Missouri Estimation of Distribution Algorithms Laboratory

Order or Not: Does Parallelization of Model Building in hBOA Affect Its Scalability?

Martin Pelikan and James D. Laury, Jr.

MEDAL Report No. 2006005

June 2006

Abstract

It has been shown that model building in the hierarchical Bayesian optimization algorithm (hBOA) can be efficiently parallelized by randomly generating an ancestral ordering of the nodes of the network prior to learning the network structure and allowing only dependencies consistent with the generated ordering. However, it has not been thoroughly shown that this approach to restricting probabilistic models does not affect scalability of hBOA on important classes of problems. This paper demonstrates that although the use of a random ancestral ordering restricts the structure of considered models to allow efficient parallelization of model building, its effects on hBOA performance and scalability are negligible.

Keywords

Hierarchical Bayesian optimization algorithm, efficiency enhancement, parallel model building, scalability.

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/

Order or Not: Does Parallelization of Model Building in hBOA Affect Its Scalability?

Martin Pelikan
Missouri Estimation of Distribution Algorithms Lab (MEDAL)
Dept. of Math and Computer Science, 320 CCB
University of Missouri at St. Louis
One University Blvd.
St. Louis, MO 63121
pelikan@cs.umsl.edu

James D. Laury, Jr.
Missouri Estimation of Distribution Algorithms Lab (MEDAL)
Dept. of Math and Computer Science, 356 CCB
University of Missouri at St. Louis
One University Blvd.
St. Louis, MO 63121
jlaury01@arch.umsl.edu

Abstract

It has been shown that model building in the hierarchical Bayesian optimization algorithm (hBOA) can be efficiently parallelized by randomly generating an ancestral ordering of the nodes of the network prior to learning the network structure and allowing only dependencies consistent with the generated ordering. However, it has not been thoroughly shown that this approach to restricting probabilistic models does not affect scalability of hBOA on important classes of problems. This paper demonstrates that although the use of a random ancestral ordering restricts the structure of considered models to allow efficient parallelization of model building, its effects on hBOA performance and scalability are negligible.

1 Introduction

The hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan, 2005) can solve nearly decomposable and hierarchical problems in a quadratic or subquadratic number of function evaluations. Nonetheless, low-order polynomial scalability may still be insufficient to solve problems with a large number of decision variables or large order of interactions. That is why a number of efficiency enhancement techniques have been proposed that address the most important computational bottlenecks of hBOA: fitness evaluation and model building.

One of the most promising approaches to enhancing the efficiency of model building in hBOA is to parallelize the learning of model structure, which is the most expensive part of model building. However, efficient parallelization of model building represents a challenging problem because straightforward approaches to parallelizing model building necessitate heavy communication between the computational nodes (Ocenasek & Schwarz, 2000; Ocenasek, 2002). Nonetheless, if one

restricts the class of models by generating an ancestral ordering of the variables and then considering only models consistent with the generated ordering, model building can be parallelized with nearly no communication between the computational nodes and nearly linear speedups of model building can be obtained even for relatively many computational nodes (Ocenasek & Schwarz, 2000; Ocenasek, 2002).

While it has been shown that the above approach enables efficient utilization of computational resources, it has not been investigated thoroughly how it affects scalability of hBOA on important classes of problems. The purpose of this paper is to test hBOA scalability with and without the modification that restricts models according to a random topological ordering of variables generated prior to model construction. As test problems, we consider standard nearly decomposable and hierarchical problems, including concatenated traps, hierarchical traps, and Ising spin glasses.

The paper is organized as follows. Section 2 outlines hBOA and its basic components. Section 3 describes the considered method for parallelizing model construction in hBOA. Section 4 provides and discusses experimental results. Finally, section 5 concludes and summarizes the paper.

2 Hierarchical Bayesian Optimization Algorithm (hBOA)

This section outlines the hierarchical Bayesian optimization algorithm and its components. Additionally, the section briefly discusses efficiency enhancement of hBOA and other EDAs.

2.1 Basic hBOA Algorithm

The hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan, 2005) evolves a population of candidate solutions represented by fixed-length strings over a finite alphabet. Here we assume that candidate solutions are represented by n-bit binary strings. The population is initially generated at random according to a uniform distribution over all n-bit strings. Each iteration starts by selecting a population of promising solutions using any common selection method of genetic and evolutionary algorithms, such as tournament and truncation selection. We use binary tournament selection. New solutions are generated by building a Bayesian network with decision trees (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999) for the selected solutions and sampling the built Bayesian network. To ensure useful diversity maintenance, the new candidate solutions are incorporated into the original population using restricted tournament replacement (RTR) (Harik, 1995). The run is terminated when termination criteria are met.

2.2 Bayesian Networks with Decision Trees

To model promising solutions and sample new candidate solutions, hBOA uses Bayesian networks with decision trees. A Bayesian network consists of two components: structure and parameters. The structure consists of a directed acyclic graph where the nodes correspond to variables (string positions) and the edges represent direct conditional dependencies. A Bayesian network represents the joint probability distribution

$$p(X) = \prod_{i=0}^{n-1} p(X_i|\Pi_i), \tag{1}$$

where $X = (X_0, \ldots, X_{n-1})$ is a vector of all the variables in the problem; Π_i is the set of parents of X_i in the network (the set of nodes from which there exists an edge to X_i); and $p(X_i|\Pi_i)$ is the conditional probability of X_i given its parents Π_i.

To represent conditional probabilities efficiently, hBOA uses decision trees. For variable X_i, there is a special decision tree T_i that encodes conditional probabilities $p(X_i|\Pi_i)$; 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, whereas 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.

2.3 Learning and Sampling Bayesian Networks

Learning a Bayesian network with decision trees consists of two steps (Heckerman, Geiger, & Chickering, 1994): (1) learn the structure, and (2) learn the parameters (conditional probabilities).

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 (Heckerman et al., 1994). 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.

To learn the structure of a Bayesian network with decision trees, a simple greedy algorithm is used (Heckerman, Geiger, & Chickering, 1994; Chickering, Heckerman, & Meek, 1997). The greedy algorithm starts with an empty network, which is represented by single-node decision trees. Each iteration splits one leaf of any decision tree. The split is selected in order to maximize the improvement of network quality. The algorithm terminates when no more improvement is possible. There are several approaches to measuring quality of network structures, for example, one can use the Bayesian-Dirichlet metric with likelihood equivalence (BDe) (Cooper & Herskovits, 1992; Heckerman et al., 1994; Chickering et al., 1997).

Splitting a leaf may increase the number of parents of the corresponding variable, depending on whether a split on this variable has already been made in this tree. The corresponding Bayesian network structure is implicitly defined by the set of decision trees. For more details on learning BNs with decision trees, see (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999; Pelikan, 2005).

The sampling of a Bayesian network with decision trees can be done using probabilistic logic sampling (Henrion, 1988), where the ancestral ordering of the variables is first determined, in which each variable is preceded by its parents. Then, the values of all variables are generated according to the ancestral ordering.

The asymptotic time complexity of building the network structure dominates the overall complexity of the variation operator that consists of building and sampling a BN (Pelikan, 2005). A similar behavior can be observed in other EDAs that use complex probabilistic models, for example, in the extended compact genetic algorithm (ECGA) (Harik, 1999; Sastry & Goldberg, 2000). If model building is the computational bottleneck, building the network structure is thus the most

important component to tackle with efficiency enhancement techniques.

2.4 Efficiency Enhancement Techniques for hBOA and Other EDAs

To solve large and complex problems, it is necessary to use scalable optimization techniques. Nonetheless, even when the time complexity of an algorithm grows only quadratically with the number of decision variables, the computation may become intractable for extremely large problems with thousands of decision variables and problems for which the evaluation of solution quality is computationally expensive. That is why it is important to develop efficiency enhancement techniques, which provide additional mechanisms for speeding up hBOA and other evolutionary algorithms (Goldberg, 2002; Sastry, 2001; Pelikan, 2005; Sastry, Pelikan, & Goldberg, 2006). From the practitioner's point of view, while scalability of optimizers addresses the transition *from intractability to tractability*, efficiency enhancement techniques address the transition *from tractability to practicality* (Goldberg, 2002). There are two potential computational bottlenecks of hBOA and other EDAs: (1) fitness evaluation and (2) model building. Efficiency enhancement techniques that address the above bottlenecks in hBOA can be classified into the following categories:

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.

This paper addresses parallelization; specifically, we analyze how parallelization of model building in hBOA affects hBOA performance and scalability.

3 Parallelization of Model Building in hBOA

There are several potential approaches to parallelization of the greedy algorithm for learning the Bayesian network structure (Ocenasek, 2002). Probably the most efficient approach to parallelization is to distribute the nodes of the Bayesian network and let each processor identify parents of the nodes assigned to this processor. However, since the network must remain acyclic at all times, a straightforward approach to implementing this method will necessitate communication between all processors after adding a new parent to any node of the network, which will result in a highly inefficient use of parallel computer architectures.

To tackle the problem of heavy interprocessor communication, Ocenasek and Schwarz (2000) proposed to generate a random ancestral ordering of the variables in the network before learning the network structure and to allow only network structures that are consistent with the generated ordering so that the parents of each variable are restricted to the variables that precede this variable in the generated ordering. A new ancestral ordering is generated in each generation to ensure that the bias on model structures changes over time. In this manner, the set of parents of each variable

can be determined independently of the set of parents of any other variable and the different processors thus do not need to exchange any information until the model is finalized.

In addition to minimizing the communication between processors, it is important to ensure that the computational load on each processor is approximately equal. Since the computational complexity of processing each variable grows linearly with the number of potential parents of the variable (Ocenasek & Schwarz, 2000), it is best to assign nodes to processors so that each processor contains nodes with the same total number of potential parents. For example, two nodes can be assigned to each processor, where ith processor will be given the ith node from the beginning of the ancestral ordering and the ith node from the end of the ordering (Ocenasek & Schwarz, 2000).

While restricting the models to ensure their consistency with the generated ancestral ordering ensures an efficient use of parallel computers (Ocenasek & Schwarz, 2000; Ocenasek, 2002; Ocenasek, Schwarz, & Pelikan, 2003), it also restricts the class of considered models analogically to the K2 learning algorithm (Cooper & Herskovits, 1992; Heckerman et al., 1994). The models can thus be expected to represent the actual distribution of promising solutions less accurately and the performance of hBOA may be affected, especially for problems with complex structure. The next section verifies the effects of the aforementioned restrictions of model structure on hBOA performance on several important classes of nearly decomposable and hierarchical problems.

4 Experiments

We have performed experiments on several common nearly decomposable and hierarchical problems, including concatenated deceptive problems of order 3 and 5, hierarchical traps, and the problem of finding ground states of 2D $\pm J$ Ising spin glasses. The primary goal of experiments is to analyze the effects of modifying the model building algorithm according to the last section on hBOA performance; specifically, we study *scalability*, that is, how computational complexity of hBOA measured by the number of evaluations grows with problem size.

4.1 Test Problems

A brief description of the test problems follows:

(1) *Dec-3: Concatenated deceptive of order 3.* In dec-3 (Deb & Goldberg, 1994), 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:

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

where u is the number of ones in the input string of 3 bits. An n-bit dec-3 function has one global optimum in the string of all ones 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 (Thierens, 1995; Bosman & Thierens, 1999; Pelikan, Goldberg, & Cantú-Paz, 1999).

(2) *Trap-5: Concatenated trap of order 5.* Trap-5 can be 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 (Ackley, 1987; Deb & Goldberg, 1991)

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

where u is the number of ones in the input string of 5 bits. An n-bit trap-5 has one global optimum in the string of all ones and $2^{n/5} - 1$ other local optima. Trap-5 also necessitates that all bits in each group are treated together, because statistics of lower order are misleading (Deb & Goldberg, 1991).

(3) *hTrap: Hierarchical trap.* Hierarchical traps (hTraps) (Pelikan, 2002) cannot be tractably solved using single-level decomposition, but can be efficiently solved using a competent hierarchical optimizer. hTraps 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, $n = 3^l$ where l is the number of levels.

In the variant of hTrap used in this work, on the lowest level, groups of 3 bits contribute to the overall fitness using a generalized 3-bit trap

$$trap_3(u) = \begin{cases} f_{high} & \text{if } u = 3 \\ f_{low} - u\frac{f_{low}}{2} & \text{otherwise} \end{cases}, \tag{5}$$

where $f_{high} = 1$ and $f_{low} = 1 + 0.1/l$. Note that for these values of f_{high} and f_{low}, the optimum of the trap is 000.

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 (see eq. 5), and the groups are mapped to an even higher level. This continues until the top level is evaluated that contains 3 bits total. However, on the top level, a trap with $f_{high} = 1$ and $f_{low} = 0.9$ is applied. As a result, the optimum of hTrap is in the string of all ones despite the superior performance of blocks of zeros on any level except for the top one. Any group of bits containing the null symbol does not contribute to the overall fitness.

To make the overall contribution at each level of the same magnitude, the contributions of traps on ith level from the bottom are multiplied by 3^i.

hTraps have many local optima, but only one global optimum in the string of all ones. Nonetheless, any single-level decomposition into subproblems of bounded order will lead away from the global optimum. 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 if found.

(4) *Ising spin glass.* A 2D Ising spin glass is arranged on a regular 2D grid where each node i corresponds to a spin $s_i \in \{+1, -1\}$ 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 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 element along each dimension. A specific set of coupling constants define a spin-glass instance. Each possible setting of all spins is called a spin configuration.

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

$$E(C) = \sum_{\langle i,j \rangle} s_i J_{i,j} s_j \ , \tag{6}$$

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

Here the task is to find ground states given a set of coupling constants, where a ground state is defined as a spin configuration with the minimum possible energy. Finding ground states is a challenging problem because of the rough energy landscape and a large order of interactions (Hartmann, 2001; Pelikan & Goldberg, 2003; Dayal, Trebst, Wessel, Würtz, Troyer, Sabhapandit, & Coppersmith, 2004; Pelikan, 2005).

For each problem size, we generate and test 1000 random spin glass instances where each coupling is set randomly to either $+1$ or -1 with equal probabilities.

4.2 Description of Experiments

To study scalability, we consider a range of problem sizes for each test problem. The results are then used to investigate the growth of the number of function evaluations until successful convergence to the global optimum with respect to the problem size.

For all problems and problem sizes except for spin glasses, bisection is ran to determine the minimum population size to ensure reliable convergence to the global optimum in 30 out of 30 independent runs (Pelikan, Goldberg, & Lobo, 2002). For spin glasses, only 5 successful runs out of 5 runs are required. Binary tournament selection is used to select promising solutions. The population of new solutions has the same size as the original population and RTR (where $w = \min\{n, N/20\}$) is used to incorporate new solutions. To construct the model, we use the BDe metric for decision trees (Chickering, Heckerman, & Meek, 1997) modified by subtracting a complexity penalty term (Pelikan, Goldberg, & Sastry, 2001; Pelikan, 2005). A greedy network construction algorithm is used to construct network structure.

For spin glasses, we also use the deterministic hill climber (DHC) to improve all candidate solutions in the population. In each iteration, DHC performs a single-bit change on the solution that leads 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.

4.3 Results

Figures 1 and 2 show the growth of the number of evaluations with problem size for the two separable problems of bounded difficulty: dec-3 and trap-5. The growth of the number of evaluations on dec-3 and trap-5 is also summarized in the following table:

Number of bits, n	dec-3	dec-3, ordered nodes	trap-5	trap-5, ordered nodes
30	5616.00	4653.67	9062.40	8195.83
60	21594.00	23766.67	28562.50	29625.00
90	41750.00	46250.00	77866.67	75250.00
120	103400.00	107158.33	157516.67	132166.67
150	153766.67	184933.33	229866.67	253016.67

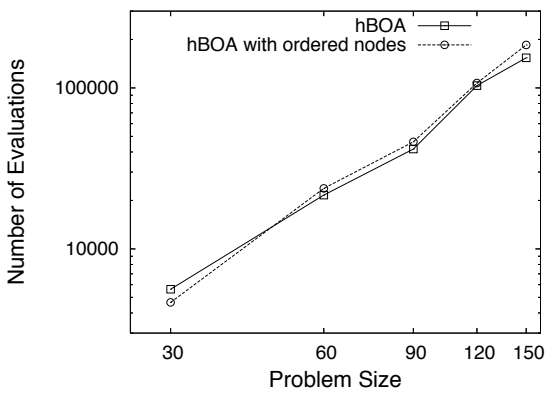

Figure 1: Results on dec-3.

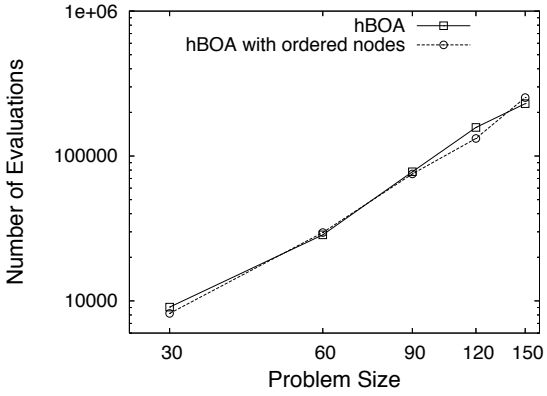

Figure 2: Results on trap-5.

The results on dec-3 and trap-5 indicate that for separable problems of bounded difficulty, restricting the Bayesian network according to a randomly generated ancestral ordering of variables does not affect hBOA performance significantly.

Figures 3 and 4 show the growth of the number of evaluations with problem size for test problems that cannot be decomposed into independent subproblems of bounded order: htrap and 2D Ising spin glass. The growth of the number of evaluations on htrap and spin 2D Ising spin glass is shown also in the following two tables:

Number of bits, n	htrap	htrap, ordered nodes
27	3567.20	3737.30
81	28833.33	26923.13
243	182466.67	169416.67

Number of bits, n	2D spin glass	2D spin glass, ordered nodes
$6 \times 6 = 36$	70.19	66.40
$8 \times 8 = 64$	231.86	229.95
$10 \times 10 = 100$	546.88	571.62
$12 \times 12 = 144$	1032.54	1103.11
$14 \times 14 = 196$	1806.87	2001.57

Analogically to the results on dec-3 and trap-5, restricting the probabilistic models does not affect hBOA performance on htrap and 2D Ising spin glasses significantly.

The results thus indicate that for all test problems, restricting the probabilistic models in hBOA to models that are consistent with a randomly generated ancestral ordering does not significantly affect hBOA performance. Consequently, using this modification will ensure efficient parallelization of model building when model building becomes the computational bottleneck.

5 Summary and Conclusions

To efficiently parallelize model building in hBOA, it is useful to generate an ancestral ordering of nodes in the network prior to building the network structure, and restrict models to those consistent with the generated ancestral ordering. While modifying the model building procedure allows its

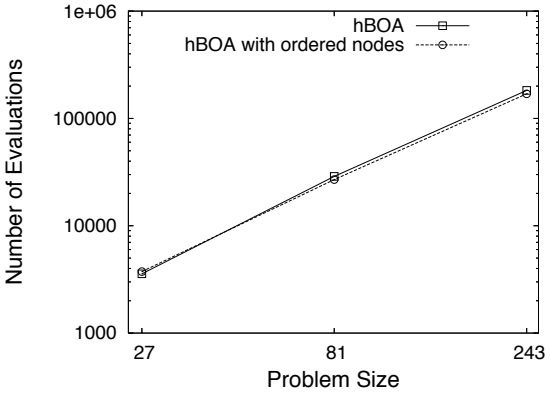

Figure 3: Results on hierarchical traps.

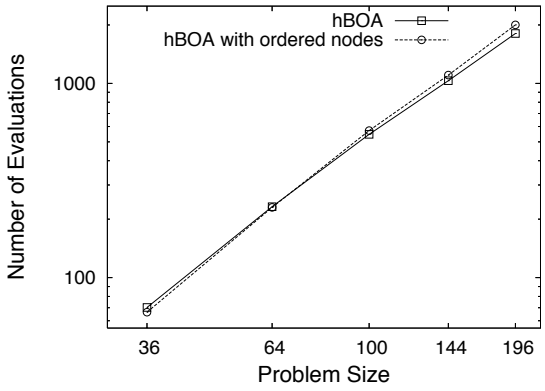

Figure 4: Results on 2D $\pm J$ Ising spin glasses.

efficient parallel implementation, it also restricts the considered class of network structures, potentially affecting performance and scalability of hBOA. This paper provides empirical evidence that the modification of the model building algorithm does not significantly affect hBOA performance and scalability.

6 Acknowledgments

This work was supported by the National Science Foundation under CAREER grant ECS-0547013, the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, and 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. Most experiments were done using the hBOA software developed by Martin Pelikan and David E. Goldberg at the University of Illinois at Urbana-Champaign.

References

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

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.

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

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

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

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

Laboratory.

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

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

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.

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

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

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

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.

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.

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

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

Ocenasek, J., Schwarz, J., & Pelikan, M. (2003). Design of multithreaded estimation of distribution algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, 1247–1258.

Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. Also IlliGAL Report No. 2002023.

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

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

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

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

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

Pelikan, M., Goldberg, D. E., & Sastry, K. (2001). Bayesian optimization algorithm, decision graphs, and Occam's razor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 519–526. Also IlliGAL Report No. 2000020.

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. Also IlliGAL Report No. 2002004.

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.

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

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