Missouri Estimation of Distribution Algorithms Laboratory

Learn from the Past: Improving Model-Directed Optimization by Transfer Learning Based on Distance-Based Bias

Martin Pelikan and Mark W. Hauschild

MEDAL Report No. 2012007

August 2012

Abstract

For many optimization problems it is possible to define a problem-specific distance metric over decision variables that correlates with the strength of interactions between the variables. Examples of such problems include additively decomposable functions, facility location problems, and atomic cluster optimization. However, the use of such a metric for enhancing efficiency of optimization techniques is often not straightforward. This paper describes a framework that allows optimization practitioners to improve efficiency of model-directed optimization techniques by combining such a distance metric with information mined from previous optimization runs on similar problems. The framework is demonstrated and empirically evaluated in the context of the hierarchical Bayesian optimization algorithm (hBOA). Experimental results provide strong empirical evidence that the proposed approach provides significant speedups and that it can be effectively combined with other efficiency enhancements. The paper demonstrates how straightforward it is to adapt the proposed framework to other model-directed optimization techniques by presenting several examples.

Keywords

Model-directed optimization, transfer learning, estimation of distribution algorithms, hierarchical Bayesian optimization algorithm, decomposable problems, inductive transfer, learning from experience, efficiency enhancement.

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@medal-lab.org
WWW: http://medal-lab.org/

Learn from the Past: Improving Model-Directed Optimization by Transfer Learning Based on Distance-Based Bias

Martin Pelikan martin@martinpelikan.net
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), Department of Mathematics and Computer Science, University of Missouri, One University Blvd., St. Louis, MO 63121, USA

Mark W. Hauschild mwh308@umsl.edu
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), Department of Mathematics and Computer Science, University of Missouri, One University Blvd., St. Louis, MO 63121, USA

Abstract

For many optimization problems it is possible to define a problem-specific distance metric over decision variables that correlates with the strength of interactions between the variables. Examples of such problems include additively decomposable functions, facility location problems, and atomic cluster optimization. However, the use of such a metric for enhancing efficiency of optimization techniques is often not straightforward. This paper describes a framework that allows optimization practitioners to improve efficiency of model-directed optimization techniques by combining such a distance metric with information mined from previous optimization runs on similar problems. The framework is demonstrated and empirically evaluated in the context of the hierarchical Bayesian optimization algorithm (hBOA). Experimental results provide strong empirical evidence that the proposed approach provides significant speedups and that it can be effectively combined with other efficiency enhancements. The paper demonstrates how straightforward it is to adapt the proposed framework to other model-directed optimization techniques by presenting several examples.

1 Introduction

Even for large-scale optimization problems that are extremely difficult to solve, it may be straightforward to extract information about important dependencies between variables and other problem regularities directly from the problem definition (Baluja, 2006; Drezner and Salhi, 2002; Hauschild and Pelikan, 2010; Schwarz and Ocenasek, 2000a; Stonedahl et al., 2008). Furthermore, when solving many problem instances of similar type, it may be possible to gather information about variable interactions and other problem features by examining previous runs of the optimization algorithm, and to use this information to bias optimization of future problem instances to increase its speed, accuracy and reliability (Hauschild and Pelikan, 2008; Hauschild et al., 2012; Kaedi and Ghasem-Aghaee, 2011; Pelikan and Hauschild, 2012a,b; Santana et al., 2012). The use of information from previous runs to introduce bias into future runs of an evolutionary algorithm is often referred to as *learning from experience* (Hauschild and Pelikan, 2008; Hauschild et al., 2012; Pelikan, 2002). The use of bias based on the results of other learning tasks in the same problem domain is also commonplace in machine learning where it is referred to as *inductive transfer* or *transfer learning* (Pratt et al., 1991; Caruana, 1997). Numerous studies have shown that using prior knowledge and learning from experience promise improved efficiency and problem solving capabilities (Baluja, 2006; Drezner and Salhi, 2002; Hauschild and Pelikan, 2010; Hauschild et al., 2012; Rothlauf, 2006; Stonedahl et al., 2008; Schwarz and Ocenasek, 2000a). However, most prior work in this area was based on hand-crafted search operators, model restrictions, or representations.

This paper describes an approach that combines prior problem-specific knowledge with learning from experience. The basic idea of the proposed approach consists of (1) defining a problem-specific distance metric, (2) analyzing previous EDA models to quantify the likelihood and nature of dependencies at various distances, and (3) introducing bias into EDA model building based on the results of the analysis using

Bayesian statistics. One of the key goals of this paper is to develop an automated procedure capable of introducing bias based on a distance metric and prior EDA runs, without requiring much expert knowledge or hand-crafted model restrictions from the practitioner. Furthermore, the proposed approach is intended to be applicable in a more practical manner than other approaches to learning from experience. For example, the proposed approach makes it feasible to use prior runs on problems of a smaller size to introduce bias when solving problem instances of a larger size, and the bias can be introduced even when the importance of dependencies between specific pairs of variables varies significantly across the problem domain. Although this paper focuses on the hierarchical Bayesian optimization algorithm (hBOA) and additively decomposable functions (ADFs), the proposed approach can also be applied to other model-directed optimization techniques and other problem types. The paper describes a framework that can be used to adapt the proposed approach to a different context, and it outlines in more depth how the framework can be used in the context of the extended compact genetic algorithm (ECGA) (Harik, 1999), the dependency structure matrix genetic algorithm (DSM-GA) (Yu et al., 2003, 2009), and the linkage-tree genetic algorithm (LTGA) (Thierens, 2010a).

The paper is organized as follows. Section 2 describes hBOA. Section 3 outlines the framework for introducing bias based on prior runs on similar problems in model-directed optimization. Section 4 presents the proposed approach to introducing bias based on prior runs into hBOA model building for additively decomposable problems. Section 5 presents experimental results. Section 6 discusses extensions of the proposed approach to other model-directed optimization techniques. Section 7 summarizes and concludes the paper.

2 Hierarchical BOA

The hierarchical Bayesian optimization algorithm (hBOA) (Pelikan and Goldberg, 2001, 2003b; Pelikan, 2005) is an estimation of distribution algorithm (EDA) (Baluja, 1994; Larrañaga and Lozano, 2002; Pelikan et al., 2002a; Lozano et al., 2006; Pelikan et al., 2006; Hauschild and Pelikan, 2011). hBOA works with a population of candidate solutions represented by fixed-length strings over a finite alphabet. In this paper, candidate solutions are represented by n-bit binary strings. The initial population of binary strings is generated at random according to the uniform distribution over candidate solutions. Each iteration starts by selecting promising solutions from the current population; here binary tournament selection without replacement is used. Next, hBOA (1) learns a Bayesian network with local structures for the selected solutions and (2) generates new candidate solutions by sampling the distribution encoded by the built network (Chickering et al., 1997; Friedman and Goldszmidt, 1999). To maintain useful diversity in the population, the new candidate solutions are incorporated into the original population using restricted tournament selection (RTS) (Harik, 1995). The run is terminated when termination criteria are met. In this paper, each run is terminated either when the global optimum is found or when a maximum number of iterations is reached. Since the basic understanding of probabilistic models used in hBOA is necessary for the remainder of the paper, the rest of this section discusses the class of probabilistic models used in hBOA.

hBOA represents probabilistic models of candidate solutions by Bayesian networks with local structures (Chickering et al., 1997; Friedman and Goldszmidt, 1999). A Bayesian network is defined by two components: (1) an acyclic directed graph over problem variables specifying direct dependencies between variables and (2) conditional probabilities specifying the probability distribution of each variable given the values of the variable's parents. A Bayesian network encodes a joint probability distribution as

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

where X_i is the ith variable and Π_i are the parents of X_i in the underlying graph.

To represent conditional probabilities of each variable given the variable's parents, hBOA uses decision trees. Each internal node of a decision tree specifies a variable, and the subtrees of the node correspond to the different values of the variable. Each leaf of the decision tree for a particular variable defines the probability distribution of the variable given a condition specified by the constraints given by the path from the root of the tree to this leaf (constraints are given by the assignments of the variables along this path). Fig. 1 shows an example dependency tree encoding conditional probabilities $p(X_1|X_2, X_3, X_4)$ and the corresponding conditional probability table for binary variables.

X2	X3	X4	p(X1 | X2, X3, X4)
0	0	0	0.79
0	0	1	0.48
0	1	0	0.21
0	1	1	0.21
1	0	0	0.15
1	0	1	0.88
1	1	0	0.15
1	1	1	0.88

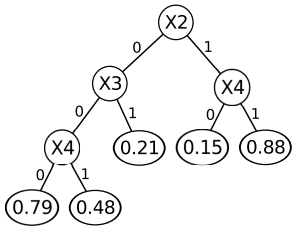


Figure 1: An example decision tree encoding conditional probabilities $p(X_1|X_2, X_3, X_4)$ and the corresponding conditional probability table. The decision tree encodes identical probabilities as the conditional probability table with only 5 parameters instead of 8. However, decision trees can often yield much more significant savings, because in practice the probabilities do not have to be encoded exactly.

To build probabilistic models, hBOA typically uses a greedy algorithm that initializes the decision tree for each problem variable X_i to a single-node tree that encodes the unconditional probability distribution of X_i. In each iteration, the model building algorithm tests how much a model would improve after splitting each leaf of each decision tree on each variable that is not already located on the path to the leaf. The algorithm executes the split that provides the most improvement, and the process is repeated until no more improvement is possible.

Improvement of the model after a split is often evaluated using the Bayesian-Dirichlet (BDe) metric with penalty for model complexity. Bayesian measures evaluate the goodness of a Bayesian network structure given data D and background knowledge ξ as (Cooper and Herskovits, 1992; Heckerman et al., 1994)

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

where c is a normalization constant. For the Bayesian-Dirichlet metric, the term $p(D|B, \xi)$ is estimated as (Chickering et al., 1997)

$$\begin{aligned} p(D|B, \xi) = \prod_{i=1}^{n} \prod_{l \in L_i} \frac{\Gamma(m'_i(l))}{\Gamma(m_i(l) + m'_i(l))} \\ \prod_{x_i} \frac{\Gamma(m_i(x_i, l) + m'_i(x_i, l))}{\Gamma(m'_i(x_i, l))}, \end{aligned} \tag{3}$$

where L_i is the set of leaves in the decision tree T_i for X_i; $m_i(l)$ is the number of instances in the selected population which end up the traversal through the tree T_i in the leaf l; $m_i(x_i, l)$ is the number of instances that have $X_i = x_i$ and end up the traversal of the tree T_i in the leaf l; $m'_i(l)$ represents the prior knowledge about the value of $m_i(l)$; and $m'_i(x_i, l)$ represents the prior knowledge about the value of $m_i(x_i, l)$. Without any prior knowledge, an uninformative prior $m'_i(x_i, l) = 1$ is typically used. To favor simpler networks to the more complex ones, the prior probability of each network decreases exponentially fast with respect to the description length of this network's parameters (Friedman and Goldszmidt, 1999; Pelikan, 2005):

$$p(B|\xi) = c2^{-0.5(\sum_i |L_i|) \log_2 N}, \tag{4}$$

where c is a normalization constant required for the prior probabilities of all network structures to sum to one. One may also modify the prior probability distribution $p(B|\xi)$ over network structures to prefer models that seem to match the problem description (Schwarz and Ocenasek, 1999), or to bias models according to the models obtained in previous runs (Hauschild et al., 2011; Kaedi and Ghasem-Aghaee, 2011; Pelikan and Hauschild, 2012a,b). The prior distribution $p(B|\xi)$ over network structures will be the main tool used in this paper for incorporating information mined from probabilistic models obtained in previous hBOA runs on similar problems.

3 Bias Based on Previous EDA Runs

Building an accurate probabilistic model in hBOA and other model-directed optimization algorithms can be time consuming and it may require rather large populations of solutions. That is why much effort has been

put into enhancing efficiency of model building in EDAs and improving quality of EDA models even with smaller populations of candidate solutions (Baluja, 2006; Hauschild and Pelikan, 2008, 2009; Höns, 2005; Hauschild et al., 2012; Lozano et al., 2001; Mendiburu et al., 2007; Mühlenbein and Mahnig, 2002; Ocenasek and Schwarz, 2000; Ocenasek, 2002; Ochoa et al., 2003; Pelikan et al., 2008; Santana et al., 2012). Learning from experience (Hauschild and Pelikan, 2008, 2009; Hauschild et al., 2012; Kaedi and Ghasem-Aghaee, 2011; Pelikan, 2005; Santana et al., 2012) represents one approach to dealing with this issue. In learning from experience (transfer learning), models discovered by EDAs in previous runs are mined to identify regularities and the discovered regularities are used to bias model building in future runs on problems of similar type. Since learning model structure is often the most challenging task in model building (Ocenasek, 2002), learning from experience often focuses on identifying regularities in model structure and using these regularities to bias structural learning in future runs.

It is straightforward to collect statistics on the most frequent dependencies in EDA models. Nonetheless, for the collected statistics to be useful, it is important to ensure that the statistics are meaningful with respect to the problem being solved. For example, consider optimization of NK landscapes (Kauffman, 1989), in which the fitness function is defined as the sum of n subfunctions $\{f_i\}_{i=1}^n$, and the subfunction f_i is applied to the ith bit and its k neighbors. The neighbors of each bit are typically chosen at random for each problem instance. Therefore, if we consider 1,000 problem instances of NK landscapes, looking at the percentage of models that included a dependency between the first two bits for the first 999 instances will not say much about the likelihood of the same dependency for the last instance. A similar observation can be made for many other important problem classes, such as MAXSAT or the quadratic assignment problem. That is why it is important to develop a more general framework that allows one to learn and use statistics on dependencies in EDA models across a range of problem domains of different structure and properties. In the remainder of this section we describe one such framework.

To formalize the framework for identification of structural regularities in EDA models, let us define a set of m dependency categories $D = \{D_1, \ldots, D_m\}$ and denote the background knowledge about the problem by ξ. Then, we can define a function $\gamma(i, j, \xi)$ that, given ξ, maps any dependency (i, j) covered by the probabilistic model into one of the m categories so that $\gamma(i, j, \xi) = k$ if and only if $(i, j) \in D_k$. Two straightforward possibilities for defining γ function were proposed by Hauschild and Pelikan (2008, 2009); Hauschild et al. (2012): (1) Each pair of problem variables X_i and X_j defines a unique category, and (2) categories are defined using a discretization of a problem-specific distance metric between variables. The first approach is useful especially when solving a number of instances of a problem where each variable has a fixed meaning across the entire set of instances; this is the case for example in spin glasses defined on a regular lattice, where every pair of variables in the lattice can be assigned a special category because the structure of the problem does not change from one instance to another (Hauschild et al., 2009). The latter approach is useful especially when one can define a distance metric on variables so that the distance between two variables correlates strongly with the likelihood or strength of their interaction; for example, one may define a distance metric such that variables that interact more strongly are closer to each other according to the metric. Such a distance metric can be defined for example in the quadratic assignment problem, traveling salesman problem or, more generally, classes of additively decomposable functions. While these two approaches are applicable to many important classes of problems, one may envision many other approaches based on this framework. The key issue in defining γ is that the categories should be related to the problem, so that each category contains pairs of variables that have a lot in common and that can be expected to be either correlated or independent most of the time.

The statistics obtained from previous EDA models can be used to bias the search operators of model-directed optimization methods using either a soft bias or a hard bias. A soft bias allows one to define preference to some models using a prior distribution over network structures or partial variable assignments (Schwarz and Ocenasek, 2000a; Hauschild and Pelikan, 2009; Hauschild et al., 2012; Kaedi and Ghasem-Aghaee, 2011; Pelikan and Hauschild, 2012a,b). A hard bias encodes hard restrictions on model structure or variable assignments, restricting the class of allowable models (Mühlenbein and Mahnig, 2002; Baluja, 2006; Hauschild and Pelikan, 2008; Hauschild et al., 2012). While in most prior work on bias in EDAs the bias was based on expert knowledge, in learning from experience the focus is on *automated learning* of a bias from past EDA runs.

In this paper we describe one way of using the above framework to facilitate learning from experience

in hBOA for additively decomposable problems based on a problem-specific distance metric. However, note that the framework can be applied to many other model-directed optimization techniques and the function γ can be defined in many other ways. To illustrate this, we outline how this approach can be extended to several other model-directed optimization techniques in section 6.

4 Distance-Based Bias

4.1 Additively Decomposable Functions

For many optimization problems, the objective function (fitness function) can be expressed in the form of an additively decomposable function (ADF) of m subproblems:

$$f(X_1, \ldots, X_n) = \sum_{i=1}^{m} f_i(S_i), \tag{5}$$

where $(X_1, \ldots, X_n)$ are problem's decision variables, f_i is the ith subfunction, and $S_i \subset \{X_1, X_2, \ldots, X_n\}$ is the subset of variables contributing to f_i (subsets $\{S_i\}$ can overlap). While they may often exist multiple ways of decomposing the problem using additive decomposition, one would typically prefer decompositions that minimize the sizes of subsets $\{S_i\}$. As an example, consider the following objective function for a problem with 6 variables:

$$f_{example}(X_1, X_2, X_3, X_4, X_5, X_6) = f_1(X_1, X_2, X_5) + f_2(X_3, X_4) + f_3(X_2, X_5, X_6).$$

In the above objective function, there are three subsets of variables, $S_1 = \{X_1, X_2, X_5\}$, $S_2 = \{X_3, X_4\}$ and $S_3 = \{X_2, X_5, X_6\}$, and three subfunctions $\{f_1, f_2, f_3\}$, each of which can be defined arbitrarily.

It is of note that the difficulty of ADFs is not fully determined by the order (size) of subproblems, but also by the definition of the subproblems and their interaction. In fact, there exist a number of NP-complete problems that can be formulated as ADFs with subproblems of order 2 or 3, such as MAXSAT for 3-CNF formulas. On the other hand, one may easily define ADFs with subproblems of order n that can be solved by a simple bit-flip hill climbing in low-order polynomial time.

4.2 Measuring Variable Distances for ADFs

The definition of a distance between two variables of an ADF used in this paper is based on the work of Hauschild and Pelikan (2008) and Hauschild et al. (2012). Given an additively decomposable problem with n variables, we define the distance between two variables using a graph G of n nodes, one node per variable. For any two variables X_i and X_j in the same subset S_k, that is, $X_i, X_j \in S_k$, we create an edge in G between the nodes X_i and X_j. See fig. 2 for an example of an ADF and the corresponding graph. Denoting by $l_{i,j}$ the number of edges along the shortest path between X_i and X_j in G (in terms of the number of edges), we define the distance between two variables as

$$D(X_i, X_j) = \begin{cases} l_{i,j} & \text{if a path between } X_i \text{ and } X_j \text{ exists, and} \\ n & \text{otherwise.} \end{cases} \tag{6}$$

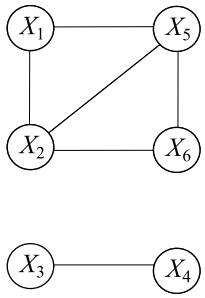
Fig. 2 illustrates the distance metric on a simple example. The above distance measure makes variables in the same subproblem close to each other, whereas for the remaining variables, the distances correspond to the length of the chain of subproblems that relate the two variables. The distance is maximal for variables that are completely independent (the value of a variable does not influence the contribution of the other variable in any way).

Since interactions between problem variables are encoded mainly in the subproblems of the additive problem decomposition, the above distance metric should typically correspond closely to the likelihood of dependencies between problem variables in probabilistic models discovered by EDAs. Specifically, the variables located closer with respect to the metric should more likely interact with each other. Fig. 3 illustrates this on two ADFs discussed later in this paper—the NK landscape with nearest neighbor interactions and the two-dimensional Ising spin glass (for a description of these problems, see section 5.1). The figure analyzes probabilistic models discovered by hBOA in 10 independent runs on each of the 1,000 random instances for each problem and problem size. For a range of distances d between problem variables, the figure

Subsets in additive decomposition

$S_1 = \{X_1, X_2, X_5\}$
$S_2 = \{X_3, X_4\}$
$S_3 = \{X_2, X_5, X_6\}$

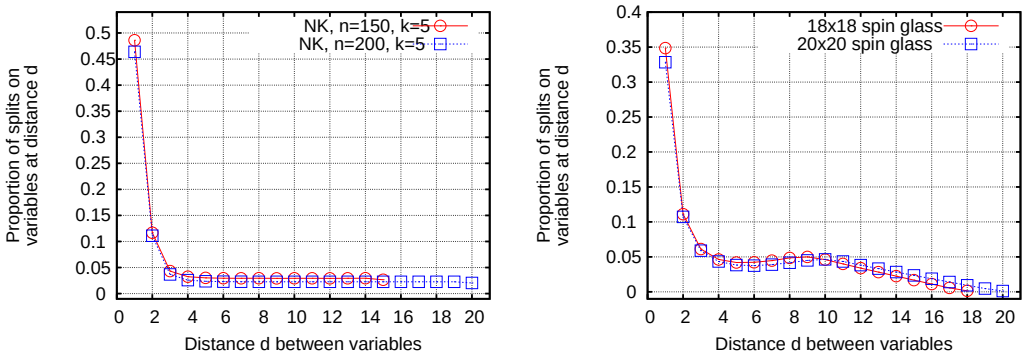
Corresponding dependency graph



Distance metric constructed from graph

D	X_1	X_2	X_3	X_4	X_5	X_6
X_1	0	1	6	6	1	2
X_2	1	0	6	6	1	1
X_3	6	6	0	1	6	6
X_4	6	6	1	0	6	6
X_5	1	1	6	6	0	1
X_6	2	1	6	6	1	0

Figure 2: An example of an additive decomposition, and the corresponding dependency graph and distance metric (displayed as a matrix).



(a) Nearest-neighbor NK.

(b) 2D spin glass.

Figure 3: Dependencies between variables that are closer to each other are stronger than the dependencies between other variables. Furthermore, the proportion of splits capturing dependencies between variables at a given distance changes only little between instances of various sizes. This indicates that the statistics acquired from models of one size should be useful for biasing hBOA runs on problems of another size. The results are based on 979,020 models obtained from hBOA on 1,000 unique problem instances of each problem, 10 runs per instance.

shows the proportion of splits on a variable located at distance d. The proportion of splits at distance d is the ratio of the number of splits on variable X_i in a dependency tree T_j for variable X_j with $D(X_i, X_j) = d$, and the total number of splits in the models. The results clearly support the fact that hBOA models indicate strongest dependencies between variables located close to each other according to the aforementioned metric and that there is a clear correlation between the distance metric and the likelihood of dependencies. Furthermore, the figure indicates that the likelihood of dependencies at a specific distance does not change much from one problem size to another, indicating that the bias based on these statistics should be applicable across a range of problem sizes.

It is important to note that other approaches may be envisioned to defining a distance metric for ADFs. For example, a weight may be added on each edge that would decrease with the number of subsets that contain the two connected variables. Another interesting possibility would be to consider the subfunctions themselves in measuring the distances, so that only correlations that lead to nonlinearities are considered or that some correlations are given priority over others. Finally, the distance of variables may depend on the problem definition itself, not on the decomposition only. For example, in the quadratic assignment problem, a distance between two facility locations is specified directly by the problem instance. The key is to use problem-specific information to specify a distance metric so that the distance between a pair of variables correlates with the likelihood or strength of their interaction. This will allow the approach to be useful even when the structure of the problem changes from instance to instance or when prior runs were

done on problems of different size or type; both of these features are demonstrated in section 5.

4.3 Using Distance-Based Bias in hBOA

The basic idea of incorporating the distance-based bias based on prior runs into hBOA is inspired mainly by the work of Hauschild et al. (Hauschild and Pelikan, 2009; Hauschild et al., 2011, 2012). Hauschild et al. proposed to incorporate learning from experience into hBOA by modifying prior probabilities of network structures using the statistics that capture the number of splits on each variable in the decision tree for each other variable in past hBOA runs on similar problems. Nonetheless, the approach of Hauschild et al. is only applicable to problems where the strength of interactions between any two variables is not expected to change much from instance to instance; that is why this approach can only be applied in a limited set of problem domains and it is difficult to use the approach when problem size is not fixed in all runs. In this paper, we propose to capture the *nature of dependencies between variables with respect to their distance* using the distance metric defined for ADFs or another distance metric. This allows one to apply the technique in more problem domains and also allows models from problem instances of one size to be useful in solving problem instances of different sizes (see fig. 3). Both these claims will be supported by a large number of experimental results presented in section 5. Note that the proposed approach is also closely related to the hard distance-based bias proposed by Hauschild and Pelikan (2008); Hauschild et al. (2012). However, in this paper, soft bias is incorporated for the entire range of distances instead of using hard bias that cuts off dependencies at distances greater than a user-specified threshold; this should provide for a more robust approach.

Recall that the BDe metric used to evaluate probabilistic models in hBOA contains two parts: (1) prior probability $p(B|\xi)$ of the network structure B, and (2) posterior probability $p(D|B,\xi)$ of the data (population of selected solutions) given B. In EDAs, prior probabilities of network structures typically either represent the uniform distribution over admissible network structures or encode a bias toward simple models regardless of the problem. However, the prior probability distribution of network structures can also be used to specify preferable structures. In this paper, we will use the prior probability distribution over network structures to introduce bias toward models that resemble models obtained in previous runs on problems of similar type with the focus on distance-based statistics. An analogous approach can be used to incorporate bias into hBOA for a different mapping γ of pairs of variables into dependency categories $\{D_i\}$; we describe necessary modifications in section 4.4.

Let us assume a set M of hBOA models from prior hBOA runs on similar ADFs. Before applying the bias by modifying the prior probability distribution of models in hBOA, the models in M are first processed to generate data that will serve as the basis for introducing the bias. The processing starts by analyzing the models in M to determine the number $s(m,d,j)$ of splits on any variable X_i such that $D(X_i,X_j)=d$ in a decision tree T_j for variable X_j in a model $m \in M$ (see fig. 4 for an example). Then, the values $s(m,d,j)$ are used to compute the probability $P_k(d,j)$ of a kth split on a variable at distance d from X_j in a dependency tree T_j given that $k-1$ such splits were already performed in T_j:

$$P_k(d,j) = \frac{|\{m \in M : s(m,d,j) \geq k\}|}{|\{m \in M : s(m,d,j) \geq k-1\}|}. \tag{7}$$

Given the terms $P_k(d,j)$, we can now define the prior probability of a network B as

$$p(B,\xi) = c \prod_{d=1}^{n} \prod_{j=1}^{n} \prod_{k=1}^{n_s(d,j)} P_k^{\kappa}(d,j), \tag{8}$$

where $n_s(d,j)$ denotes the number of splits on any variable X_i such that $D(X_i,X_j)=d$ in T_j, $\kappa > 0$ is used to tune the strength of bias (the strength of bias increases with κ), and c is a normalization constant. Since log-likelihood is typically used to assess model quality, to evaluate the contribution of any particular split, the main difference from the standard version of the BDe metric with the complexity penalty is that instead of reducing the metric according to the additional complexity of $\log_2(N)/2$ for each new split, we reduce the metric by the corresponding $\kappa \log_2 P_k(d,j)$. Therefore, the computation of the change of the prior probability of the network structure can still be done in constant time. Of course, the change in $p(D|B,\xi)$ requires computation of marginal frequencies, so it cannot be done in constant time.

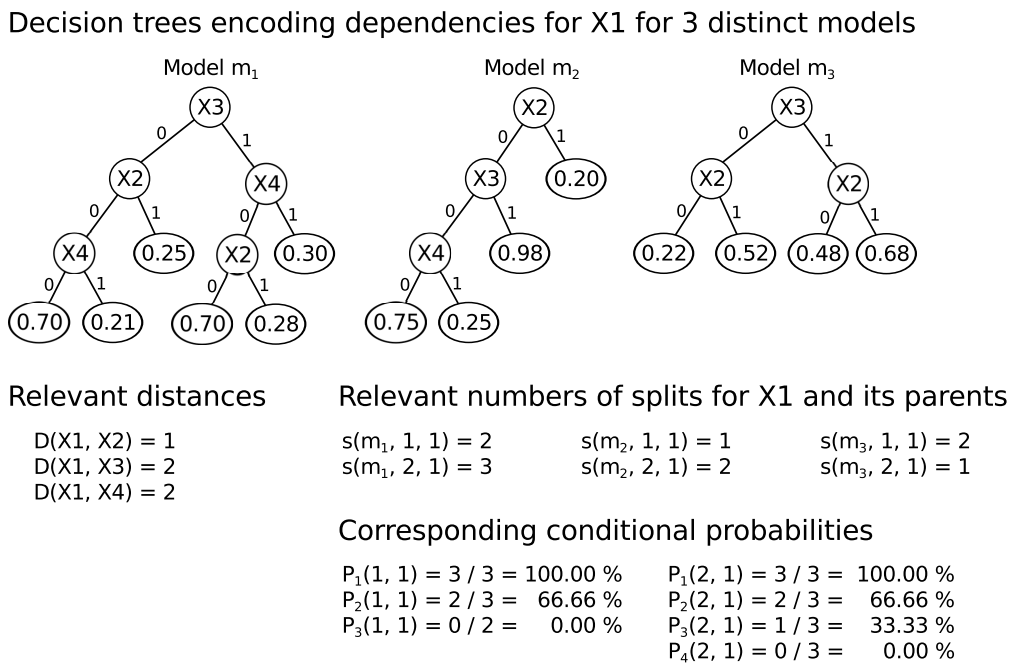


Figure 4: An example of the computation of $s(m, d, j)$ and $P_k(d, j)$ for variable X_1. In this example, X_1 has 3 parents, X_2, X_3, and X_4. The dependency trees encoding dependence of X_1 on its parents in three distinct models are shown on the top. Next, the distances between these variables (according to the distance metric) and values of $s(m, d, j)$ are shown. Finally, the conditional probabilities $P_k(d, j)$ are shown based on the values $s(m, d, j)$ on the bottom of the figure.

It is important to note that the prior probability of hBOA models defined in eq. (8) is certainly not the only possible approach to incorporating learning from experience using distance-based statistics into hBOA. The main source of inspiration for the proposed approach is the work on incorporating bias in learning Bayesian networks using Bayesian metrics (Heckerman et al., 1994) and the prior work on learning from experience in hBOA by Schwarz and Ocenasek (2000b) and Hauschild and Pelikan (2009). The experimental results presented in the next section confirm that this approach leads to substantial speedups in all the problem classes considered in this paper. Of course, there may be other ways to define $p(B, \xi)$ based on distance-based statistics from previous hBOA models that would yield even better speedups.

4.4 Generalizing the Approach Beyond Distance-Based Statistics

Section 3 presented the framework that can be used to formalize the use of bias based on prior runs of a model-directed optimization algorithm on problems of similar type and just now we described how this framework can be applied to hBOA using distance-based statistics. However, the presented approach to incorporating bias based on prior runs into hBOA and a number of similar EDAs is certainly not limited to the use of distance-based statistics. Here we describe how the approach can be used in a more general setting in the context of the aforementioned framework.

Recall that the framework presented in section 3 assumed that the pairs of variables are classified using a γ function into m categories $\{D_1, D_2, \ldots, D_m\}$. For distance-based statistics, the categories are defined using the distance metric; therefore, $\gamma(i, j, \xi) = D(i, j)$ where $D(\cdot)$ denotes the distance metric. Of course, in general the γ function can be defined arbitrarily. In the general case, to introduce bias into hBOA, one can start by computing the number $s(m, q, j)$ of splits in a dependency tree T_j in the model m such that $\gamma(i, j, \xi) = q$. Then, the probability $P_k(q, j)$ of a kth split on a variable X_i in a dependency tree T_j such that $\gamma(i, j, \xi) = q$ given that $k - 1$ such splits were already performed in T_j can be computed analogically to

eq. (7):

$$P_k(q,j) = \frac{|\{m \in M : s(m,q,j) \geq k\}|}{|\{m \in M : s(m,q,j) \geq k-1\}|}. \tag{9}$$

Then, the prior probability of a network B can be computed as

$$p(B,\xi) = c \prod_{q=1}^{m} \prod_{j=1}^{n} \prod_{k=1}^{n_s(q,j)} P_k^{\kappa}(q,j), \tag{10}$$

where $n_s(d,j)$ denotes the number of splits on any variable X_i in T_j such that $\gamma(X_i, X_j, \xi) = q$, $\kappa > 0$ is used to tune the strength of bias, and c is a normalization constant. Note that the only differences between the prior probability for the general case and the one for the distance-based bias are minor; essentially, instead of dealing with n categories based on the distance metric we must consider m categories $\{D_i\}$ defined by the γ function.

5 Experiments

5.1 Test Problems

To test the proposed approach to biasing hBOA model building using distance-based statistics from prior hBOA runs, we consider the following problem classes:

1. shuffled nearest-neighbor NK landscapes,
2. minimum vertex cover for graphs with a fixed ratio of the number of edges to the number of nodes,
3. two-dimensional and three-dimensional Ising spin glasses with ±J couplings,
4. MAXSAT instances created from graph coloring of graphs that combine structure and randomness.

All these problem classes are known to be challenging for conventional genetic and evolutionary algorithms and many other optimization techniques due to the rugged landscape, strong epistasis, and complex structure of interactions between variables. For each problem, a variety of problem settings and a number of instances for each setting are considered in order to provide a more informative set of statistics that demonstrate benefits of the proposed approach. The test problems are described in more detail next.

Nearest-neighbor NK landscapes. An NK fitness landscape (Kauffman, 1989) is fully defined by the following components: (1) The number of bits, n, (2) the number of neighbors per bit, k, (3) a set of k neighbors $\Pi(X_i)$ of the ith bit for every $i \in \{1,\ldots,n\}$, and (4) a subfunction f_i defining a real value for each combination of values of X_i and $\Pi(X_i)$ for every $i \in \{1,\ldots,n\}$. Typically, each subfunction is defined as a lookup table. The objective function f_{nk} to maximize is defined as

$$f_{nk}(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} f_i(X_i, \Pi(X_i)). \tag{11}$$

The difficulty of optimizing NK landscapes depends on all components defining an NK problem instance. In this paper, we consider nearest-neighbor NK landscapes, in which neighbors of each bit are restricted to the k bits that immediately follow this bit. The neighborhoods wrap around; thus, for bits which do not have k bits to the right, the neighborhood is completed with the first few bits of solution strings. The reason for restricting neighborhoods to nearest neighbors was to ensure that the problem instances can be solved in polynomial time even for $k > 1$ using dynamic programming (Pelikan, 2010). The subfunctions are represented by look-up tables (a unique value is used for each instance of a bit and its neighbors), and each entry in the look-up table is generated with the uniform distribution from $[0, 1)$. The used class of NK landscapes with nearest neighbors is thus the same as that in Pelikan (2010). In all experiments, we use $k = 5$ and $n \in \{100, 150, 200\}$. For each n, we use 1,000 unique, independently generated instances; overall, 3,000 unique instances of NK landscapes were tested.

Minimum vertex cover. In the minimum vertex cover, the task is to select a minimum number of nodes in a given graph so that every edge has at least one node in the selected set of nodes. Candidate solutions represent subsets of nodes using binary strings with one bit per node. To ensure that candidate solutions represent valid vertex covers, a repair operator proposed by Pelikan et al. (2007) was used. The repair operator adds nodes from uncovered edges at random, until a valid cover is obtained. Then, some nodes are eliminated if this does not violate the cover (each edge must still have at least one node in the selected set of nodes). The fitness is then given by the number of nodes *not* in the cover, which is to be maximized.

In this paper, we focus on the class of random graphs with a fixed ratio of the number of edges and the number of nodes. Specifically, denoting the number of edges by $|E|$ and the number of nodes by $|V|$, the ratio $c = |E|/|V|$ is fixed to $c = 2$ and $c = 4$, respectively, and for each value of c, we generate 1,000 unique graphs for each of the three considered graph sizes (150 nodes, 175 nodes, and 200 nodes). Edges in each instance are distributed at random, so the degrees of graph nodes are expected to vary. Altogether, we consider 6,000 instances of minimum vertex cover.

Ising spin glasses on 2D and 3D lattices with ±J couplings and periodic boundary conditions. Ising spin glasses are prototypical models for disordered systems (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 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

$$E(C) = -\sum_{\langle i,j\rangle} s_i J_{i,j} s_j, \tag{12}$$

where the sum runs over all couplings $\langle i, j\rangle$. Here the task is to find a spin configuration for a given set of coupling constants that minimizes the energy of the spin glass. The states with minimum energy are called *ground states*. The spin configurations are encoded with binary strings where each bit specifies the value of one spin (0 for a spin +1, 1 for a spin -1). One generally analyzes a large set of random spin glass instances for a given distribution of the spin-spin couplings. In this paper we consider the $\pm J$ spin glass, where each spin-spin coupling is set randomly to either $+1$ or -1 with equal probability.

For the 2D spin glass, we use instances arranged on square, two-dimensional, four-neighbor grids of sizes 10×10, 12×12, 14×14, 16×16, 18×18 and 20×20 spins; that is, the problem sizes range from 100 to 400 spins. We consider periodic boundary conditions. For each problem size, we use 1,000 unique, independently generated problem instances; overall, 6,000 unique instances of the 2D spin glass were tested. Furthermore, we consider instances on three-dimensional, 6-neighbor grids of sizes $5 \times 5 \times 5$, $6 \times 6 \times 6$, and $7 \times 7 \times 7$, with 1,000 unique, independently generated instances for each problem size. The total number of 3D spin glass instances was thus 3,000. All instances were obtained from the Spin Glass Ground State Server (Spin Glass Server, 2004).

Maximum Satisfiability (MAXSAT). MAXSAT for 3-CNF (conjunctive normal form) formulas was also examined. In MAXSAT, candidate solutions represent interpretations of propositions, and the fitness is given as the number of satisfied clauses in a CNF formula.

MAXSAT instances were created by mapping instances of graph coloring for random graphs that combine regular ring lattices (with probability $1 - p$) and random graphs (with probability p) (Gent et al., 1999; Pelikan and Goldberg, 2003a); 100 unique problem instances of $n = 500$ bits (propositions) were used for each considered value of p, from $p = 2^{-8}$ (graphs nearly identical to a regular ring lattice) to $p = 2^{-1}$ (graphs with half of the edges random). All tested instances were satisfiable. The main motivation for selecting this type of MAXSAT instances was that these instances are known to be difficult for WalkSAT (Gent et al., 1999; Pelikan and Goldberg, 2003a; Pelikan, 2005), which is one of the most successful heuristics for solving MAXSAT instances. The instances were downloaded from SATLIB (Hoos and Stützle, 2000).

5.2 Experimental Setup

This section describes the basic experimental setup.

Basic settings. The maximum number of iterations for each problem instance was set to the overall number of bits in the problem; according to preliminary experiments, this upper bound substantially exceeded the actual number of iterations required to solve each problem. Each run was terminated either when the global optimum was found, when the population consisted of copies of a single candidate solution, or when the maximum number of iterations was reached. For each problem instance, we used bisection (Sastry, 2001; Pelikan, 2005) to ensure that the population size was within 5% of the minimum population size to find the optimum in 10 out of 10 independent runs.

The proposed approach to distance-based bias in hBOA is tested for $\kappa = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ to assess how the strength of the bias represented by κ affects hBOA performance.

Local search. Bit-flip local search is used to improve every solution in the population until a local optimum is reached. In each iteration of local search, solutions obtained by changing any single bit are examined and the one that improves the solution the most is retained. The process is repeated until no more single-bit flip improves the solution and the solution is thus locally optimal with respect to single-bit modifications. Since in the minimum vertex cover the repair operator ensures that the solution is locally optimal, we do not use any additional local search on this problem. Every solution undergoes local search for all other problem classes.

Using local search enabled us to examine problem instances of relatively large size, and for each problem setting we could consider a large number of instances. This helped eliminate the possibility of missing a particular type of problem instances or focusing on impractically small problem sizes. Furthermore, a wealth of results exist that confirm that the use of local search improves performance of EDAs and it is thus likely that practitioners will use local search in EDA applications; that is why the use of local search is simply a more practical scenario to examine. However, to examine how the proposed approach works *without local search*, we also present a series of experiments that do *not* use any local search at all.

Quantifying hBOA performance and benefits of the proposed approach. To evaluate hBOA performance, we focus on (1) the execution time per run, (2) the required population size, and (3) the number of iterations (generations) until the global optimum. To evaluate the benefits of distance-based bias, the paper uses *multiplicative speedups*, where the speedup is defined as a multiplicative factor by which a particular complexity measure improves by using the distance-based bias compared to the base case with no distance-based bias. For example, an execution-time speedup of 2 indicates that the bias allowed hBOA to find the optimum twice as fast as without the bias. We used the most efficient implementation of hBOA available for the base case with no bias and we only modified it for the remaining cases to incorporate the bias. Although the code could be further optimized for efficiency, the primary focus of our experiments concerning the execution times was on the *speedups* of the CPU times rather than their absolute values.

10-fold crossvalidation. To ensure that the same problem instances were *not* used for defining the bias as well as for testing it, 10-fold crossvalidation was used. For most problems except for MAXSAT, 1,000 random problem instances were used in the experiments for each problem setting (size and parameters). The 1,000 instances in each set were randomly split into 10 equally sized subsets of 100 instances each. In each round of crossvalidation, 1 subset of 100 instances was left out and hBOA was run on the remaining 9 subsets of 900 instances total. The runs on the 9 subsets produced a number of models that were analyzed in order to obtain the probabilities $P_k(d, j)$ for all d, j, and k. The bias based on the obtained values of $P_k(d, j)$ was then used in hBOA runs on the remaining subset that was left out. The same procedure was repeated for each subset; overall, 10 rounds of crossvalidation were performed for each set of 1,000 instances. Each problem instance was used exactly once in the test of the proposed approach to biasing hBOA models and in every test, models used to generate statistics for hBOA bias were obtained from hBOA runs on different problem instances. For MAXSAT, the procedure was identical except that the number of instances for each problem setting was only 100 and the subsets thus contained 10 instances each.

Testing bias for problems of different size. We also performed experiments to evaluate benefits of the proposed approach in the scenario where the bias was based on hBOA runs on problems of *smaller* size. This is a common situation for optimization practitioners, who often start by applying an optimization technique to smaller, toy instances of the target problem and then use the results obtained to guide the solution of larger, more difficult problem instances. Since in this case, models used to create the bias are guaranteed to come from different problem instances, there is no need for crossvalidation. Instead, we selected two problem sizes for each problem. The bias was based on the probabilities $P_k(d, j)$ for models obtained on the smaller of the two problem sizes, and the bias was then incorporated into hBOA on the larger of the two problem sizes.

Comparisons with respect to execution time. While the experiments were performed across a variety of computer architectures and configurations, it was always ensured that the base case with no bias and the case with bias were both run on the same computational node and the results of the two runs can therefore be compared against each other with respect to the actual CPU time.

5.3 Results

This section presents and discusses experimental results. The section starts by presenting the results obtained using hBOA with local search. Next, the section shows the results for hBOA *without* local search. The section then discusses the results of using bias based on problem instances of *smaller* size and on problem instances from a *different class* of problems. The section closes by looking at the combined effect of sporadic model building *and* the proposed approach to distance-based bias based on prior runs.

5.3.1 Results with Local Search

Fig. 5 shows the effects of κ on multiplicative speedups with respect to the execution time, the population size, and the number of iterations (generations) on NK landscapes, minimum vertex cover, and 2D and 3D spin glasses. Fig. 6 shows analogical results for MAXSAT. The results confirm that, for adequate values of κ, the speedups in terms of execution time were significant for all the problem classes. The maximum speedup for NK landscapes was over 2.27; for minimum vertex cover it was over 3.15 for $c = 2$ and over 2.33 for $c = 4$; for spin glasses it was over 1.66 for the 2D case and over 1.34 for the 3D case; and for MAXSAT it was from over 1.39 to over 3.08 depending on the type of instances. For NK landscapes, minimum vertex cover, and most MAXSAT instances, the speedups increased with κ. For 2D and 3D spin glasses and one MAXSAT type, the speedups increased until an optimal value of κ and then decreased again as κ was increased further. However, in all cases a broad range of values of κ led to significant speedups, confirming robustness of the proposed approach. The results indicated that the reduction in the population sizes was one of the most important factors contributing to the reduction in the overall computational cost. We believe that the main reason for this is that the population sizing in hBOA and similar EDAs is driven primarily by the need for building accurate probabilistic models (Pelikan et al., 2002b; Yu et al., 2007). A useful bias makes it easier to learn accurate probabilistic models and population sizes can thus decrease. A decrease in the population size in turn translates into a decrease in execution time due to the reduction in both the number of evaluations of candidate solutions as well as the time to build each model.

Fig. 7 shows the percentage of instances for different problems and different settings. These results confirm that distance-based bias improves hBOA execution times on a great majority of instances for a broad range of κ. This further supports significance of the improvements due to distance-based bias.

Fig. 8 shows the speedups obtained with respect to problem size for a range of values of κ and all problems for which a range of problem sizes was considered; these results are useful for visualizing the relationship between the problem size and the speedups obtained with the distance-based bias. The results confirm that for NK landscapes, the speedups grew at least linearly with the problem size, regardless of the value of κ. For 2D spin glass, the speedups also grow with problem size but only for the best value of $\kappa = 4$. For most other problems, the speedups either change nonmonotically or they slightly decrease with problem size. It is important to note though that here the dependence of speedups on problem size is analyzed with respect to a fixed value of κ, but in practice, one can tune κ as problem size increases in order to maximize the benefits.

In summary, figs. 5, 6, 7, and 8 provide ample empirical evidence that the speedups obtained are substantial and that the proposed approach to learning from experience is useful in practice.

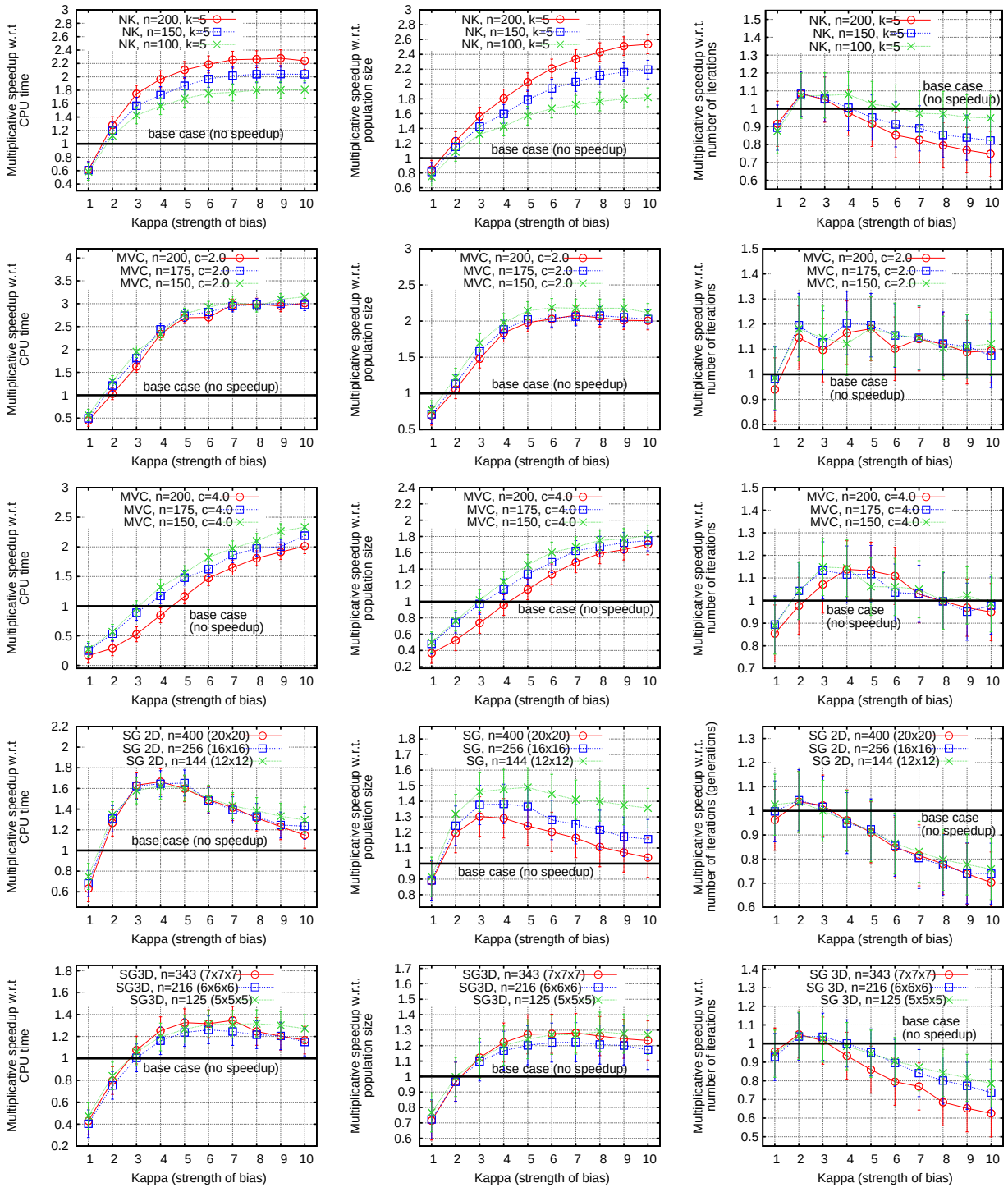


Figure 5: Speedups obtained with respect to the overall CPU time per run, the population sizes, and the numbers of iterations (generations) until the optimum was reached. The problems listed from top to bottom are NK landscapes with nearest neighbors and $k = 5$; the minimum vertex cover for regular graphs with fixed ratio $c = 2$ and $c = 4$ of the number of edges and the number of nodes; the two dimensional Ising spin glass with $\pm J$ couplings; and the three-dimensional Ising spin glass with $\pm J$ couplings. The focus of this figure is on the effects of κ (strength of bias) on the speedups obtained.

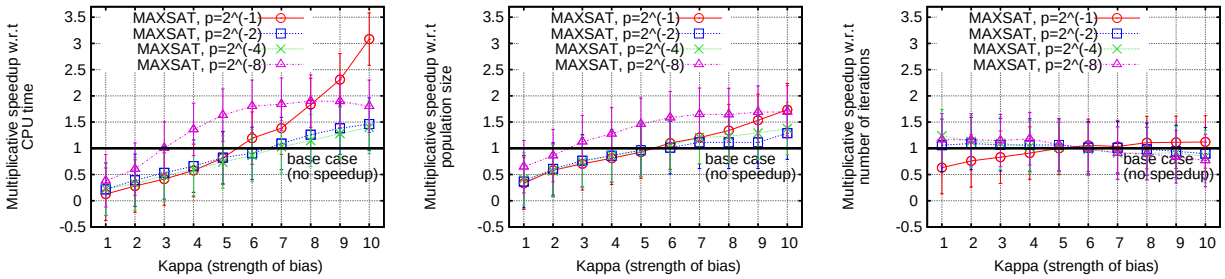


Figure 6: Speedups obtained on MAXSAT with respect to the overall CPU time per run, the population sizes, and the numbers of iterations (generations) until the optimum was reached. MAXSAT instances are created by transforming graph coloring problem instances for graphs created by combining regular ring lattices and fully random graphs; the parameter p controls the balance between structure and randomness.

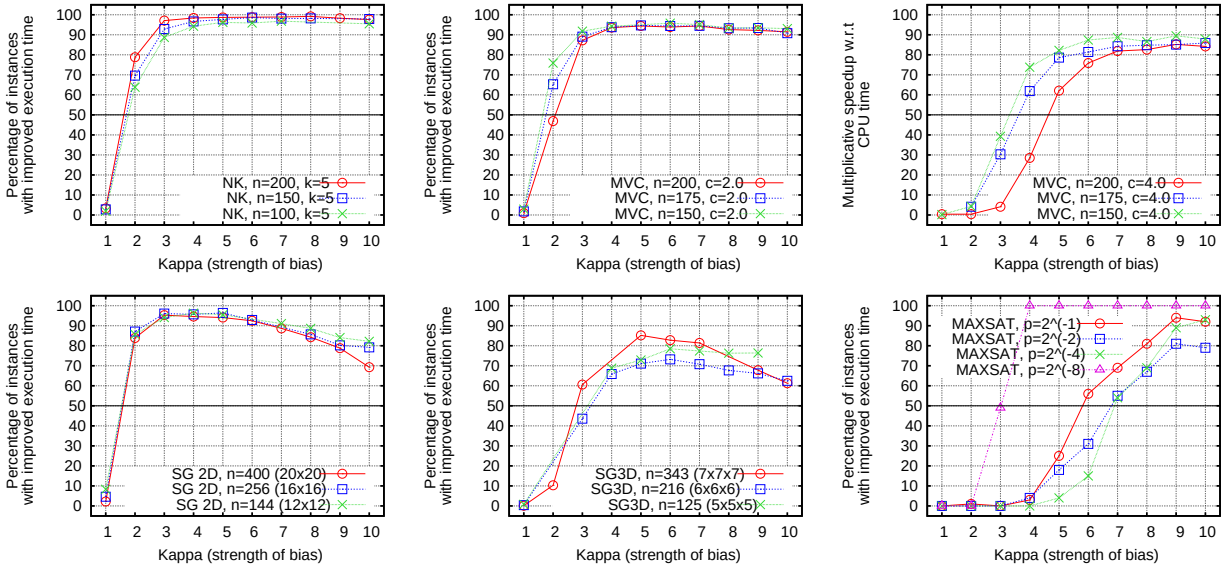
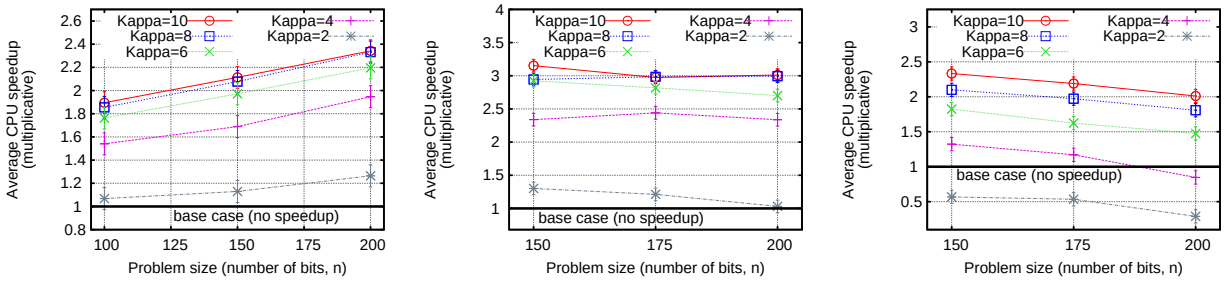
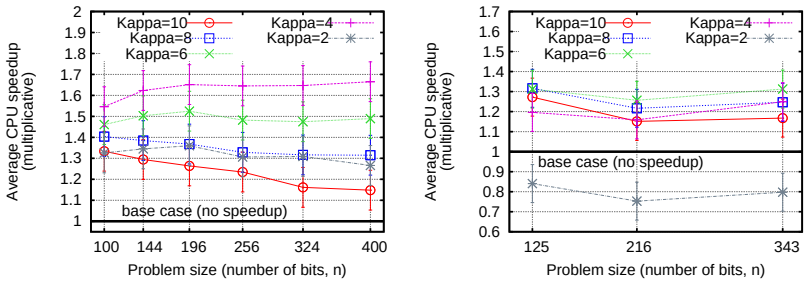


Figure 7: The percentages of hBOA instances for which execution time strictly improved using distance-based bias for a range of values of κ and all the test problems.



(a) NK landscapes with nearest neighbors. (b) Minimum vertex cover, $c = 2$ (c) Minimum vertex cover, $c = 4$



(d) 2D $\pm J$ Ising spin glass (e) 3D $\pm J$ Ising spin glass

Figure 8: Effect of problem size on the speedups obtained with respect to the overall CPU time per run.

5.3.2 Results without Local Search

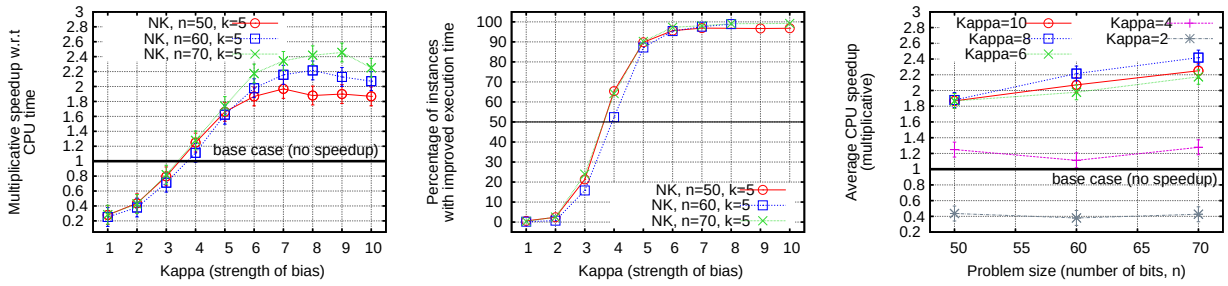
To provide further empirical evidence of the benefits of distance-based bias based on prior runs, we applied the approach to two of the problem classes *without* local search. Since execution times without using local search were much worse, these experiments use much smaller problem sizes than those considered in other experiments presented in this section. The results are shown in fig. 9. The results on both NK landscapes and 2D spin glasses show that, for adequate values of κ, the speedups obtained for hBOA without local search are even better than those obtained for hBOA with local search. However, as was discussed earlier in this paper and also in a number of other published studies, local search significantly improves performance of hBOA with or without distance-based bias, and the use of hBOA with local search is thus likely going to be a better approach to use in optimization practice.

5.3.3 Learning from Smaller Problems to Solve Bigger Ones

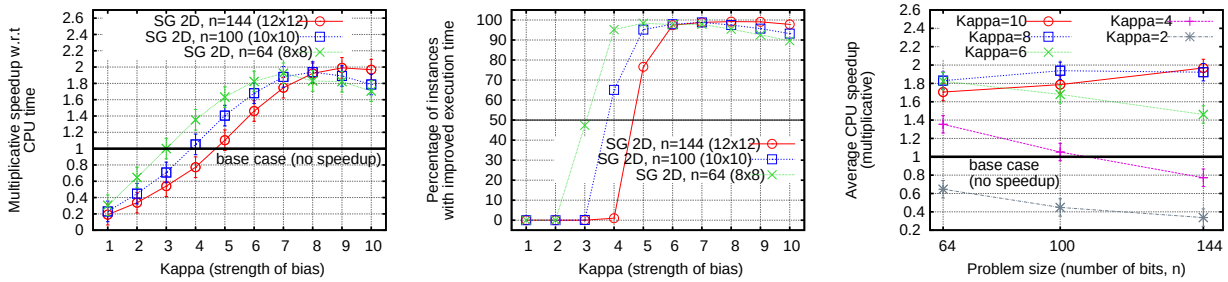
We mentioned that one of the primary advantages of the proposed approach to transfer learning using distance-based bias is that it should be applicable also when prior runs were obtained on problems of smaller size. Fig. 10 shows the speedups obtained with the bias based on problems of smaller size for all test problem classes except for MAXSAT. MAXSAT was excluded because problem instances of only one size were available. In all cases, significant speedups were obtained even with the bias based on prior runs on problems of smaller size. In fact, in a number of cases, the results outperformed the base case with the bias based on the problems of the same size. This indicates that while introducing bias based on prior runs is beneficial across a broad range of test cases, one may imagine introducing even more effective bias. Nonetheless, all results indicate that the bias can be used to speed up solution of problems of larger size when prior runs are available for problems of smaller size.

5.3.4 Using Prior Runs from Another Problem Class

One of the interesting questions we were asked while discussing preliminary results of distance-based bias with other researchers in evolutionary computation and machine learning was whether one can bias hBOA runs using information mined from hBOA models on a *different* problem class. For example, what would happen if we stored models obtained while solving the minimum vertex cover instances, and used those models to bias runs on NK landscapes or spin glasses? To help answer this question, we performed a series

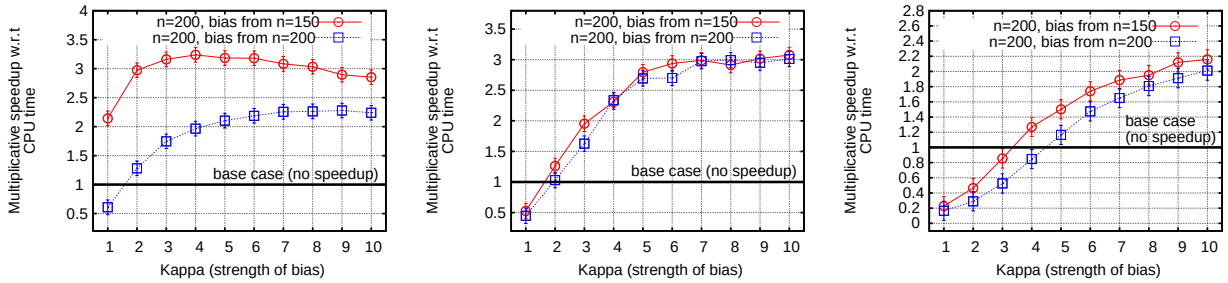


(a) NK landscapes with nearest neighbors.



(b) 2D $\pm J$ Ising spin glass

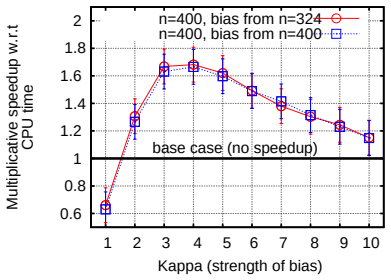
Figure 9: Speedups obtained on NK landscapes and 2D spin glasses *without* using local search.



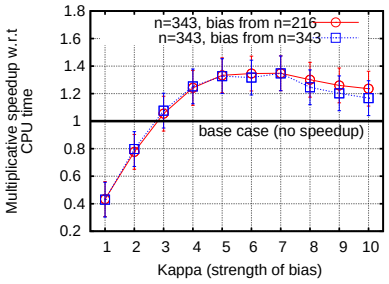
(a) NK landscapes with nearest neighbors, $n = 200$, $k = 5$.

(b) Minimum vertex cover, $n = 200$, $c = 2$.

(c) Minimum vertex cover, $n = 200$, $c = 4$.

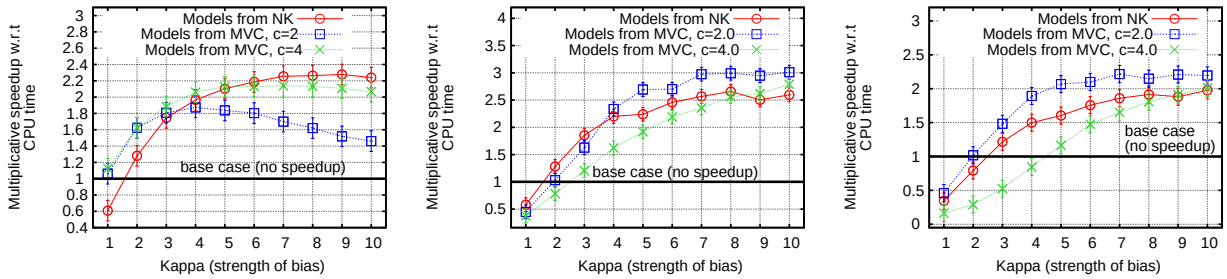


(d) 2D $\pm J$ Ising spin glass, $n = 20 \times 20 = 400$



(e) 3D $\pm J$ Ising spin glass, $n = 7 \times 7 \times 7 = 343$

Figure 10: Speedups obtained on all test problems except for MAXSAT with bias based on models obtained from problems of *smaller* size, compared to the base case with bias based on the same problem size.



(a) NK landscapes with nearest neighbors, $n = 200$, $k = 5$. (b) Minimum vertex cover, $n = 200$, $c = 2$. (c) Minimum vertex cover, $n = 200$, $c = 4$.

Figure 11: Speedups obtained on NK landscapes and minimum vertex cover with the bias from models on another problem class compared to the base case (the base case corresponds to 10-fold crossvalidation using the models obtained on the problems from the same class).

of experiments; the results of these experiments can be found in fig. 11. Specifically, the figure presents the results of using bias from the NK landscapes while solving the minimum vertex cover instances for the two sets of graphs (with $c = 2$ and $c = 4$), the results of using bias from the minimum vertex cover instances for both the classes of graphs when solving the NK landscapes, and also the results of using bias from the minimum vertex cover on another class of graphs (with different c). It can be seen that significant speedups were obtained even when using models from another problem class, although in some cases the speedups obtained when using the same problem class were significantly better. Some combinations of problem classes discussed in this paper were omitted because of the high computational demands of testing every pair of problems against each other. Despite that, the results presented here indicate that even the bias based on problems from a different class can be beneficial.

5.3.5 Combining Distance-Based Bias and Sporadic Model Building

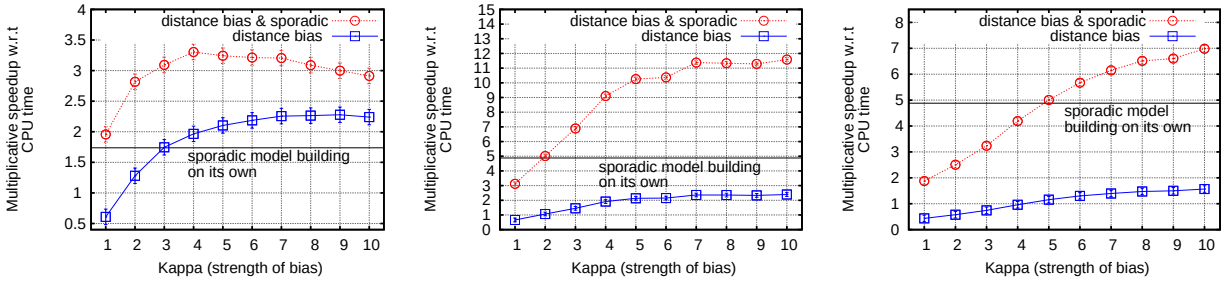
While most efficiency enhancement techniques provide significant speedups on their own right (Hauschild and Pelikan, 2008; Hauschild et al., 2012; Lozano et al., 2001; Mendiburu et al., 2007; Ocenasek, 2002; Pelikan et al., 2008; Sastry et al., 2006), their combinations are usually significantly more powerful because the speedups often multiply. For example, with the speedup of 4 due to sporadic model building (Pelikan et al., 2008) and the speedup of 20 due to parallelization using 20 computational nodes (Lozano et al., 2001; Ocenasek and Schwarz, 2000), combining the two techniques can be expected to yield speedups of about 80; obtaining comparable speedups using parallelization on its own would require at least 4 times as many computational nodes. How will the distance-based bias from prior hBOA runs combine with other efficiency enhancements?

To illustrate the benefits of combining the distance-based bias discussed in this paper with other efficiency enhancement techniques, we combined the approach with sporadic model building and tested the two efficiency enhancement techniques separately and in combination. Fig. 12 shows the results of these experiments. The results show that even in cases where the speedup due to distance-based bias is only moderate, the overall speedup increases substantially due to the combination of the two techniques. Nearly multiplicative speedups can be observed in most scenarios.

6 Thoughts on Extending the Approach to Other Model-Directed Optimizers

6.1 Extended Compact Genetic Algorithm

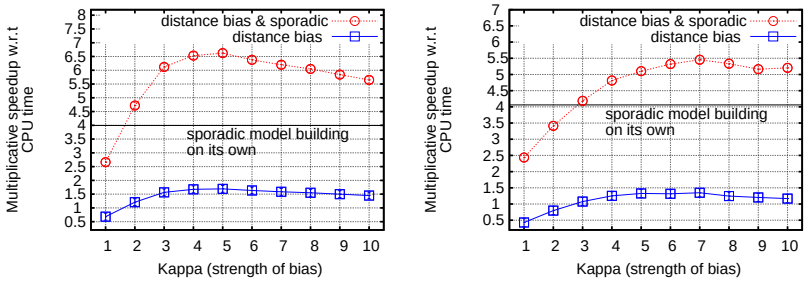
The extended compact genetic algorithm (ECGA) (Harik, 1999) uses marginal product models (MPM) to guide optimization. An MPM splits variables to disjoint linkage groups. The distribution of variables in each group is encoded by a probability table that specifies a probability for each combination of values of these variables. The overall probability distribution is given by the product of the distributions of the individual linkage groups. In comparison with Bayesian network models, instead of connecting variables that influence each other strongly using directed edges, ECGA puts such variables into the same linkage



(a) NK landscapes with nearest neighbors, $n = 200$, $k = 5$.

(b) Minimum vertex cover, $n = 200$, $c = 2$.

(c) Minimum vertex cover, $n = 200$, $c = 4$.



(d) 2D $\pm J$ Ising spin glass, $n = 20 \times 20 = 400$

(e) 3D $\pm J$ Ising spin glass, $n = 7 \times 7 \times 7 = 343$

Figure 12: Speedups obtained on all test problems except for MAXSAT with and without sporadic model building and the comparison of these speedups to the base case with only sporadic model building.

group. In other words, variables in any linkage group are expected to interact more strongly than variables located in different linkage groups.

One way to mine information from past models of ECGA using a γ function discussed in section 3 (such as a distance metric) is to estimate the probability that pairs of variables (X_i, X_j) that fall into the same category $q \in \{1, 2, \ldots, m\}$ according to the γ function are located in the same linkage group. This probability can be estimated by iterating through all models, and for each model, computing (1) the number of times that a pair (X_i, X_j) such that $\gamma(X_i, X_j) = q$ was encountered in the same linkage group and (2) the number of times that such a pair was encountered regardless of being in the same group. Then, the ratio of the number of times such pairs were located in the same linkage group and the total number of such pairs gives an estimate of the probability of a pair of variables in category q being in the same linkage group in future ECGA runs on similar problems. While one may extract more complex statistics, such as the probabilities of larger groups of variables taking the same linkage group, the computational complexity of such an approach would likely outweigh the benefits from such a bias.

To incorporate bias based on prior ECGA models represented by probabilities that pairs from each category take the same linkage group, there are two interesting possibilities. First of all, one may take the approach similar to that in hBOA and use a Bayesian metric to score models instead of the minimum description length metric used in the original ECGA. This would enable the bias be included via prior probabilities of model structures, in a similar way as it was done in hBOA. Another, a somewhat less rigorous, approach is to simply provide a penalty or reinforcement for putting a pair of variables into the same linkage group that depends on the probabilities of these variables being in the same linkage group in the past (with respect to the category for the variable pair defined by γ). There are certainly many approaches one may envision to doing this. The framework proposed in this paper can be used in this context as the main source of inspiration.

6.2 Dependency-Structure-Matrix Genetic Algorithm

The dependency-structure-matrix genetic algorithm (DSM-GA) (Yu et al., 2003, 2009) uses a dependency structure matrix to determine disjoint groups of interacting variables. The decomposition itself has similar semantics as the clustering of variables expressed by MPM in ECGA, but the method to discover such a clustering is significantly different. Specifically, DSM-GA first computes the interaction strength for each pair of variables; this can be done for example by computing the mutual information or another dependency measure. Then, various clusterings are searched using a variant of an evolution strategy (Rechenberg, 1973; Schwefel, 1974) to find a model that covers the most important dependencies and, at the same time, that is as simple as possible. The tradeoff between the coverage of dependencies and model complexity is dealt with using a variant of the minimum description length metric for evaluating the models.

From the perspective of learning from experience using distance-based bias, most of the discussion of ECGA applies also in this case and similar statistics can be extracted as was done for ECGA. Besides estimating probabilities of pairs of variables in each category being in the same linkage group, one may also focus on the dependency measures themselves (in fact, this could also be done in ECGA, especially if pairwise dependency measures are used to construct the models as proposed by Duque et al. (2008)). Furthermore, similar to ECGA, the resulting statistics can be used to bias model building either by using Bayesian metrics to evaluate models instead of the conventional, minimum description length metric, or by introducing penalties and reinforcements for dependencies based on their likelihood or strength in previous runs.

6.3 Linkage-tree Genetic Algorithm

In the linkage-tree genetic algorithm (Thierens, 2010a,b), the model is a linkage tree that is created by a sequence of merge operations using a hierarchical clustering algorithm. The main driving force of the clustering algorithm is the distance metric, which defines a distance for any two clusters of variables (this distance metric should not be confused with the distance metric used in this paper; in LTGA, the distance metric is based on the degree of interaction between variables). Distances are expected to be smaller for variables or groups of variables that interact more strongly in the statistical sense, and they are typically measured using the variation of information. However, other approaches were proposed and analyzed (Bosman and Thierens, 2012; Pelikan et al., 2011), such as the distance metric used in this paper.

There are several sources of information that one can obtain from prior runs of LTGA. First of all, one may collect distances computed by the clustering algorithm between specific pairs of variables that fall into the same category q according to γ function. These distances could be used in full or in part in future LTGA runs (it would probably be beneficial to only use these in part by weighing information from prior models and that from the current population). Another approach would be to consider linkage trees themselves, especially in terms of how early variables that fall into different categories were merged during the clustering or in terms of the percentages of linkage groups in which variables in different categories are located simultaneously. This approach would be more difficult to use, because it is not as straightforward to introduce bias into clustering based on prior clusterings directly. One feasible approach would be to use a system of penalties and reinforcements that will affect decisions taken by the clustering algorithm to a limited degree. Of course, there are other approaches one may easily envision and more research is necessary to find the best way of introducing the bias in LTGA. Nonetheless, all these approaches can derive inspiration from using a γ function to map pairs of variables into separate categories (such as categories based on a distance metric between the variables), and then using the obtained information to bias model building in future runs on similar problems.

7 Summary and Conclusions

This paper introduced a practical approach to incorporating bias in model-directed optimization algorithms (MDOs) based on models built in previous MDO runs on problems of similar type. The approach was demonstrated on the hierarchical Bayesian optimization algorithm (hBOA) and additively decomposable functions, although the framework can be applied also to other MDOs and other problem types. For example, it should be straightforward to adapt this framework to the extended compact genetic algorithm (Harik, 1999) and the linkage-tree genetic algorithm (Thierens, 2010a), or to apply a similar approach to facility location and scheduling problems.

The key idea of the proposed approach was to define a distance metric that corresponds to the likelihood or strength of dependencies between variables, and to use the statistics on dependencies at various distances in previous hBOA runs as the basis for introducing bias in future hBOA runs. The bias was introduced via prior probability distribution of Bayesian network structures in the Bayesian scoring metric. The models were thus learned using a combination of the selected population of candidate solutions and the prior knowledge extracted from previous hBOA models. The strength of the bias can be tuned with a user-defined parameter $\kappa > 0$.

The proposed approach was tested on four challenging additively decomposable functions: (1) NK landscapes with nearest-neighbor interactions, (2) minimum vertex cover, (3) two-dimension and three-dimensional Ising spin glasses, and (4) MAXSAT. The results on over 18,000 unique problem instances from the four problem classes provided ample empirical evidence that the proposed approach provides substantial speedups across a variety of settings and problem types. Furthermore, significant speedups were obtained on a great majority problem instances of each problem type. Best speedups were achieved for NK landscapes and minimum vertex cover; worst speedups were achieved for 3D spin glass and MAXSAT.

The paper argued that the proposed approach can be used even when prior runs were done on problems of smaller size. This represents one of the main advantages of the proposed approach in comparison with prior work in this area. Furthermore, it was shown that significant speedups can often be expected even when prior runs were done on problems of different type, although in that case, speedups may become much less significant. It was shown that the proposed approach combines well with other efficiency enhancement techniques; this was demonstrated by combining the proposed approach to distance-based bias with sporadic model building. It was also argued that the proposed framework can be adapted to other model-directed optimizers, such as the extended compact genetic algorithm, the dependency-structure-matrix genetic algorithm, and the linkage-tree genetic algorithm. Finally, the framework was discussed in a more general setting where an arbitrary mapping can be used to categorize pairs of variables in place of the distance metric; this further expands the potential for using the discussed framework in new contexts.

The results reaffirm that one of the key advantages of MDOs is that MDOs provide practitioners with a rigorous framework for (1) incorporating prior knowledge and (2) automated learning from solving instances of similar type so that future problem instances can be solved with increased speed, accuracy, and reliability. MDOs such as EDAs thus not only allow practitioners to scalably solve problems with high levels of epistasis (variable interactions) and a large number of local optima, but they also allow effective inductive transfer (transfer learning) in optimization.

One of the key topics for future work is to adapt the proposed approach to other model-directed optimization techniques; some ideas along these lines were already discussed in this paper. The proposed approach should also be evaluated on other problem classes, including real-world problems. Most importantly, the approach should be modified to introduce bias on problems that cannot be formulated using an additive decomposition in a straightforward manner, or such a decomposition does not provide an adequate distance metric. Doing this should be feasible for a number of important problem classes, including facility location problems and classes of problems from physics, such as Sherrington-Kirkpatrick spin glasses and atomic cluster optimization, because for all these problem classes it is relatively straightforward to define a distance metric between problem variables. It would also be interesting to explore the ways in which only a subset of models would be selected for creating the bias or the bias would depend on the stage of the optimization run (e.g. different bias would be incorporated at the beginning of the run than in the middle of the run). Finally, it would be useful to provide automated techniques or theory for tuning the strength of the bias to maximize its impact.

Acknowledgments

We would like to thank Kumara Sastry and David E. Goldberg for inspiring some of the ideas presented in this paper. We would also like to thank anonymous reviewers for pointing out aspects that deserved more attention. This project was sponsored by the National Science Foundation under grants ECS-0547013 and IIS-1115352, and by the University of Missouri in St. Louis through the High Performance Computing Collaboratory sponsored by Information Technology Services. Most experiments were performed on the Beowulf cluster maintained by ITS at the University of Missouri in St. Louis and the HPC resources at the University of Missouri Bioinformatics Consortium. hBOA was developed at the Illinois Genetics Algorithm

Laboratory at the University of Illinois at Urbana-Champaign. 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.

References

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

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

Bosman, P. and Thierens, D. (2012). On measures to build linkage trees in LTGA. *Parallel Problem Solving from Nature*.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28:41–75.

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

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

Drezner, Z. and Salhi, S. (2002). Using hybrid metaheuristics for the one-way and two-way network design problem. *Naval Research Logistics*, 49(5):449–463.

Duque, T., Goldberg, D., and Sastry, K. (2008). Enhancing the efficiency of the ECGA. *Parallel Problem Solving from Nature*, pages 165–174.

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

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

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

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

Hauschild, M., Pelikan, M., Sastry, K., and Goldberg, D. E. (2012). Using previous models to bias structural learning in the hierarchical boa. *Evolutionary Computation*, 20(1):135–160.

Hauschild, M. W. and Pelikan, M. (2008). Enhancing efficiency of hierarchical BOA via distance-based model restrictions. *Parallel Problem Solving from Nature*, pages 417–427.

Hauschild, M. W. and Pelikan, M. (2009). Intelligent bias of network structures in the hierarchical BOA. *Genetic and Evolutionary Computation Conference (GECCO-2009)*, pages 413–420.

Hauschild, M. W. and Pelikan, M. (2010). Network crossover performance on NK landscapes and deceptive problems. *Genetic and Evolutionary Computation Conference (GECCO-2010)*, pages 713–720.

Hauschild, M. W. and Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128.

Hauschild, M. W., Pelikan, M., Sastry, K., and Goldberg, D. E. (2011). Using previous models to bias structural learning in the hierarchical boa. *Evolutionary Computation*. In press. Also published as MEDAL Report No. 2008003.

Hauschild, M. W., Pelikan, M., Sastry, K., and Lima, C. F. (2009). Analyzing probabilistic models in hierarchical BOA. *IEEE Transactions on Evolutionary Computation*, 13(6):1199–1217.

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

Höns, R. (2005). *Estimation of Distribution Algorithms and Minimum Relative Entropy*. PhD thesis, University of Bonn, Germany.

Hoos, H. H. and Stützle, T. (2000). SATLIB: An online resource for research on sat. In Gent, I. P., v. Maaren, H., and Walsh, T., editors, *SAT 2000*, pages 283–292. IOS Press.

Kaedi, M. and Ghasem-Aghaee, N. (2011). Biasing Bayesian optimization algorithm using case based reasoning. *Knowledge-Based Systems*, 24(8):1245–1253.

Kauffman, S. (1989). Adaptation on rugged fitness landscapes. In Stein, D. L., editor, *Lecture Notes in the Sciences of Complexity*, pages 527–618. Addison Wesley.

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

Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E., editors (2006). *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer.

Lozano, J. A., Sagarna, R., and Larrañaga, P. (2001). Parallel estimation of Bayesian networks algorithms. In *Evolutionary Computation and Probabilistic Graphical Models, Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, pages 137–144.

Mendiburu, A., Santana, R., and Lozano, J. A. (2007). Introducing belief propagation in estimation of distribution algorithms: A parallel approach. Technical Report EHU-KAT-IK-11-07, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

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

Ocenasek, J. (2002). *Parallel Estimation of Distribution Algorithms*. PhD thesis, Faculty of Information Technology, Brno University of Technology, Brno.

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

Ochoa, A., Höns, R., Soto, M., and Mühlenbein, H. (2003). A maximum entropy approach to sampling in eda: The single connected case. *Progress in Pattern Recognition, Speech and Image Analysis, LNCS 2905*, pages 683–690.

Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

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

Pelikan, M. (2010). NK landscapes, problem difficulty, and hybrid evolutionary algorithms. *Genetic and Evolutionary Computation Conference (GECCO-2010)*, pages 665–672.

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

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

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

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

Pelikan, M. and Hauschild, M. (2012a). Distance-based bias in model-directed optimization of additively decomposable problems. pages 273–280.

Pelikan, M. and Hauschild, M. (2012b). Transfer learning, soft distance-based bias, and the hierarchical boa. Technical report.

Pelikan, M., Hauschild, M., and Thierens, D. (2011). Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. pages 1005–1012.

Pelikan, M., Kalapala, R., and Hartmann, A. K. (2007). Hybrid evolutionary algorithms on minimum vertex cover for random graphs. *Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 547–554.

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

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

Pelikan, M., Sastry, K., and Goldberg, D. E. (2008). Sporadic model building for efficiency enhancement of the hierarchical BOA. *Genetic Programming and Evolvable Machines*, 9(1):53–84.

Pratt, L. Y., Mostow, J., Kamm, C. A., and Kamm, A. A. (1991). Direct transfer of learned information among neural networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, page 584589.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.

Rothlauf, F. (2006). *Representations for Genetic and Evolutionary Algorithms*. Springer.

Santana, R., Mendiburu, A., and Lozano, J. A. (2012). Structural transfer using EDAs: An application to multi-marker tagging SNP selection. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2012)*, pages 3484–3491.

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

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

Schwarz, J. and Ocenasek, J. (1999). Experimental study: Hypergraph partitioning based on the simple and advanced algorithms BMDA and BOA. *Proceedings of the Int. Conf. on Soft Computing*, pages 124–130.

Schwarz, J. and Ocenasek, J. (2000a). A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning. In *Proceedings of the Fourth Joint Conf. on Knowledge-Based Software Engineering*, pages 51–58, Brno, Czech Republic. IO Press.

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

Schwefel, H.-P. (1974). Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Technical Report of the Working Group of Bionics and Evolution Techniques at the Institute for Measurement and Control Technology Re 215/3, Technical University of Berlin, Berlin.

Spin Glass Server (2004). Retrieved from http://www.informatik.uni-koeln.de/spinglass/. University of Köln, Germany.

Stonedahl, F., Rand, W., and Wilensky, U. (2008). Crossnet: a framework for crossover with network-based chromosomal representations. *Genetic and Evolutionary Computation Conference (GECCO-2008)*, pages 1057–1064.

Thierens, D. (2010a). The linkage tree genetic algorithm. *Parallel Problem Solving from Nature*, pages 264–273.

Thierens, D. (2010b). Linkage tree genetic algorithm: First results. In *Genetic and Evolutionary Computation Conference (GECCO-2010) Companion*, pages 1953–1958.

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

Yu, T.-L., Goldberg, D. E., and Chen, Y.-P. (2003). A genetic algorithm design inspired by organizational theory: A pilot study of a dependency structure matrix driven genetic algorithm. IlliGAL Report No. 2003007, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL.

Yu, T.-L., Goldberg, D. E., Sastry, K., Lima, C. F., and Pelikan, M. (2009). Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary Computation*, 17(4):595–626.

Yu, T.-L., Sastry, K., Goldberg, D. E., and Pelikan, M. (2007). Population sizing for entropy-based model building in estimation of distribution algorithms. *Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 601–608.