of similar type, this research direction has not received much attention in the past. We believe that the study presented in this paper represents an important step toward the design practical and effective methods for exploiting problem-specific knowledge in form of the previously discovered probabilistic models in order to speed up future EDA runs on problems of similar type.

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,



(a) 16×16 (b) 20×20



(c) 24×24

Figure 7: The effects of κ on the number of bits examined in model building in hBOA using SPM bias on the 2D Ising spin glass.

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. Most experiments were completed at the Beowulf cluster at the University of Missouri–St. Louis.

References

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

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.

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.

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. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, *10*, 385–408.

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

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

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

Hauschild, M., Pelikan, M., Sastry, K., & Goldberg, D. E. (2008). Using previous models to bias structural learning in the hierarchical BOA. *Genetic and Evolutionary Computation Conf. (GECCO-2008)*, 415–422.

Heckerman, D., Geiger, D., & Chickering, D. M. (1994). *Learning Bayesian networks: The combination of knowledge and statistical data* (Technical Report MSR-TR-94-09). Redmond, WA: Microsoft Research.

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., Lobo, F. G., & Pelikan, M. (2008). From mating pool distributions to model overfitting. In *GECCO '08: Proc. of the 10th annual conference on Genetic and evolutionary computation* (pp. 431–438). New York, NY, USA: ACM.

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., Mahnig, T., & Rodriguez, A. O. (1998). *Schemata, distributions and graphical models in evolutionary optimization*. Submitted for publication.

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 Conf. (GECCO-2001)*, 511–518.

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

Pelikan, M., & Goldberg, D. E. (2006). Hierarchical Bayesian optimization algorithm. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable optimization via probabilistic modeling: From algorithms to applications* (pp. 63–90). Springer.

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., 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. *Int. Journal of Approximate Reasoning*, *31*(3), 221–258.

Santana, R., Larrañaga, P., & Lozano, J. A. (2007). The role of a priori information in the minimization of contact potentials by means of estimation of distribution algorithms. In Marchiori, E., Moore, J. H., & Rajapakse, J. C. (Eds.), *Proc. of the 5th European Conf. on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, Volume 4447 of *Lecture Notes in Computer Science* (pp. 247–257). Springer.

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.

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

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

Thierens, D. (1999). Scalability problems of simple genetic algorithms. *Evolutionary Computation*, *7*(4), 331–352.