Missouri Estimation of Distribution Algorithms Laboratory

Sporadic Model Building for Efficiency Enhancement of the Hierarchical BOA

Martin Pelikan, Kumara Sastry, and David E. Goldberg

MEDAL Report No. 2007009

November 2007

Abstract

Efficiency enhancement techniques—such as parallelization and hybridization—are among the most important ingredients of practical applications of genetic and evolutionary algorithms and that is why this research area represents an important niche of evolutionary computation. This paper describes and analyzes *sporadic model building*, which can be used to enhance the efficiency of the hierarchical Bayesian optimization algorithm (hBOA) and other estimation of distribution algorithms (EDAs) that use complex multivariate probabilistic models. With sporadic model building, the structure of the probabilistic model is updated once in every few iterations (generations), whereas in the remaining iterations, only model parameters (conditional and marginal probabilities) are updated. Since the time complexity of updating model parameters is much lower than the time complexity of learning the model structure, sporadic model building decreases the overall time complexity of model building. The paper shows that for boundedly difficult nearly decomposable and hierarchical optimization problems, sporadic model building leads to a significant *model-building speedup*, which decreases the asymptotic time complexity of model building in hBOA by a factor of $\Theta(n^{0.26})$ to $\Theta(n^{0.5})$, where n is the problem size. On the other hand, sporadic model building also increases the number of evaluations until convergence; nonetheless, if model building is the bottleneck, the *evaluation slowdown* is insignificant compared to the gains in the asymptotic complexity of model building. The paper also presents a dimensional model to provide a heuristic for scaling the structure-building period, which is the only parameter of the proposed sporadic model-building approach. The paper then tests the proposed method and the rule for setting the structure-building period on the problem of finding ground states of 2D and 3D Ising spin glasses.

Keywords

Bayesian optimization algorithm, hierarchical BOA, estimation of distribution algorithms, efficiency enhancement, sporadic model building.

Note

Preprint of the paper accepted by the *Genetic Programming and Evolvable Hardware* journal published by Springer.

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/`

Sporadic Model Building for Efficiency Enhancement of the Hierarchical BOA

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory, 321 CCB
Dept. of Mathematics and Computer Science
University of Missouri in St. Louis
One University Blvd., St. Louis, MO 63121
pelikan@cs.umsl.edu

Kumara Sastry
Illinois Genetic Algorithms Laboratory, 117 TB
Dept. of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave., Urbana, IL 61801
kumara@kumarasastry.com

David E. Goldberg
Illinois Genetic Algorithms Laboratory, 117 TB
Dept. of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave., Urbana, IL 61801
deg@uiuc.edu

Abstract

Efficiency enhancement techniques—such as parallelization and hybridization—are among the most important ingredients of practical applications of genetic and evolutionary algorithms and that is why this research area represents an important niche of evolutionary computation. This paper describes and analyzes *sporadic model building*, which can be used to enhance the efficiency of the hierarchical Bayesian optimization algorithm (hBOA) and other estimation of distribution algorithms (EDAs) that use complex multivariate probabilistic models. With sporadic model building, the structure of the probabilistic model is updated once in every few iterations (generations), whereas in the remaining iterations, only model parameters (conditional and marginal probabilities) are updated. Since the time complexity of updating model parameters is much lower than the time complexity of learning the model structure, sporadic model building decreases the overall time complexity of model building. The paper shows that for boundedly difficult nearly decomposable and hierarchical optimization problems, sporadic model building leads to a significant *model-building speedup*, which decreases the asymptotic time complexity of model building in hBOA by a factor of $\Theta(n^{0.26})$ to $\Theta(n^{0.5})$, where n is the problem size. On the other hand, sporadic model building also increases the number of evaluations until convergence; nonetheless, if model building is the bottleneck, the *evaluation slowdown* is insignificant compared to the gains in the asymptotic complexity of model building. The paper also presents a dimensional model to provide a heuristic for scaling the structure-building period, which is the only parameter of the proposed sporadic model-building approach. The paper then tests the proposed method and the rule for setting the structure-building period on the problem of finding ground states of 2D and 3D Ising spin glasses.

Keywords: Bayesian optimization algorithm, hierarchical BOA, estimation of distribution algorithms, efficiency enhancement, sporadic model building.

1 Introduction

The hierarchical Bayesian optimization algorithm (hBOA) [65, 67, 62] replaces standard variation operators of genetic and evolutionary algorithms [40, 25, 75, 43] by building a Bayesian network for selected solutions and sampling the built network to generate new candidate solutions. Additionally, hBOA uses restricted tournament replacement [32] to effectively maintain useful diversity and preserve alternative partial solutions. It was theoretically and empirically shown that hBOA can solve nearly decomposable and hierarchical optimization problems in a quadratic number of evaluations or faster [62, 74].

However, even quadratic performance may be insufficient for problems with thousands of decision variables or high-order interactions. Consequently, efficiency enhancement techniques [27, 78, 81, 62] may have to be incorporated into hBOA to make this algorithm practical even for extremely large and complex problems. A number of efficiency enhancement techniques can be incorporated into hBOA and other genetic and evolutionary algorithms [29, 26, 83, 10, 3, 78, 85, 86, 6, 27, 81, 62, 47].

This paper describes and analyzes *sporadic model building*, which can significantly speed up model building in hBOA and other estimation of distribution algorithms (EDAs) that use complex multivariate probabilistic models, the learning of which may become a computational bottleneck. Specifically, with sporadic model building, hBOA updates the structure of the Bayesian network used to sample new solutions once in every few iterations (generations); in the remaining iterations, the structure from the previous iteration is used and only the parameters of the network (conditional and marginal probabilities) are updated based on the selected solutions. Since learning the structure is the most expensive component of model building [62], sporadic model building should lead to a significant model-building speedup.

Next, we put sporadic model building through extensive empirical analysis on the boundary of its design envelope by testing this technique on several problems that bound the class of nearly decomposable and hierarchical problems [27]. The results confirm a significant model-building speedup, which decreases the asymptotic complexity of model building on all test problems. Since most complex problems can be solved using techniques that exploit decomposition or hierarchical decomposition and many complex systems can be simplified using near decomposability and hierarchy [84, 59, 76, 4, 88, 87, 16, 44], the results should carry over to broad classes of real-world problems.

In the speedup analysis in the initial experiments, we always use an optimal value of the structure-building period (based on empirical results), which is the only parameter of sporadic model building. Of course, having to perform a set of experiments with various values of this parameter seems to go against the purpose of efficiency enhancement; that is why in the second part of the paper we propose a dimensional model, which can be used to *automatically* set the structure-building period for separable problems of bounded difficulty. The dimensional model is then verified with experiments on the problem of finding ground states of 2D and 3D Ising spin glasses with $\pm J$ couplings and periodic boundary conditions. Although Ising spin glasses can be efficiently solved using near decomposability and hierarchy [66, 62], they are not fully consistent with the assumptions of the theoretical model, and fully accurate models (even with full prior problem knowledge) are infeasible [50]. Even more importantly, Ising spin glasses are highly multimodal, their local optima are often at great distances from the global optima, they contain

high order interactions that grow in size with the number of variables, and they yield exponential time complexity for most standard optimization and simulation methods [33, 12, 62] although some difficult classes of spin glasses can be solved in polynomial time using analytical methods [22, 23]. That is why the experiments on Ising spin glasses should provide a solid empirical argument for the validity of the proposed method.

The paper is organized as follows. Section 2 describes hBOA and briefly reviews efficiency enhancement techniques for hBOA and other estimation of distribution algorithms. Section 3 presents sporadic model building and discusses its effects on the time complexity of hBOA. Section 4 presents experimental results on several common artificial decomposable and hierarchical problems. Section 5 presents a dimensional model that provides a bound on the structure-building period and estimates the expected optimal speedup. Section 6 verifies the proposed method and the theoretical model on the problem of finding ground states of $\pm J$ Ising spin glasses with periodic boundary conditions. Finally, Section 7 summarizes and concludes the paper.

2 Hierarchical Bayesian Optimization Algorithm (hBOA)

Estimation of distribution algorithms (EDAs) evolve a population of candidate solutions to the given problem by building and sampling a probabilistic model of promising solutions [5, 53, 70, 45]. The hierarchical Bayesian optimization algorithm (hBOA) [65, 67, 62, 73] is an EDA that uses Bayesian networks to represent the probabilistic model and incorporates restricted tournament replacement for useful-diversity maintenance. This section outlines hBOA and reviews efficiency enhancement techniques for hBOA and other EDAs.

2.1 Basic hBOA Procedure

hBOA evolves a population of candidate solutions represented by fixed-length strings over a finite alphabet (for example, binary strings). The initial population is generated at random according to a uniform distribution over the set of all potential solutions. Each iteration (generation) starts by selecting promising solutions from the current population using any standard selection method of genetic and evolutionary algorithms. For example, k-ary tournament selection can be used, which selects one solution at a time by first choosing a random subset of k candidate solutions from the current population and then selecting the best solution out of this subset; random tournaments are repeated until there are sufficiently many solutions in the selected population.

After selecting the promising solutions, hBOA builds a Bayesian network with local structures in the form of decision trees [42, 60, 11] as a model for these solutions. New solutions are generated by sampling the built network. The new solutions are then incorporated into the original population using restricted tournament replacement (RTR) [32], which ensures effective diversity maintenance. RTR with window size $w > 1$ incorporates each new candidate solution X into the original population using the following three steps:

1. Randomly select a subset W of w candidate solutions from the original population.
2. Let Y be a solution from W that is most similar to X (based on genotypic distance).
3. Replace Y with X if X is better; otherwise, discard X.

A robust rule of thumb is to set $w = \min\{n, N/20\}$, where n is the number of decision variables in the problem and N is the population size [62].

```
Hierarchical BOA (hBOA)
  t := 0;
  generate initial population P(0);
  while (not done) {
    select population of promising solutions S(t);
    build Bayesian network B(t) with decision trees for S(t);
    sample B(t) to generate offspring O(t);
    incorporate O(t) into P(t) using RTR yielding P(t+1);
    t := t+1;
  };
```

Figure 1: Pseudocode of the hierarchical Bayesian optimization algorithm.

The next iteration is executed unless some predefined termination criteria are met. For example, the run can be terminated when the maximum number of generations is reached or the entire population consists of copies of the same candidate solution. The pseudocode of hBOA is shown in Figure 1.

hBOA was shown to solve nearly decomposable and hierarchical problems in a quadratic or sub-quadratic number of function evaluations [61, 62]. Since most complex problems can be solved using techniques that exploit decomposition or hierarchical decomposition and many complex systems can be simplified using near decomposability and hierarchy [84, 59, 76, 4, 88, 87, 16, 44], hBOA should provide a robust and scalable solution to broad classes of challenging real-world problems.

In the remainder of this section, we discuss hBOA variation, which consists of learning and sampling Bayesian networks with decision trees. Additionally, we discuss the time complexity of the different components of hBOA variation and present a brief overview of various efficiency enhancement techniques that can be used to speed up hBOA on difficult problems.

2.2 Learning and Sampling Bayesian Networks with Decision Trees

Bayesian networks (BNs) [60, 42] combine graph theory, probability theory and statistics to provide a flexible and practical tool for probabilistic modeling and inference. hBOA uses Bayesian networks to model promising solutions in the selected population and to sample new candidate solutions. A Bayesian network consists of two components:

(1) Structure, which is defined by an acyclic directed graph with one node per variable and the edges corresponding to conditional dependencies between the variables, and

(2) parameters, which consist of the conditional probabilities of each variable given the variables that this variable depends on.

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

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

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

X_2	X_3	X_4	$p(X_1\|X_2, X_3, X_4)$
0	0	0	0.75
0	0	1	0.25
0	0	0	0.25
0	0	1	0.25
1	1	0	0.20
1	1	1	0.20
1	1	0	0.20
1	1	1	0.20

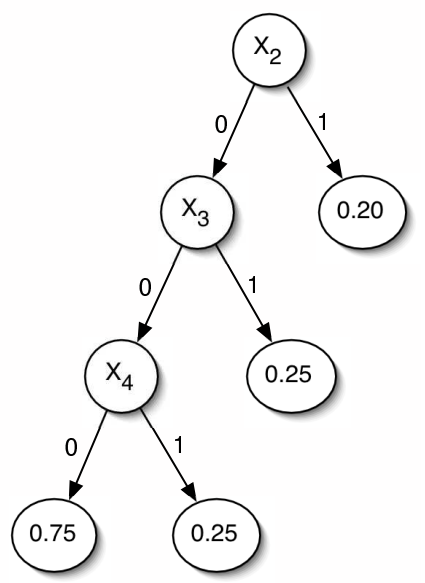


Figure 2: A conditional probability table for $p(X_1|X_2, X_3, X_4)$ and a decision tree that reduces the number of parameters from 8 to 4.

To make the representation of BN parameters more efficient, *local structures* can be used to store the conditional probabilities [20, 11]. hBOA uses decision trees for this purpose. In BNs with decision trees [20, 11], the conditional probabilities $p(X_i|\Pi_i)$ for each variable X_i are encoded by a special 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 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 some other variables. Each leaf node of T_i then stores the probabilities of X_i given the constraint defined by the traversal of T_i that ends in this leaf.

For the binary alphabet, there are two children of any internal node (one child corresponds to a 0, wheres the other one corresponds to a 1) and only one probability must be stored in each leaf because $p(X_i = 0|\Pi_i = \pi_i) + p(X_i = 1|\Pi_i = \pi_i) = 1$ for any instance π_i of Π_i.

To illustrate the use of decision trees in Bayesian networks, consider the example shown in Figure 2 [68], which considers a conditional probability table for X_1 where $\Pi_1 = \{X_2, X_3, X_4\}$ and all variables are binary. In this example, the full conditional probability table must contain 8 values, whereas the decision tree can store the same information with only 4 parameters.

The problem of learning Bayesian networks with decision trees can be split into two subproblems: (1) learn the structure and (2) learn the parameters.

To estimate the parameters of a Bayesian network with decision trees, hBOA uses the maximum likelihood estimate of the probabilities in the leaves of all decision trees. Consider a decision tree T_i for X_i and a leaf in this decision tree that specifies a condition C (based on the traversal). Then, the maximum likelihood estimate of $p(X_i = x_i|C)$ where x_i is a potential value of X_i, is given by

$$p(X_i = x_i|C) = \frac{m(X_i = x_i, C)}{m(C)}, \tag{2}$$

where $m(X_i = x_i, C)$ denotes the number of instances with $X_i = x_i$ that satisfy C, and $m(C)$ denotes the number of instances that satisfy C.

Before we describe how hBOA learns the BN structure, we first discuss how the quality of a specific BN structure can be evaluated with respect to the set of selected solutions. To measure quality of competing network structures, scoring metrics for standard Bayesian networks can be adapted to Bayesian networks with decision trees [11, 20]. We use the BDe metric for Bayesian networks with decision trees [11], which can be computed for a network B as

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

where $p(B)$ is the prior probability of B; 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 D 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 $m_i(i, l)$; and $m'_i(x_i, l)$ represents the prior knowledge about the value of $m_i(x_i, l)$. We use an uninformative prior that sets $m'_i(x_i, l) = 1$ for all i.

Bayesian metrics tend to be more sensitive to the noise in data and, in practice, they often lead to overly complex models [38, 20, 51, 64]. To bias the learning of Bayesian networks with decision trees toward simple models, we adjust the prior probability of each network to favor simpler models. This is done by first computing the description length of the parameters required by the network. One frequency in the data set of size N can be encoded using $\log_2 N$ bits; however, only half of the bits suffice to encode the frequencies with sufficient accuracy [21]. Therefore, to encode all parameters, $0.5(\sum_i |L_i|) \log_2 N$ bits are needed, where $\sum_i |L_i|$ is the total number of leaves in all decision trees. To favor simpler networks over the more complex ones, we decrease the prior probability of each network exponentially fast with the description length of this network's parameters [20]:

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

where c is a normalization constant required for the prior probabilities of all possible network structures to sum to one. The normalization constant does not affect the result because we are interested in only relative quality of networks and not the absolute value of their marginal likelihood.

To learn the structure of a Bayesian network with decision trees, a simple greedy algorithm is used [38, 11]. The greedy algorithm starts with an empty network, which is 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 [11], [20], and [62].

The sampling of a Bayesian network with decision trees can be done using probabilistic logic sampling [39], which proceeds in two steps. The first step computes an ancestral ordering of the nodes, where each node is preceded by its parents. In the second step, the values of all variables of a new candidate solution are generated according to the computed ordering. Since the algorithm generates the variables according to the ancestral ordering, when the algorithm attempts to generate the value of each variable, the parents of the variable must have already been generated.

2.3 Time Complexity of Learning and Sampling Bayesian Networks

Assuming that the problem is decomposable into subproblems of order k, the time complexity of learning the structure of a BN with decision trees can be bounded by $\Theta(kn^2N)$ [62]. Under the same assumptions, the conditional probabilities can be computed in $\Theta(knN)$ time steps. Thus, the asymptotic time complexity of learning the parameters is much lower than the asymptotic

complexity of learning the network structure. In agreement with the last statement, in actual runs of hBOA, the time complexities of learning the network structure and learning the parameters are often different by orders of magnitude.

Assuming an order-k decomposable problem, sampling one candidate solution from a given BN can be done in $\Theta(kn)$ steps. Consequently, the asymptotic complexity of sampling a population of N candidate solutions can be bounded by $\Theta(knN)$.

Therefore, the asymptotic time complexity of building the network structure dominates the overall complexity of hBOA variation, which consists of building and sampling a BN. A similar situation can be observed in other EDAs that use complex multivariate probabilistic models, for example, in the extended compact genetic algorithm (ECGA) [30, 79].

2.4 Efficiency Enhancement Techniques for hBOA and other EDAs

Solving difficult nearly decomposable and hierarchical problems in a quadratic or subquadratic number of evaluations yields many intractable problems tractable [27, 62]; nonetheless, for problems with thousands of variables or an extremely expensive objective function, even quadratic or subquadratic performance may not be enough. That is why a number of efficiency enhancement techniques have been proposed for hBOA and other evolutionary algorithms [29, 26, 83, 10, 3, 78, 85, 86, 6, 27, 81, 62, 47].

In hBOA, there are two potential computational bottlenecks, depending on the properties of the solved problem:

1. Model building, which becomes a computational bottleneck either when the problem contains a large number of variables (thousands or more), or when the problem exhibits large-order dependencies even on the lowest level of problem decomposition. Model building dominates the overall complexity of hBOA especially when fitness evaluation is simple; these results were confirmed with the time complexity analysis of the different components of hBOA in [56]; for a number of other EDAs, analogical analysis was performed in [7].

2. Fitness evaluation, which becomes a computational bottleneck when the objective function is computationally expensive and each fitness evaluation takes long time (tens of seconds or more); for example, the fitness evaluation may involve a complex finite element analysis or a simulation of a stochastic system.

Efficiency enhancement techniques for hBOA address the above bottlenecks and can be split into several categories [62]:

1. Parallelization.

2. Hybridization.

3. Time continuation.

4. Fitness evaluation relaxation.

5. Prior knowledge utilization.

6. Incremental and sporadic model building.

7. Learning from experience.

Past work on efficiency enhancement of EDAs focused on parallelization [57, 58], hybridization [52, 61, 66, 81, 46], fitness evaluation relaxation [80, 81, 72], prior knowledge utilization [83, 6], and incremental model building [15, 7]. For a general overview of efficiency enhancement techniques for EDAs, please see [82].

In this paper, we focus on sporadic model building (SMB), which can be used to speed up model building by rebuilding the structure only once in every few iterations. Since learning the structure of BNs has been shown to be the most time consuming component of hBOA variation (see Section 2.3), by rebuilding the structure only once in every few iterations, the model-building procedure of hBOA should take considerably less time. While other efficiency enhancement techniques can be used to speed up model building in hBOA as well, SMB has several advantages over other efficiency enhancement techniques:

- Unlike parallelization, SMB does not require a parallel computer and leads to high speedups even with a single processor; this will be confirmed by experimental results in sections 4 and 6.
- Unlike prior knowledge utilization, SMB does not require any prior problem-specific knowledge.
- Unlike incremental model building, SMB can be applied when model structure cannot be learned incrementally in a straightforward manner, as is the case in hBOA because of using local structures to represent local dependencies.

SMB can be combined with other efficiency enhancement techniques, often yielding multiplicative speedups. For example, if a parallel computer with 10 processors is used to parallelize model building with the speedup of about 10 [58] and additionally we use SMB with the speedup of 7, the overall speedup of model building becomes $7 \times 10 = 70$.

3 Sporadic Model Building (SMB)

The above section indicated that building the network structure is the most expensive part of hBOA variation and a similar observation can be made for other EDAs with complex probabilistic models. Consequently, for large problems, building the network structure may become a bottleneck. To speed up the structure-building procedure, one can exploit the fact that although the model structure does not remain the same throughout the run, the structures in consequent iterations of hBOA are often similar because consequent populations have similar structure with respect to the conditional dependencies and independencies. *Sporadic model building* (SMB) exploits this behavior by building the structure once in every few iterations, while in the remaining iterations only the parameters are updated for the structure used in the previous iteration.

Next, we describe a simple schedule that can be used to control SMB. The section then closes by discussing the effects of SMB on time complexity of hBOA.

3.1 Simple Periodic Schedule to Control SMB

One of the most important factors that influences the effectiveness of SMB is the *schedule* that determines when to build the structure and when to only update the parameters for the previous structure. This paper investigates a simple approach that uses a parameter t_{sb} called the structure-building period, which denotes the period with which the structure is updated. For example, $t_{sb} = 1$ denotes the scenario in which the structure is built in every iteration, whereas $t_{sb} = 2$ denotes the

scenario in which the structure is built in every other iteration. Of course, other types of schedules can be used; nonetheless, this paper indicates that even a simple schedule based on a fixed period yields asymptotic speedups of model building in hBOA and similar results can be expected for other similar EDAs.

3.2 Effects of SMB on BOA Time Complexity

There are two potential effects of using SMB in BOA: (1) speedup of model building and (2) slowdown of evaluation:

1. *Speedup of model building.* Since with SMB, the most expensive part of the model building procedure is run only in each t_{sb}th iteration, increasing t_{sb} should lead to a speedup of model building. Ideally, the speedup of building the model structure would be linearly proportional to t_{sb}; nonetheless, SMB may lead to an increase of the population size and the number of iterations and, as a result, with SMB each model building step may become more computationally expensive than without SMB. Furthermore, the actual speedup for a fixed problem size must be upper bounded by the expected number of iterations until convergence (the structure must be learned at least once).

   Since the number of iterations for decomposable problems of bounded difficulty grows as $O(n)$ [54, 90, 27], the speedup of model building is expected to be bounded by $O(n)$. Another way to estimate an upper bound of the expected speedup is to argue that the asymptotic complexities of building the structure and learning the parameters differ by a factor proportional to n and no matter whether the structure is built or not, the parameters must be updated in each iteration.

2. *Slowdown of evaluation.* SMB may lead to a decreased accuracy of the probabilistic models used to sample new candidate solutions. As a result, the population size N for building a sufficiently accurate model may increase with t_{sb}. Additionally, because of a less frequent adaptation of the model structure to the current population of promising solutions, SMB may slow down the convergence and the total number G of iterations may increase with t_{sb}. Consequently, an increase in the overall time spent in the evaluation of candidate solutions can be expected as a result of the increase of N and G, because the time spent in evaluation is linearly proportional to the number E of evaluations, where $E = N \times G$.

It can be expected that for each particular problem, there exists an optimal value of t_{sb}, which leads to the maximum overall speedup of hBOA on this problem. Nonetheless, to make our results independent of the time complexity of the evaluation function, which changes from problem to problem, we studied the above two effects in separation. The following section provides empirical results that approximate the two effects for several nearly decomposable and hierarchical problems. Next, a dimensional model is presented that provides a bound on t_{sb} for decomposable problems of bounded difficulty. The dimensional model is then verified with experiments on the problem of finding ground states of 2D and 3D $\pm J$ Ising spin glasses.

4 Initial Experiments on the Boundary of the Design Envelope

The last section argued that SMB should lead to significant speedup of the time complexity of model building. To support these results empirically, this section presents empirical analysis of SMB in hBOA. As test problems, we use common artificial separable and hierarchical problems that bound the class of nearly decomposable and hierarchical problems of bounded difficulty [27].

The behavior of SMB on the test problems should thus represent the behavior of SMB on difficult classes of nearly decomposable and hierarchical problems of bounded difficulty. Later, in Section 6 we present additional experiments for the 2D and 3D Ising spin glass problem, which is a challenging problem that cannot be factored into groups of bounded order and is known for its rough and complex landscape, which is difficult to explore effectively with common variation operators [66].

We start by describing the test problems used in the initial set of experiments. Then, we present and discuss the results.

4.1 Test Problems

This section describes test problems used in the empirical analysis of SMB in hBOA. All test problems are nearly decomposable or hierarchical and assume that candidate solutions are represented by n-bit binary strings. Three test problems were used in this set of experiments: (1) Dec-3, (2) Trap-5, and (3) hTrap. The test problems are described below:

1. *Dec-3: Concatenated 3-bit deceptive function.* In dec-3 [2, 24, 93, 13, 14], the input string is first partitioned into independent groups of 3 bits each. This partitioning is unknown to the algorithm and it does not change during the run. A 3-bit deceptive function is applied to each group of 3 bits and the contributions of all deceptive functions are added together to form the fitness. Each 3-bit deceptive function is defined as follows:

$$dec(u) = \begin{cases} 1 & \text{if } u = 3 \\ 0 & \text{if } u = 2 \\ 0.8 & \text{if } u = 1 \\ 0.9 & \text{if } u = 0 \end{cases}, \tag{5}$$

   where u is the number of 1s in the input string of 3 bits. The task is to maximize the function. An n-bit dec-3 function has one global optimum in the string of all 1s and $(2^{n/3} - 1)$ other local optima. To solve dec-3, it is necessary to consider interactions among the positions in each partition because when each bit is considered independently, the optimization is misled away from the optimum [89, 9, 69].

2. *Trap-5: Concatenated 5-bit trap.* Trap-5 is defined analogically to dec-3, but instead of 3-bit groups, 5-bit groups are considered. The contribution of each group of 5 bits is computed as

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

   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. Traps of order 5 also necessitate that all bits in each group are treated together, because statistics of lower order are misleading.

3. *hTrap: Hierarchical Trap.* Dec-3 and trap-5 problems can be decomposed into independent subproblems of a fixed order; in other words, they both are separable problems of bounded order. Nonetheless, hBOA can also solve problems that *cannot* be decomposed into subproblems of bounded order on a single level, but that must be solved hierarchically by building the solution from low-order building blocks over a number of levels of difficulty.

   Hierarchical traps (hTraps) [62] constitute a class hierarchical problems that cannot be efficiently solved using a single-level decomposition. Hierarchical traps are created by combining trap

functions of order 3 over multiple levels of difficulty. For hTraps, the string length should be an integer power of 3, that is, $n = 3^l$. On the lowest level, groups of 3 bits contribute to the overall fitness using generalized 3-bit traps defined as follows:

$$trap_3(u) = \begin{cases} f_{high} & \text{if } u = 3 \\ f_{low} - u\frac{f_{low}}{2} & \text{otherwise} \end{cases}, \quad (7)$$

where $f_{high} = 1$ and $f_{low} = 1 + 0.1/l$.

Each group of 3 bits corresponding to one of the traps is then mapped to a single symbol on the next level; a 000 is mapped to a 0, a 111 is mapped to a 1, and everything else is mapped to the null symbol '-'. The bits on the next level again contribute to the overall fitness using 3-bit traps defined above, and the groups are mapped to an even higher level. This continues until the top level is evaluated, which contains 3 bits total. However, on the top level, a trap with $f_{high} = 1$ and $f_{low} = 0.9$ is applied. Any group of bits containing the null symbol does not contribute to the overall fitness. To make the overall contribution of each level of the same magnitude, the contributions of traps on ith level from the bottom are multiplied by 3^i.

The task is to maximize the function. hTraps have many local optima, but only one global optimum in the string of all 1s [63]. Nonetheless, any single-level decomposition into subproblems of bounded order will lead away from the global optimum for sufficiently large problems. That is why hTraps necessitate an optimizer that can build solutions hierarchically by juxtaposing good partial solutions over multiple levels of difficulty until the global optimum is found [92, 63, 65, 62].

For more details on hierarchical traps and solving hierarchical optimization problems, please see [61]. Other hierarchical problems can be found in [92].

4.2 Experimental Methodology

To analyze scalability of SMB, for each problem, we performed experiments for a range of problem sizes (n = 30 to 210 with step 15 for dec-3 and trap-5; n = 9, 27, 81, and 243 for hTrap). For each problem and problem size, hBOA with SMB was tested for t_{sb} from 1 to 20 with step 1. For each problem, problem size, and value of t_{sb}, a minimum population size required to find the global optimum in 10 out of 10 independent runs was determined using the bisection method [78, 62]. To obtain more accurate estimates of the expected performance, for each setting, 10 bisection runs were performed. Therefore, for each problem, problem size, and value of t_{sb}, 100 successful independent runs were performed.

Each run was terminated either when the global optimum was found or when hBOA completed a large number of generations and it was unlikely that the algorithm would find the optimum in a reasonable time. The maximum number of iterations was n, which is a conservative upper bound determined based on preliminary experiments and convergence theory [54, 27, 74].

Binary tournament selection was used in all experiments and the window size for RTR was set as $w = \min\{n, N/20\}$ where n is the problem size and N is the population size. BDe metric with a penalty for complex networks as described in section 2.2 was used in all experiments to measure quality of Bayesian networks with decision trees.

4.3 Results

The experimental results were processed to obtain the following quantities:

1. *Speedup of structure building for each problem, problem size, and structure-building period.* To compute the structure-building speedup, for each setting, we first determine the average number of times the model structure was learned until the global optimum was found and the average population size (determined by bisection). Ideally, the structure-building speedup would then be equal to the ratio of the number of times the structure was learned without SMB ($t_{sb} = 1$) and the number of times the structure was learned with SMB ($t_{sb} > 1$). However, as shown in Section 2.3, the time complexity of the structure-building procedure increases linearly with the population size and since SMB may lead to larger population sizes, we need to multiply the aforementioned ratio by the ratio of the population size without SMB and the population size with SMB. The product of the two ratios represents the expected structure-building speedup. The larger the structure-building speedup, the better. For example, the speedup of 2 indicates that SMB made the structure learning in hBOA twice as fast.

2. *Optimal speedup of structure building for each problem and problem size.* For each problem and problem size, an optimal speedup is determined empirically by using the value of t_{sb} that maximizes the speedup of structure building.

3. *Evaluation slowdown for optimal speedup of structure building.* For each problem and problem size, the evaluation slowdown is determined for the best value of t_{sb}, which corresponds to the optimal speedup. The evaluation slowdown is defined as the factor by which the number of evaluations increases compared to the case with no SMB (that is, $t_{sb} = 1$). The smaller the evaluation slowdown, the better.

4. *Scalability.* Scalability with various values of t_{sb} is evaluated to investigate the effects of SMB on the scalability of hBOA on decomposable problems. Specifically, scalability with $t_{sb} = 1$ and $t_{sb} = 10$ is evaluated and compared.

Figure 3 shows the structure-building speedup and the evaluation slowdown for dec-3, trap-5, and hTrap.

Figure 4 shows the optimal speedup of structure building and the corresponding evaluation slowdown for dec-3, trap-5, and hTrap.

Figure 5 shows the growth of the number of evaluations with problem size on dec-3, trap-5 and hTrap for $t_{sb} = 1$ (without SMB) and $t_{sb} = 10$ (with SMB where the structure is built in every 10th iteration).

4.4 Discussion of Results

The results presented in Figure 3 indicate that for all test problems, the structure-building speedup grows with the structure-building period t_{sb} until it reaches a local maximum at the optimal value of t_{sb} that maximizes the speedup; the rate of speedup growth increases with problem size.

The results presented in Figure 4 indicate that the evaluation slowdown is much less significant than the structure-building speedup. For separable problems (dec-3 and trap-5), the evaluation slowdown corresponding to the optimal structure-building speedup decreases with problem size, whereas for the hierarchical problem hTrap, the evaluation slowdown slowly increases with problem size. We believe that the reason for better results on separable problems is that in separable problems, the same model can be used during the entire run, whereas in hierarchical problems, the model must be learned at least once for each level of decomposition.

Additionally, these results indicate that the optimum speedup grows with problem size and its polynomial approximation grows as $\Theta(n^{0.26})$ to $\Theta(n^{0.65})$; that means that the maximum speedup

of structure building is expected to grow with problem size and lead to a significant decrease of asymptotic complexity of model building in hBOA.

The results presented in Figure 5 indicate that SMB does not negatively affect the polynomial growth of the number of evaluations until convergence with problem size; specifically, they show that with a fixed structure-building period $t_{sb} = 10$, the number of evaluations appears to grow with a polynomial of the same order as without SMB or even better.

Therefore, the results lead to four important observations:

1. SMB leads to a significant decrease of the asymptotic time complexity of model building given an appropriate period t_{sb} for rebuilding the structure.
2. The optimal speedup obtained with SMB grows with problem size.
3. The factor by which SMB increases the number of evaluations is insignificant compared to the speedup of model building.
4. SMB with a constant structure-building period does not lead to an increase of the asymptotic complexity of hBOA with respect to the number of evaluations until convergence.

The following section presents a dimensional model that provides an upper bound on the structure-building period t_{sb} and suggests that for separable problems of bounded difficulty the structure-building period should grow proportionally to the square root of the problem size, $t_{sb} \propto \sqrt{n}$.

5 Dimensional Model for Separable Problems of Bounded Order

The last section indicated that SMB leads to a significant speedup of model building. The only parameter required for using SMB is the structure-building period t_{sb}, which denotes the interval with which the structure is rebuilt. Since the structure-building speedup grows with problem size and is upper bounded by t_{sb}, to achieve optimal performance, the value of t_{sb} should also grow with problem size. This section provides a dimensional model that estimates the growth of t_{sb} and the achieved speedup, assuming that the problem can be decomposed into independent subproblems of bounded order. Nonetheless, the following section will show that the derived rule for scaling t_{sb} with problem size works well even on problems that violate these assumptions because they cannot be factored into subproblems of bounded order.

5.1 Assumptions

The basic idea of the dimensional model for t_{sb} is to assume that most of the subproblems have been captured by the current structure correctly, and to compute the maximum number of iterations without structural updates to ensure that the model is rebuilt before the incorrectly modeled subproblems converge to non-optimal values.

More specifically, given an additively separable problem with m building blocks, each of order k, we assume that the model builder accurately discovers $(m-1)$ subproblems, and fails to model one subproblem. We further assume that the variables in the incorrectly modeled subproblem are modeled as independent of each other as shown in Figure 6. To provide a bounding (worst-case) analysis, we assume that building an accurate model is essential for successfully converging to the global optima; in other words, the decision variables in the incorrectly modeled subproblems converge to non-optimal values. Consequently, for the incorrectly modeled subproblem, inferior

partial solutions get increased market share every time the model is sampled. For example, in fully deceptive trap functions of order $k = 4$, the best partial solution of each subproblem is 1111 but the average fitness of a schema ***0 is higher than that of a schema ***1 [24, 14]; consequently, if the model considers each string position independently, ***0 gets increased market share over ***1 although 1111 is the best partial solution.

Additionally, we assume that the variances of the fitness contributions of all subproblems are approximately equal and that the signal for discriminating the two best partial solutions is also similar for different subproblems. Finally, binary tournament selection is assumed.

5.2 Gambler's Ruin Model for Population Sizing

We model the growth of inferior partial solutions using the gambler's ruin model used elsewhere for analyzing population sizing of selectorecombinative genetic algorithms [31]. However, unlike the previous study, we estimate the time required for the sub-optimal partial solutions in the incorrectly modeled subproblem to take over the entire population. This time to reach the absorbing state provides an upper bound on the structure-building period. To apply the Gambler's ruin model, we assume that there is no niching and, therefore, the model is directly applicable to BOA but for hBOA the bound provided by the model may be more restrictive than necessary because niching enables hBOA to preserve some inferior solutions. Nonetheless, section 6 shows that the bound provided by the theoretical model yields good results even with hBOA.

In the gambler's ruin problem, the gambler starts with some initial capital x_o, and competes with an opponent with an initial capital $(N - x_o)$. The gambler can reach one of the two absorbing states, one in which he loses all the money and the other in which the opponent loses all the money. At each step of the game, the gambler has a probability p of increasing his capital by one unit and a probability $(1 - p)$ of losing one unit.

Harik et al. [31] drew an analogy between the gambler's ruin problem and the growth in the proportion (market share) of the best partial solution in competition with the second-best partial solution in the same problem partition. Analogically to their model, in this study, we consider the decrease in the market share of non-optimal single-bit partial solutions in the incorrectly modeled partition. We denote the non-optimal partial solution H_1 and its competitor by H_2. For example, for traps of order $k = 4$, $H_1 = ***0$ and $H_2 = ***1$. Since the solutions are initialized according to the uniform distribution and the competing partial solutions are single bits, H_1 (the gambler) starts with the expected initial market share $x_o = N/2$, and competes with H_2 (the opponent) with the expected initial market share of $N - x_o = N/2$. The random walk is illustrated in Figure 7.

The probability of increasing the capital of H_1 by one unit is given by the probability that a candidate solution with H_1 is better than a candidate solution with H_2, called the decision-making probability [28]. Note that the decision making is stochastic, because the fitness of candidate solutions with the specific one-bit partial solution depends on the remaining bits. Assuming an additively separable problem of bounded order and an effective recombination operator (like that in hBOA), the fitness of H_1 and H_2 can be approximated by $\mathcal{N}(\overline{f}_{H_1}, \sigma^2_{H_1})$ and $\mathcal{N}(\overline{f}_{H_2}, \sigma^2_{H_2})$, respectively, where $\overline{f}_{H_1}$ and $\overline{f}_{H_2}$ denote the average fitness, and $\sigma^2_{H_1}$ and $\sigma^2_{H_2}$ denote the fitness variance of H_1 and H_2, respectively. The decision making between H_1 and H_2 is illustrated in Figure 8.

The probability of an individual with H_1 winning the competition over an individual with schema H_2 is equivalent to the probability that $f_{H_1} - f_{H_2} > 0$. Moreover, since f_{H_1} and f_{H_2} are normally distributed, $(f_{H_1} - f_{H_2})$ is also normally distributed with mean d, and variance $(\sigma^2_{H_1} + \sigma^2_{H_2})$:

$$f_{H_1} - f_{H_2} \sim \mathcal{N}\left(d, \sigma^2_{H_1} + \sigma^2_{H_2}\right). \tag{8}$$

The probability of H_1 winning a tournament over H_2 is given by

$$p = \Phi\left(\frac{d}{\sqrt{\sigma_{H_1}^2 + \sigma_{H_2}^2}}\right), \tag{9}$$

where $\Phi(z)$ is the cumulative density function of a unit normal variate z. Since the fitness function is additively decomposable, $\sigma_{H_1}^2$ (and, analogically, $\sigma_{H_2}^2$) is the sum of the variance of the fitness contribution of the $(m-1)$ correctly modeled subproblems and the variances of the $(k-1)$ bits in the incorrectly modeled subproblem [28]. That is,

$$\sigma_{H_1}^2 = \sigma_{H_2}^2 = \sigma_H^2 = (m-1)\sigma_{bb}^2 + (k-1)\sigma_A^2, \tag{10}$$

where σ_{bb}^2 is the fitness variance of a subproblem (we assume that all subproblems have approximately equal variance), σ_A^2 is the average variance of a single variable (string position) in the incorrectly modeled subproblem, m is the number of subproblems, and k is the subproblem size.

Given x_o and p, the total number of tournaments to reach an absorbing state, t_A, is given by [17]

$$t_A = \left(\frac{1}{1-2p}\right)\left[x_o - N\left(\frac{1-s^{x_o}}{1-s^N}\right)\right], \tag{11}$$

where

$$s = \frac{1-p}{p} = \frac{1-\Phi\left(\frac{d}{\sqrt{2}\sigma_H}\right)}{\Phi\left(\frac{d}{\sqrt{2}\sigma_H}\right)}. \tag{12}$$

Substituting for x_o and simplifying yields

$$t_A = -\frac{N}{2}\cdot\frac{1}{1-2p}\left[\frac{1-s^{N/2}}{1+s^{N/2}}\right]. \tag{13}$$

Approximating $\Phi(z)$ with the first two terms of power series [1] leads to

$$\Phi(z) \approx \frac{1}{2} + \frac{z}{\sqrt{2\pi}}. \tag{14}$$

Substituting the above power-series approximation of $\Phi(z)$ into Equation 12 and simplifying yields

$$s = \frac{1-\frac{d}{\sqrt{\pi}\sigma_H}}{1+\frac{d}{\sqrt{\pi}\sigma_H}}. \tag{15}$$

Since $\sigma_H^2 \propto m\sigma_{bb}^2$, for moderate-to-large-sized problems ($m \geq 4$), $s \approx 1$. Using this approximation in Equation 13, and simplifying we get

$$t_A \approx \frac{N\sqrt{\pi}}{2}\left(\frac{\sigma_H}{d}\right). \tag{16}$$

It should be noted that the time to reach the absorbing state t_A is in units of number of tournaments of binary tournament selection. Since in a generation, there are N tournaments, the expected number of generations to reach an absorbing state is given by

$$t_{A,g} = \frac{t_A}{N} = \frac{\sqrt{\pi}}{2}\left(\frac{\sigma_H}{d}\right). \tag{17}$$

Recall that $\sigma_H^2 = (m-1)\sigma_{bb}^2 + (k-1)\sigma_A^2$. Since $m \gg k$ (bounded difficulty), σ_H^2 is approximately equal to $(m-1)\sigma_{bb}^2$. Substituting this approximation for σ_H^2, we get

$$t_{A,g} = \frac{\sqrt{\pi}}{2}\left(\frac{\sigma_{bb}}{d}\right)\cdot\sqrt{m}. \tag{18}$$

5.3 Scaling Structure-Building Period

Based on the estimated time to the absorbing state, the model suggests that the structure-building period t_{sb} should grow at most as fast as the square root of the number of subproblems. Since for boundedly difficult decomposable problems, $n = \Theta(m)$ and we would like to maximize t_{sb} to ensure highest speedups, we get

$$t_{sb} \propto \sqrt{n}.$$

Moreover, if the increase in the number of function evaluations due to SMB is upper bounded by a constant, as was indicated by all experiments on separable problems of bounded difficulty in the previous section, the achieved structure-building speedup η is also expected to grow proportionally to $\sqrt{n}$:

$$\eta \propto \sqrt{n}. \tag{19}$$

6 Analysis on Ising Spin Glasses

The experimental results on artificial test problems with optimal structure-building period t_{sb} were tantalizing and the dimensional model provided us with a heuristic rule for scaling t_{sb} to ensure high speedups; specifically, it suggested that $t_{sb} \propto \sqrt{n}$. An important question is whether SMB and the rule for scaling t_{sb} will achieve high speedups even on problems that are not trivially decomposable into subproblems of bounded order and that represent a challenge for any optimization algorithm.

In this section, we test SMB and the heuristic rule for scaling t_{sb} on the problem of finding ground states of 2D and 3D $\pm J$ Ising spin glasses with periodic boundary conditions. Frustration (conflicting groups of constraints) and complex structure of Ising spin glasses make these problems challenging from a number of perspectives [50, 66, 12]: (1) The number of local optima grows extremely fast with problem size. (2) Local optima are often at great distances from any global optimum. (3) Between various local optima, there are barriers of low-quality solutions that are difficult to cross with local operators. (4) Solutions of similar, high quality have often little in common. (5) We cannot decompose spin glasses into subproblems of bounded order without sacrificing model accuracy; model complexity of perfect models grows exponentially fast. The results confirm that the proposed approach to setting t_{sb} significantly decreases the asymptotic and the actual time complexity of the model building in hBOA despite the inherent complexity of Ising spin glass problems.

First, we briefly introduce the class of 2D and 3D $\pm J$ Ising spin glasses with periodic boundary conditions and the problem of finding ground states of spin glass instances. Then, we present and discuss experimental results.

6.1 Ising spin glass

Ising spin glasses are prototypical models for disordered systems and have played a central role in statistical physics during the last three decades [8, 49, 18, 94]. Examples of experimental realizations of spin glasses are metals with magnetic impurities, e.g. gold with a small fraction of iron added. Spin glasses represent also a large class of challenging optimization problems where the task is to minimize energy of a given spin-glass instance [50, 55, 34, 35, 91, 66, 36, 19, 37, 41, 77, 71]. States with the lowest energy are called *ground states*.

A very 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 $\langle i, j \rangle$ has a real value $J_{i,j}$ associated with

it that defines the relationship between the two connected spins s_i and s_j. 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 element along each dimension.

For the classical Ising model, each spin s_i can be in one of two states: $s_i = +1$ or $s_i = -1$. Note that this simplification corresponds to highly anisotropic systems, which do indeed exist in some experimental situations. Nevertheless, the two-state Ising model comprises all basic effects also found in models with more degrees of freedom. A specific set of coupling constants define a spin glass instance. Each possible setting of all spins is called a spin configuration.

Given couplings $\{J_{i,j}\}$, and a configuration of spins $C = \{s_i\}$ $(i = 1, \ldots, n)$, the energy can be computed as

$$E(C) = \sum_{\langle i,j \rangle} s_i J_{i,j} s_j \ , \tag{20}$$

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

The usual task in statistical physics is to integrate a known function over all possible configurations of spins, assuming the Boltzmann distribution of spin configurations; that means, the probability of each configuration C is proportional to $\exp(-E(C)/T)$ where T is the temperature. From the physics point of view, it is also interesting to know the ground states (configurations associated with the minimum possible energy). Finding extremal energies then corresponds to sampling the Boltzmann distribution with temperature approaching 0 and thus the problem of finding ground states is simpler *a priori* than integration over a wide range of temperatures. However, most of the conventional methods based on sampling the above Boltzmann distribution fail to find the ground states because they get often trapped in a local minimum [12].

In order to obtain a quantitative understanding of the disorder in a spin glass system introduced by the random spin-spin couplings, one generally analyzes a large set of random spin-glass instances for a given distribution of the spin-spin couplings. For each spin glass instance, the optimization algorithm is applied and the results are analyzed to obtain a measure of computational complexity. Here we consider the $\pm J$ spin glass, where each spin-spin coupling constant is set randomly to either $+1$ or -1 with equal probability.

6.2 Description of Experiments

We consider 2D spin glasses of sizes 10×10 (100 bits) to 25×25 (625 bits) and 3D spin glasses of sizes $4 \times 4 \times 4$ (64 bits) to $n = 7 \times 7 \times 7$ (343 bits). For each problem size, 1000 random instances are generated and for each problem instance, hBOA is required to find the global optimum in 5 out of 5 independent runs. Like in the experiments presented earlier in this paper, the population size is determined for each problem instance using bisection [78]. Binary tournament selection is used in all experiments. The maximum number of iterations is set to the number of spins, n.

Unlike in the previous experiments, instead of testing a range of values of t_{sb} and then choosing the best value to determine the optimal speedup, in this set of experiments we used the dimensional model presented in the last section to devise an automated method for setting t_{sb}. More specifically, we first performed several experiments on one small 10×10 problem instance with various values of t_{sb}. The results indicated that $t_{sb} = 5$ seems to lead to a reasonable speedup without affecting the fitness evaluation much. Since the theoretical model suggested that $t_{sb} \propto \sqrt{n}$, we decided to scale t_{sb} as $t_{sb} = \sqrt{n}/2$ for both 2D and 3D spin glasses. A similar approach can be used for other problems by first testing small problem instances and then extrapolating under the assumption that $t_{sb} \propto \sqrt{n}$. In general, t_{sb} should be small enough to ensure that the model structure is learned at least several times before hBOA converges.

To improve performance, we incorporate a deterministic hill climber (DHC) into hBOA. DHC takes a candidate solution represented by an n-bit binary string on input. Then, it performs one-bit changes on the solution that lead to the maximum improvement of solution quality (maximum decrease in energy). DHC is terminated when no single-bit flip improves solution quality and the solution is thus locally optimal. DHC is used to improve every solution in the population before the evaluation is performed. The hybrid created by incorporating DHC into hBOA is referred to as hBOA+DHC.

6.3 Results

Figure 9 shows the structure-building speedup and the evaluation slowdown for SMB with $t_{sb} = \sqrt{n}/2$ on 2D and 3D Ising spin glasses with $\pm J$ couplings and periodic boundary conditions. The results indicate that the structure-building speedup grows with problem size and is significantly higher than the evaluation slowdown in both 2D and 3D.

The results also confirm that for difficult hierarchical problems, the evaluation slowdown with SMB becomes more significant than for separable problems, and that it grows with problem size. We believe that the reason for this behavior is that single-level decomposable problems can be solved using a single model, whereas hierarchical problems necessitate a special model for each level of decomposition. Since we use a fixed structure-building period during the entire optimization, performance of hBOA may sometimes suffer because of using an inadequate model from the lower level despite of having a sufficient signal to proceed with the next higher level.

To verify the effects of SMB on the *overall time complexity* of hBOA, including fitness evaluation and other hBOA components, we measured the overall CPU time per run with and without SMB, and computed the actual CPU-time speedup achieved with SMB on 2D and 3D spin glasses, which is shown in Figure 10. The results show that a significant CPU-time speedup is achieved and that the speedup grows with problem size, reaching a value of about 4.5 for 2D spin glasses of size $n = 625$ and about 4.1 for 3D spin glasses of size $n = 343$. Since structure-building is not the only component of hBOA, a specific structure-building speedup should lead to a somewhat smaller speedup of overall hBOA complexity; this observation agrees with the presented results.

7 Summary and Conclusions

This paper presented and analyzed an efficiency enhancement technique called *sporadic model building*, which can be used to speed up model building in BOA, hBOA and other similar EDAs. In sporadic model building, the structure of the probabilistic model is not updated in every iteration; instead, in some iterations only the parameters of the previous structure are updated with respect to the selected population of promising solutions. Building the model structure is the most computationally expensive part of hBOA variation and can become a computational bottleneck for large and complex problems. That is why sporadic model building represents an important efficiency enhancement technique for BOA, hBOA, and other similar EDAs. The paper proposed a simple schedule for sporadic model building, where the updates of the model structure are done with a fixed period t_{sb} called the structure-building period. For example, $t_{sb} = 1$ denotes the scenario where the structure is updated in every iteration, whereas $t_{sb} = 3$ denotes the scenario where the structure is updated in every third iteration. The paper analyzed the proposed approach to sporadic model building and presented a dimensional theoretical model that can be used to guide the choice of an appropriate value of t_{sb} and to estimate the expected speedup of model building.

The experimental results presented in this paper indicate three important observations regarding

the use of sporadic model building for solving nearly decomposable and hierarchical problems by hBOA:

1. Sporadic model building leads to significant improvements of asymptotic time complexity of hBOA model building and the optimal speedup of model building grows with problem size (in our experiments the optimal speedup grows as $\Theta(n^{0.26})$ to $\Theta(n^{0.65})$ depending on the problem). The speedup of hBOA variation is upper bounded by $O(n)$ because of parameter updates and sampling.

2. Sporadic model building leads to an increase in the number of evaluations until convergence but the factor by which sporadic model building increases the overall number of evaluations is insignificant compared to the model-building speedup, especially when model building is the computational bottleneck. For separable problems of bounded order, the evaluation slowdown decreases with problem size, whereas for difficult hierarchical problems, the evaluation slowdown grows slowly.

3. Sporadic model building does not lead to an increase of the asymptotic complexity of hBOA with respect to the number of evaluations until convergence and the number of evaluations with a fixed value of t_{sb} remains a low-order polynomial for nearly decomposable and hierarchical problems.

The dimensional model provides a heuristic for setting t_{sb} and suggests that for separable problems of bounded order, t_{sb} should be proportional to the square root of the problem size, $t_{sb} \propto \sqrt{n}$. The suggested rule for scaling t_{sb} with problem size is verified with experiments on the problem of finding ground states of 2D and 3D $\pm J$ spin glasses, which represent a challenging class of optimization problems with an extremely rough landscape and high-order interactions. The results show that even on Ising spin glasses, the proposed efficiency enhancement technique yields significant speedups in terms of both the model-building time complexity as well as the overall CPU time.

Clearly, sporadic model building is going to be most effective for problems that can be solved with a single model of bounded complexity, such as separable problems of bounded order and problems that can be accurately approximated with separable problems of bounded order. On the other hand, if the problem necessitates that the model structure changes over time as might be the case with hierarchical and dynamic problems, the effectiveness of sporadic model building can be expected to be limited by the length of intervals at which the model structure must be changed. However, if a problem can be efficiently solved using hBOA or other evolutionary algorithms, it can be expected that the model structure does not have to change dramatically between consequent iterations and that is why sporadic model building is expected to be effective for most such problems. This was confirmed with experiments on several test problems for which a single model of practical complexity is clearly insufficient, including the hierarchical traps and the NP-complete 3D spin glasses, and even for those problems the sporadic model building led to significant speedups that grew with problem size in terms of both the model-building complexity as well as the overall CPU time.

For many practical applications it is important to use all available efficiency enhancement techniques to make the computation feasible. The good news is that combining multiple efficiency enhancement techniques often multiplies the speedups obtained. That is why although the speedups we obtained with hBOA even on rather small problems are significant on their own right, in combination with other techniques these speedups can become even more significant. For example, consider a combination of parallelizing model building [57, 58, 48] and sporadic model building;

in this case, even if sporadic model building would yield a model-building speedup of only 2, sporadic model building would effectively double the number of available processors for the parallel implementation of model building in hBOA.

Acknowledgments

This work was supported by the National Science Foundation under NSF CAREER grant ECS-0547013 (at UMSL) and ITR grant DMR-03-25939 (at Materials Computation Center, UIUC), by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, and by the University of Missouri in St. Louis through the High Performance Computing Collaboratory sponsored by Information Technology Services, and the Research Award and Research Board programs.

The experiments presented in this work were done using the hBOA software developed by Martin Pelikan and David E. Goldberg at the University of Illinois at Urbana-Champaign. Most experiments were completed at the Beowulf cluster at the University of Missouri at St. Louis. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

References

[1] Abramowitz, M., & Stegun, I. A. (Eds.) (1972). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications. Ninth printing.

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

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

[4] Allen, T. F. H., & Starr, T. (Eds.) (1982). *Hierarchy: Perspectives for ecological complexity*. Chicago, IL: University of Chicago Press.

[5] 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.

[6] Baluja, S. (2002). Using a priori knowledge to create probabilistic models for optimization. *International Journal of Approximate Reasoning*, *31*(3), 193–220.

[7] Bengoetxea, E. (2002). *Inexact graph matching using estimation of distribution algorithms*. Doctoral dissertation, Department of Image and Signal Treatment, Ecole Nationale Superieure des Telecommunications (ENST), Paris, France.

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

[9] Bosman, P. A. N., & Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, *I*, 60–67.

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

[11] 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.

[12] 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.

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

[14] Deb, K., & Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, *10*, 385–408.

[15] Etxeberria, R., & Larrañaga, P. (1999). Global optimization using Bayesian networks. *Second Symposium on Artificial Intelligence (CIMAF-99)*, 332–339.

[16] Faihe, Y. (1999). *Hierarchical problem solving using reinforcement learning : Methodology and methods*. Doctoral dissertation, University of Neuchâtel, Neuchâtel, Switzerland.

[17] Feller, W. (1970). *An introduction to probability theory and its applications*. New York, NY: Wiley.

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

[19] Fischer, S., & Wegener, I. (2004). The Ising model on the ring: Mutation versus recombination. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 1113–1124.

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

[21] Friedman, N., & Yakhini, Z. (1996). On the sample complexity of learning Bayesian networks. In Horvitz, E., & Jensen, F. (Eds.), *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-96)* (pp. 274–282). San Francisco: Morgan Kaufmann Publishers.

[22] Galluccio, A., & Loebl, M. (1999a). A theory of Pfaffian orientations. I. Perfect matchings and permanents. *Electronic Journal of Combinatorics*, *6*(1). Research Paper 6.

[23] Galluccio, A., & Loebl, M. (1999b). A theory of Pfaffian orientations. II. T-joins, k-cuts, and duality of enumeration. *Electronic Journal of Combinatorics*, *6*(1). Research Paper 7.

[24] Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem. In Davis, L. (Ed.), *Genetic Algorithms and Simulated Annealing* (Chapter 6, pp. 74–88). Los Altos, CA: Morgan Kaufmann.

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

[26] Goldberg, D. E. (1999). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, 212–219.

[27] Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*, Volume 7 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic Publishers.

[28] Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362.

[29] Goldberg, D. E., & Voessner, S. (1999). Optimizing global-local search hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, 220–228.

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

[31] Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, *7*(3), 231–253.

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

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

[34] Hartmann, A. K. (2001). Ground-state clusters of two, three and four-dimensional +/-J Ising spin glasses. *Phys. Rev. E*, *63*, 016106.

[35] Hartmann, A. K., & Rieger, H. (2001). *Optimization algorithms in physics*. Weinheim: Wiley-VCH.

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

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

[38] 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.

[39] Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In Lemmer, J. F., & Kanal, L. N. (Eds.), *Uncertainty in Artificial Intelligence* (pp. 149–163). Amsterdam, London, New York: Elsevier.

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

[41] Höns, R. (2005). *Estimation of distribution algorithms and minimum relative entropy*. Doctoral dissertation, University of Bonn, Germany.

[42] 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.

[43] Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: The MIT Press.

[44] Kulish, V. V. (2002). *Hierarchical methods: Hierarchy and hierarchical asymptotic methods in electrodynamics*. Dordrecht: Kluwer.

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

[46] Lima, C. F., Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E., & Lobo, F. G. (2006). Substructural neighborhoods for local search in the Bayesian optimization algorithm. In Runarsson, T. P., et al. (Eds.), *Parallel Problem Solving from Nature (PPSN IX)* (pp. 232–241). Berlin: Springer Verlag.

[47] Lima, C. F., Sastry, K., Goldberg, D. E., & Lobo, F. G. (2005). Combining competent crossover and mutation operators: A probabilistic model building approach. In Beyer, H.-G., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)* (pp. 735–742). New York: ACM Press.

[48] Mendiburu, A., J. A. Lozano, J., & Miguel-Alonso, J. (2005). Parallel implementation of edas based on probabilistic graphical models. *IEEE Transactions on Evolutionary Computation*, *9*(4), 406–423.

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

[50] Mühlenbein, H., & Mahnig, T. (1998). Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, *7*(1), 19–32.

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

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

[53] Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In Voigt, H.-M., et al. (Eds.), *Parallel Problem Solving from Nature (PPSN IV)* (pp. 178–187). Berlin: Springer Verlag.

[54] Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

[55] Naudts, B., & Naudts, J. (1998). The effect of spin-flip symmetry on the performance of the simple GA. In Eiben, A. E., Bäck, T., Schoenauer, M., & Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature (PPSN V)* (pp. 67–76). Berlin Heidelberg: Springer Verlag.

[56] Ocenasek, J., & Pelikan, M. (2004). Parallel mixed Bayesian optimization algorithm: A scaleup analysis. In S. Cagnoni (Ed.), *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*. Electronic publication.

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

[58] Ocenasek, J., Schwarz, J., & Pelikan, M. (2003). Design of multithreaded estimation of distribution algorithms. In Cantú-Paz, E., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)* (pp. 1247–1258). Berlin: Springer.

[59] Pattee, H. H. (Ed.) (1973). *Hierarchy theory: The challenge of complex systems*. New York, NY: Braziller.

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

[61] Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL.

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

[63] Pelikan, M., & Goldberg, D. E. (2000a). Hierarchical problem solving and the Bayesian optimization algorithm. In Whitley, D., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)* (pp. 275–282). San Francisco, CA: Morgan Kaufmann.

[64] Pelikan, M., & Goldberg, D. E. (2000b). Research on the Bayesian optimization algorithm. In Wu, A. (Ed.), *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)* (pp. 216–219). San Fransisco, CA: Morgan Kaufmann.

[65] Pelikan, M., & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. In Spector, L., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 511–518). San Francisco, CA: Morgan Kaufmann.

[66] Pelikan, M., & Goldberg, D. E. (2003a). Hierarchical BOA solves Ising spin glasses and MAXSAT. In Cantú-Paz, E., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, Volume II (pp. 1275–1286). San Francisco, CA: Morgan Kaufmann.

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

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

[69] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In Banzhaf, W., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, Volume I (pp. 525–532). San Fransisco, CA: Morgan Kaufmann.

[70] 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.

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

[72] Pelikan, M., & Sastry, K. (2004). Fitness inheritance in the Bayesian optimization algorithm. In Deb, K., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, Volume 2 (pp. 48–59). Berlin: Springer Verlag.

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

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

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

[76] Sacerdoti, E. D. (1975). The nonlinear nature of plans. In *Proceedings of the Fourth Annual International Joint Conference on Aritificial Intelligence* (pp. 206–214). Tbilisi, Georgia, USSR.

[77] Santana, R. (2005). Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, *13*(1), 67–97.

[78] 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.

[79] Sastry, K., & Goldberg, D. E. (2000). *On extended compact genetic algorithm* (IlliGAL Report No. 2000026). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

[80] Sastry, K., Goldberg, D. E., & Pelikan, M. (2001). Don't evaluate, inherit. In Spector, L., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 551–558). San Francisco, CA: Morgan Kaufmann.

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

[82] Sastry, K., Pelikan, M., & Goldberg, D. E. (2006). Efficiency enhancement of estimation of distribution algorithms. In Cantú-Paz, E., Pelikan, M., & Sastry, K. (Eds.), *Scalable optimization via probabilistic modeling: From algorithms to applications* (pp. 161–185). Springer.

[83] Schwarz, J., & Ocenasek, J. (2000). A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning. In *Proceedings of the Fourth Joint Conference on Knowledge-Based Software Engineering* (pp. 51–58). Brno, Czech Republic: IO Press.

[84] Simon, H. A. (1968). *The sciences of the artificial*. Cambridge, MA: The MIT Press.

[85] Sinha, A., & Goldberg, D. E. (2001). Verification and extension of the theory of global-local hybrids. In Spector, L., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 591–597). San Francisco, CA: Morgan Kaufmann.

[86] Srivastava, R., & Goldberg, D. E. (2001). Verification of the theory of genetic and evolutionary continuation. In Spector, L., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 551–558). San Francisco, CA: Morgan Kaufmann.

[87] Stefik, M. (1981a). Planning and meta-planning (MOLGEN: Part 2). *Artificial Intelligence*, *16*(2), 141–170.

[88] Stefik, M. (1981b). Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, *16*(2), 111–140.

[89] Thierens, D. (1995). *Analysis and design of genetic algorithms.* Doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium.

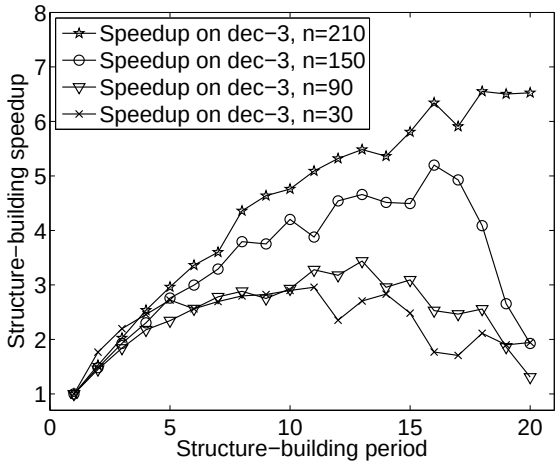
[90] Thierens, D., Goldberg, D. E., & Pereira, A. G. (1998). Domino convergence, drift, and the temporal-salience structure of problems. In *Proceedings of the International Conference on Evolutionary Computation (ICEC-98)* (pp. 535–540). Picataway, NJ: IEEE Press.

[91] Van Hoyweghen, C. (2001). Detecting spin-flip symmetry in optimization problems. In Kallel, L., et al. (Eds.), *Theoretical Aspects of Evolutionary Computing* (pp. 423–437). Berlin: Springer.

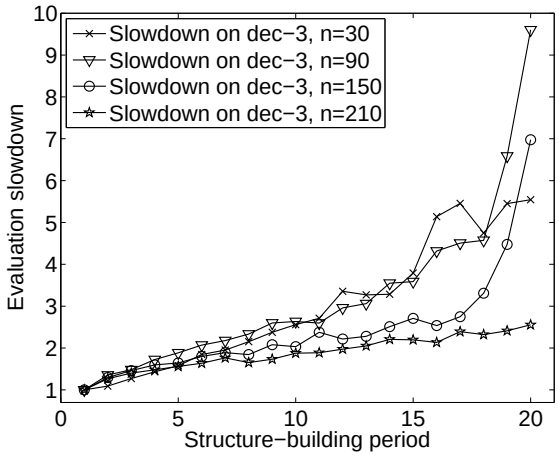
[92] Watson, R. A., Hornby, G. S., & Pollack, J. B. (1998). Modeling building-block interdependency. In Eiben, A. E., Bäck, T., Schoenauer, M., & Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature (PPSN V)* (pp. 97–106). Berlin: Springer Verlag.

[93] Whitley, L. D. (1991). Fundamental principles of deception in genetic search. In Rawlins, G. (Ed.), *Foundations of Genetic Algorithms* (pp. 221–241). San Mateo, CA: Morgan Kaufmann.

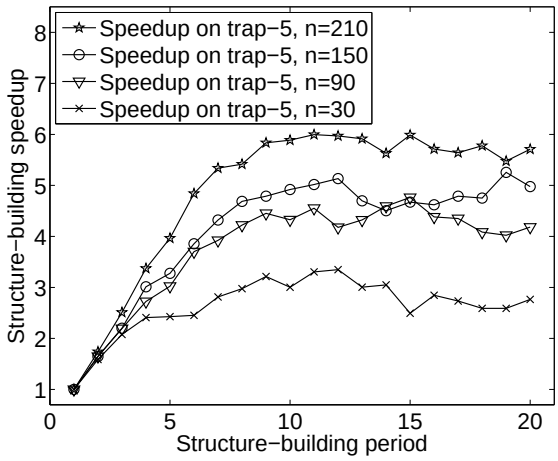
[94] Young, A. (Ed.) (1998). *Spin glasses and random fields.* Singapore: World Scientific.



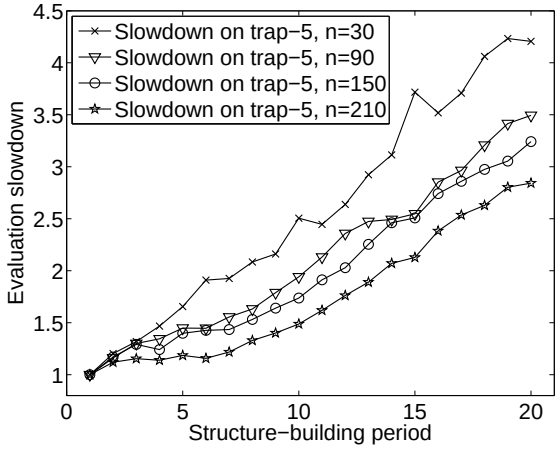
(a) Structure-building speedup for dec-3.



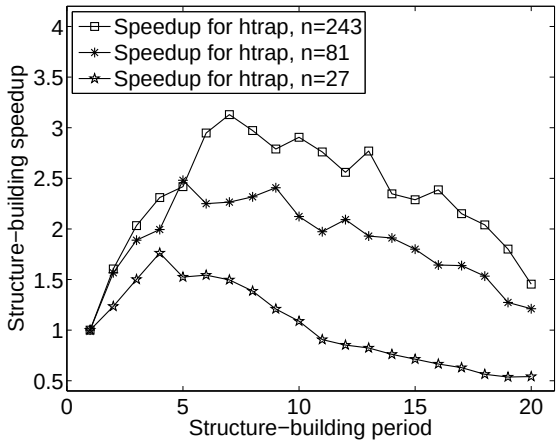
(b) Evaluation slowdown for dec-3.



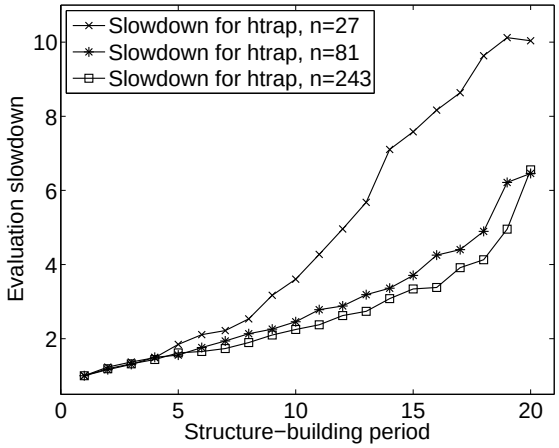
(c) Structure-building speedup for trap-5.



(d) Evaluation slowdown for trap-5.

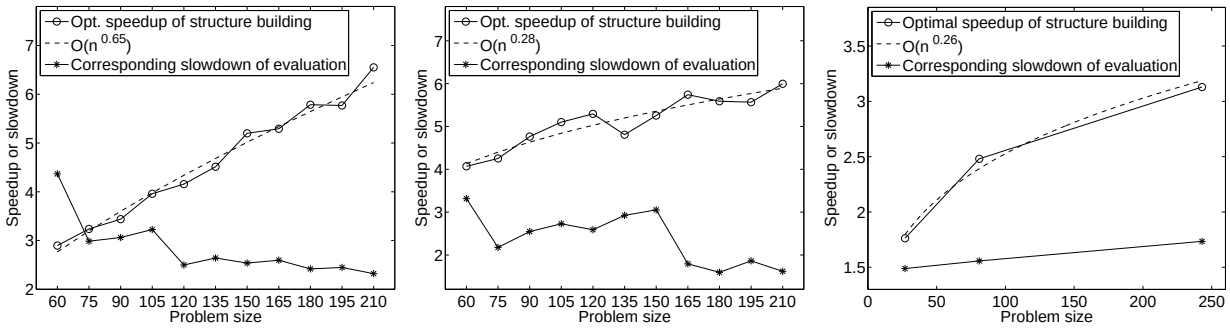


(e) Structure-building speedup for hTrap.



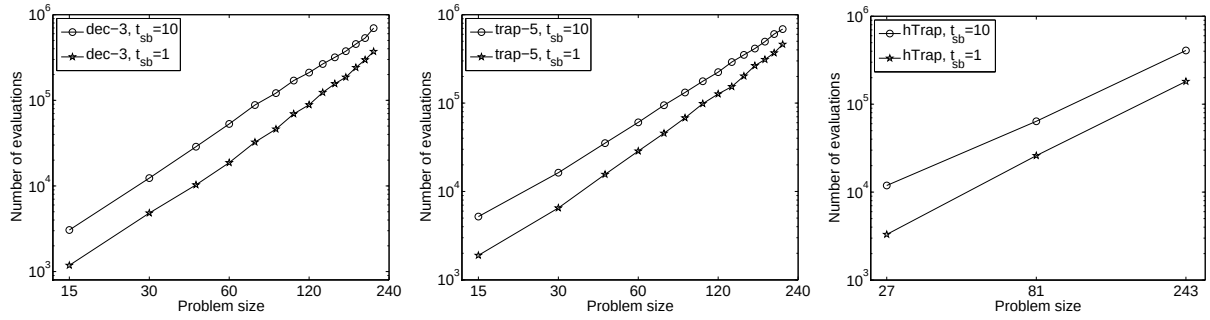
(f) Evaluation slowdown for hTrap.

Figure 3: Structure-building speedup and evaluation slowdown for dec-3, trap-5 and hTrap.



(a) Dec-3 (b) Trap-5 (c) hTrap

Figure 4: The optimal speedup of structure building and the corresponding evaluation slowdown for dec-3, trap-5, and hTrap.



(a) Dec-3 (b) Trap-5 (c) hTrap

Figure 5: The growth of the number of evaluations for dec-3, trap-5 and hTrap with $t_{sb} = 1$ (build structure always) and $t_{sb} = 10$ (build structure in every 10th iteration).

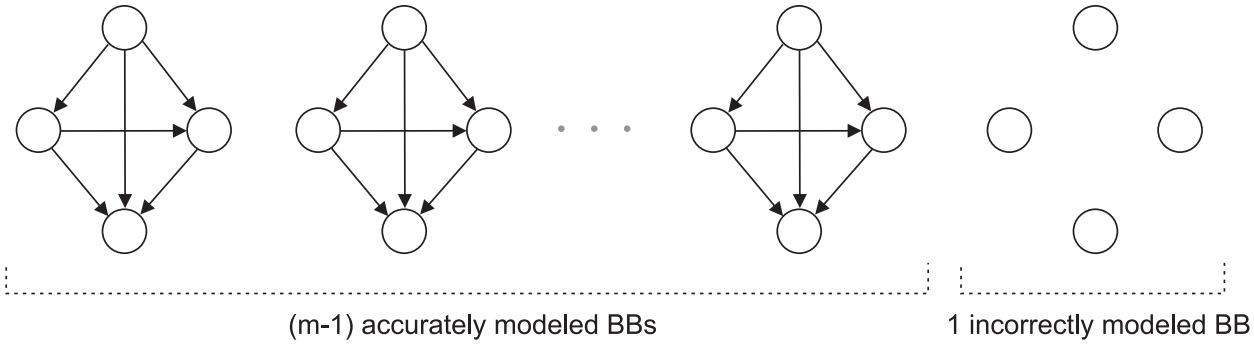


Figure 6: In deriving an upper bound on the structure-building period, we assume that BOA accurately discovers $(m - 1)$ out of m subproblems but it fails to discover one subproblem, the variables of which are modeled as being independent of each other.

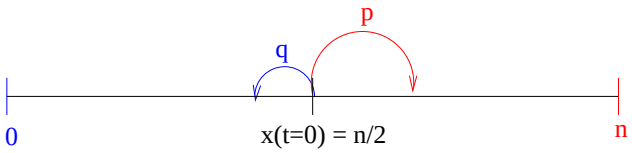


Figure 7: Gambler's ruin model illustration

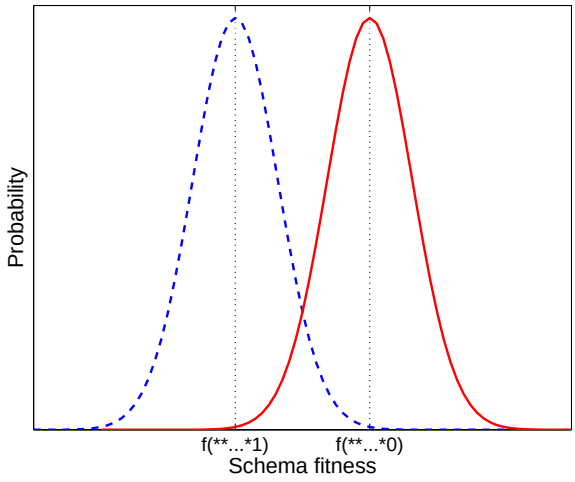
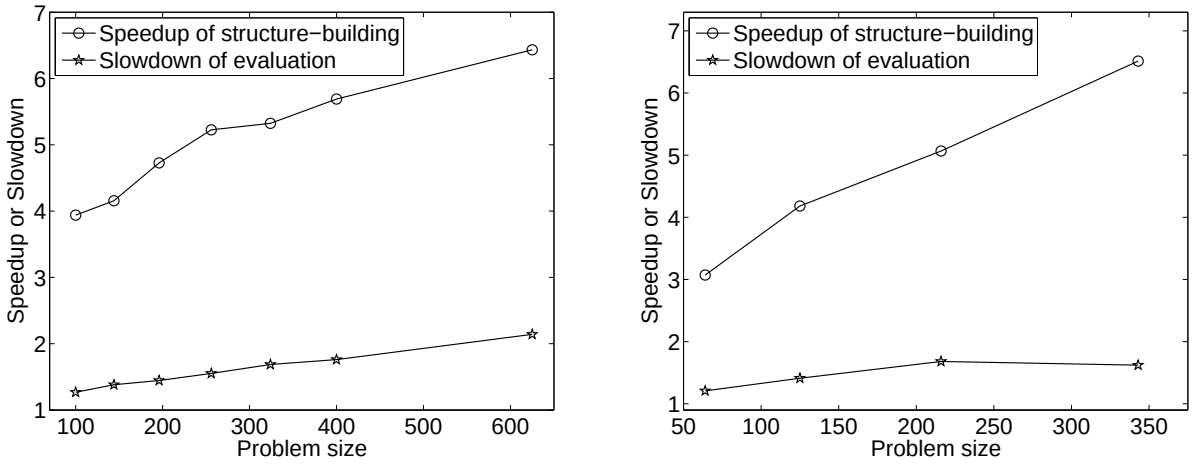


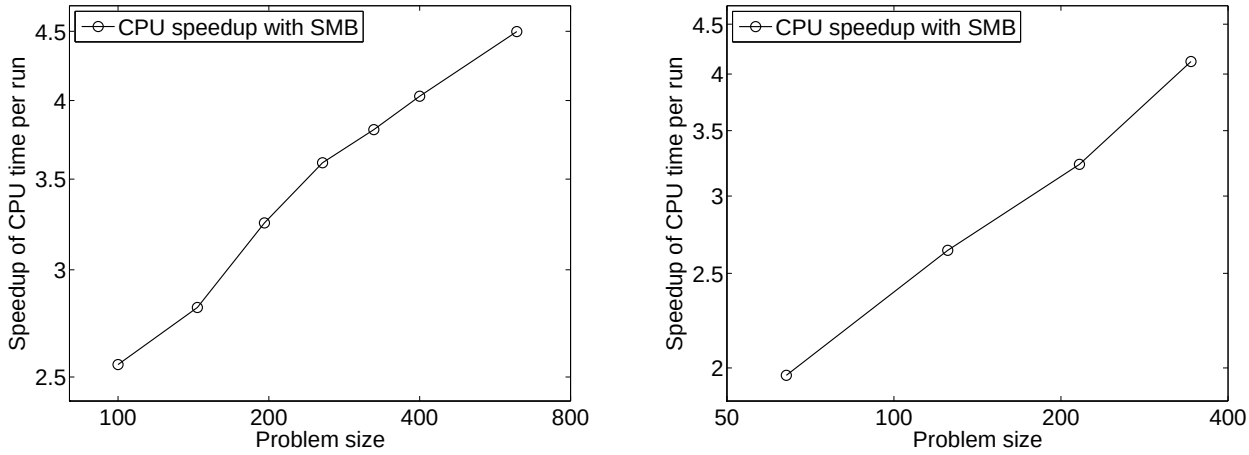
Figure 8: Decision making among competing building blocks



(a) 2D spin glass.

(b) 3D spin glass.

Figure 9: Structure-building speedup and evaluation slowdown for hBOA with SMB on 2D and 3D Ising spin glasses with $\pm J$ couplings and periodic boundary conditions. The structure-building period is $t_{sb} = \sqrt{n}/2$.



(a) 2D spin glass.

(b) 3D spin glass.

Figure 10: CPU-time speedup for hBOA with SMB on 2D and 3D Ising spin glasses with $\pm J$ couplings and periodic boundary conditions. The structure-building period is $t_{sb} = \sqrt{n}/2$.