Escaping Hierarchical Traps with Competent Genetic Algorithms

Martin Pelikan
Depts. of General Engineering and Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
pelikan@illigal.ge.uiuc.edu

David. E. Goldberg
Dept. of General Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801
deg@uiuc.edu

Abstract

To solve hierarchical problems, one must be able to learn the linkage, represent partial solutions efficiently, and assure effective niching. We propose the *hierarchical Bayesian optimization algorithm* which combines the Bayesian optimization algorithm, local structures in Bayesian networks, and a powerful niching technique. Additionally, we propose a class of hierarchically decomposable problems, called *hierarchical traps*, which are deceptive on each level. The proposed algorithm is shown to scale up subquadratically on all test problems. Empirical results are in agreement with recent theory.

1 INTRODUCTION

Genetic algorithms (GAs) (Holland, 1975; Goldberg, 1989) combine short partial solutions to form solutions of higher order. New solutions undergo selection and the process is repeated until the entire solution is formed. However, fixed, problem-independent, recombination operators have shown to perform quite poorly on problems with interactions among the variables spread across the solutions (Thierens & Goldberg, 1993; Pelikan, Goldberg, & Cantú-Paz, 1998). Moreover, the hierarchical nature of the optimization process has earned only little attention and it has been assumed that genetic algorithms do this automatically.

The purpose of this paper is to show that competent genetic algorithms which succeeded in solving problems of bounded difficulty on a single level quickly, accurately, and reliably, can be extended to solve problems that are hierarchical in their nature. We focus on the Bayesian optimization algorithm (BOA) (Pelikan et al., 1998) using decision graphs to represent the conditional probabilities of the model used to represent promising solutions (Pelikan et al., 2000).

There are three major issues one must address to succeed in solving difficult hierarchical problems: linkage learning, niching, and efficient representation of the model. Linkage learning ensures powerful recombination. Niching and efficient representation of the model ensure preservation of alternative partial solutions that are assembled to form solutions of higher order. We propose the *hierarchical Bayesian optimization algorithm* which addresses the three aforementioned issues by combining the Bayesian optimization algorithm, local structures in Bayesian networks, and a powerful niching technique. Hierarchical BOA is able to solve problems that are not only hierarchical, but that also mislead the algorithm toward some inferior optimum on each level. Additionally, we design a class of hierarchical test problems which require both efficient linkage learning and niching, and perform a number of experiments to show that hierarchical BOA is able to solve the problems efficiently. Due to their deceptive nature, the proposed problems are called *hierarchical deceptive traps*.

The paper starts by describing BOA, which uses Bayesian networks to model promising solutions and generate the new ones. Section 3 discusses the use of niching in genetic and evolutionary algorithms. Hierarchical BOA is described in Section 4. Test problems tackled in our experiments are presented in Section 5. Section 6 provides and discusses the results of our experiments. Section 7 concludes the paper.

2 BAYESIAN OPTIMIZATION ALGORITHM

By applying recombination and mutation, GAs are manipulating a large number of promising partial solutions. However, fixed, problem independent, recombination and mutation operators often result in inferior performance even on simple problems. Without knowing where the important partial solutions are and

designing problem specific operators that take this information into account, the required number of fitness evaluations and population size grow exponentially with the number of decision variables (Thierens & Goldberg, 1993).

That is why there has been a growing interest in *linkage learning* which studies methods that are able to learn where the important interactions in the problem are and use this information to combine solutions more effectively. One of the approaches to linkage learning is based on using probability distributions to model promising solutions found so far and generating new solutions according to the estimated distribution (Mühlenbein & Paaß, 1996; Pelikan, Goldberg, & Lobo, 2000). Probability distributions can capture variables which are correlated and the ones which are independent. This can subsequently be used to combine the solutions in more effective manner. An overview of methods based on this principle is beyond the scope of this paper and can be found in Pelikan, Goldberg, and Lobo (2000) and other related papers.

The Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1998) uses Bayesian networks to model promising solutions and subsequently guide the exploration of the search space. In BOA, the first population of strings is generated randomly with a uniform distribution. The initial population can be biased to the regions that we are interested in. From the current population, the better strings are selected. Any selection method can be used. A Bayesian network that fits the selected set of strings is constructed. Any metric as a measure of quality of networks and any search algorithm can be used to search over the networks in order to maximize/minimize the value of the used metric. Besides the set of good solutions, prior information about the problem can be used in order to enhance the estimation and subsequently improve convergence. New strings are generated according to the joint distribution encoded by the constructed network. The new strings are added into the old population, replacing some of the old ones.

The next subsection describes basic principles of learning and utilization of Bayesian networks. Subsequently, local structures that can be used to make the representation of the model more efficient are discussed and a simple greedy algorithm for network construction is briefly described.

2.1 BAYESIAN NETWORKS

A Bayesian network (Pearl, 1988) is a directed acyclic graph with the nodes corresponding to the variables in the modeled data set (in our case, to the positions in solution strings). Mathematically, a Bayesian network encodes a joint probability distribution. A directed edge relates the variables so that in the encoded distribution, the variable corresponding to the terminal node is conditioned on the variable corresponding to the initial node. More incoming edges into a node result in a conditional probability of the corresponding variable with a conjunctional condition containing all its parents. The network encodes independence assumptions that each variable is independent of any of its antecedents in ancestral ordering given its parents.

To encode the conditional probabilities corresponding to the nodes of the network, one can use a simple probability table listing probabilities of all possible instances of a variable and its parents. However, the size of such a table grows exponentially with the number of parents of the variable even though many probabilities of higher order may be the same. To solve hierarchical problems, it is essential to be able to represent conditional probabilities by structures that are polynomial in the order of interactions. While the order of interactions can be as high as the size of the problem, the number of corresponding alternative partial solutions must be polynomial in their order to allow efficient and reliable exploration. The next subsection presents alternative ways to represent conditional probabilities in the model which allow a more compact representation of the local densities in the model.

2.2 LOCAL STRUCTURES IN BAYESIAN NETWORKS

One simple extension of the probability table is a *default table* (Friedman & Goldszmidt, 1999). In the default table, only some instances of the variable and its parents are listed together with the corresponding probabilities. The remaining probabilities are obtained from the default entry which is simply an average of the remaining (unlisted) probabilities.

One can use more complex local structures, such as *decision trees* (Friedman & Goldszmidt, 1999) or *decision graphs* (Chickering, Heckerman, & Meek, 1997). Each internal node of a decision tree or graph corresponds to some variable. Children (successors) of each internal node correspond to disjoint subsets of values the variable can obtain. For binary variables, each non-leaf node can have exactly two children where each child corresponds to one of the values zero and one. In case of bigger alphabets, there are more possibilities. A decision graph allows different parents to have the same child. This makes the structure both more general and expressive than decision trees. In hierarchical BOA we use decision graphs. However, there is only little difference between the performance

using decision graphs and trees. Since trees are simpler to interpret we used only decision trees in our experiments.

Using local structures can reduce the space we need to represent the model. Additionally, it can refine the model building by using smaller operators and make the model more general for some data sets. One can encode interactions of a high order without having to consider exponentially many instances and probabilities. Please, see Pelikan et al. (2000) for more details on using decision graphs in the BOA.

2.3 LEARNING BAYESIAN NETWORKS

To construct the network, a simple greedy algorithm is usually used. This algorithm performs simple graph operations that improve the quality of the current network the most, starting from an empty network or a network from a different source. To measure quality of each network, various scoring metrics can be used. Recently, we have used the Bayesian-Dirichlet metric, the minimum description length (MDL) metric, and a metric which is a combination of the Bayesian-Dirichlet and MDL metric. For more details on the network construction and scoring metrics for simple Bayesian networks and for networks with local structures, see Pelikan et al. (1998) and Pelikan et al. (2000). The next section discusses various approaches to niching. Subsequently, hierarchical BOA is described.

3 NICHING

The purpose of niching in genetic and evolutionary optimization is twofold: (1) discovery of multiple solutions of the problem and (2) preservation of alternative solutions until one can decide which solution is better. In some real-world applications it is important to find multiple solutions and let the expert or experiment decide which of the solutions is the best after all. The reason for preserving multiple alternative solutions is that on some difficult problems one cannot clearly determine which alternative solutions are really on the right track until the optimization proceeds for a number of generations. Without niching the population is a subject to genetic drift which may destroy some alternatives before we find out whether or not they are the ones we are looking for.

There are three general approaches to niching: fitness-sharing, selection-based, and island models. The following paragraphs briefly discuss each approach. It is beyond the scope of this paper to give a complete overview, and we refer the reader to the extended version of this paper (Pelikan & Goldberg, 2001).

The first approach modifies the fitness landscape before the selection is performed. Fitness sharing (Goldberg & Richardson, 1987) is based on this idea. In fitness sharing, the location of each individual is set to either its genotype or phenotype. The neighborhood of each individual is defined by the *sharing function*. An individual shares a niche with any individual that is within a certain range from its location. The effect may decrease with the distance and completely vanishes for distances greater than a certain threshold.

The second approach modifies the selection itself to take into account the fitness as well as the genotype or the phenotype instead of using the fitness as the only criterion. In *preselection* of Cavicchio (1970) the offspring replaced the inferior parent. This scheme was later generalized by De Jong (1975) who proposed *crowding*. In crowding, for each new individual a subset of the population is first selected. The new individual then replaces the most similar individual in this subset. Harik (1994) proposed the restricted tournament selection as an extension of De Jong's crowding. RTS proceeds just as crowding; however, the individual replaces the closest individual from the selected subset only if it is better in terms of fitness. Therefore, RTS introduces selection pressure and can replace the selection operator.

The third approach is to isolate several groups of individuals rather than to keep the entire population in one location. The location of each individual does not depend on its genotype or phenotype. The individuals can migrate between different locations (islands or demes) at certain intervals and allow population at each location to develop in isolation. There are two reasons why spatial separation should be desirable in genetic and evolutionary computation. One reason is that in nature the populations are actually divided in a number of subpopulations that (genetically) interact only rarely or do not interact at all. Another reason is that separating a number of subpopulations allows an effective parallel implementation and is therefore interesting from the point of view of computational efficiency.

Some related work studies the preservation of diversity from a different point of view. The primary goal of these techniques is not the preservation of multiple solutions or alternative search regions, but the avoidance of premature convergence. Various techniques for niching were also proposed in the area of multiobjective optimization. These methods are not applicable to single-criterion optimization and therefore we do not discuss them in this paper.

The BOA uses the set of selected solutions to learn a model of promising solutions. Fitness sharing would

affect the fitness and subsequently also the model construction. That is why it is desirable that we use a different niching method in the BOA. Spatial separation can be directly encoded in the probabilistic model by using mixture distributions or models with hidden variables. A simple method based on mixture models to reduce negative effects of symmetry in the problem on the BOA was proposed in Pelikan and Goldberg (2000a). However, to solve hierarchical problems, we must deal with a number of niches that can be exponential in the number of variables. Even though this implies exponentially sized populations, one can use the fact that the model itself preserves diversity quite well by that it makes many independence assumptions and uses these to generate new solutions. Only little extra pressure toward diversity preservation is then required. That is why we used the restricted tournament selection to incorporate niching into hierarchical BOA. Since the technique is used as a replacement technique and not as a primary source of selection pressure, we called the method *restricted tournament replacement*.

4 HIERARCHICAL BOA

As it was discussed above, hierarchical BOA uses Bayesian networks to learn the linkage. To efficiently represent partial solutions, local structures are used to represent local densities in the model. The remainder of this section describes restricted tournament replacement (RTR) used in hierarchical BOA to ensure effective niching.

RTR localizes the replacement in hierarchical BOA by selecting a sub-set of the original population for each new offspring and letting the offspring compete with the most similar member of this subset. If the new offspring is better, it replaces the corresponding individual. The measure of similarity can be based on either the genotype or the phenotype. In our experiments, we used Hamming distance to measure similarity. Since the generation of a probabilistic model in the BOA does not encourage using a steady state genetic algorithm, we incorporate niching in the replacement step of a traditional BOA.

It is important to set the size of the subsets that are selected to incorporate each new individual into the original population. The size of these subsets is called a *window size*. A window size should be proportional to the number of niches. We have tried a number of settings on various difficult problems. A window size proportional to the size of the problem yielded the best performance.

A window size proportional to the size of the problem can be supported by the following argument. For correct decision making on a single level, the population size must grow proportionally to the problem size (Pelikan, Goldberg, & Cantú-Paz, 2000). To maintain a certain number of niches, one must lower-bound the size of each niche by a certain constant. Therefore, a population size proportional to the problem size allows for maintenance of the number of niches proportional to the problem size. The number of niches that RTR can maintain is proportional to the window size. Therefore, the window size growing linearly with the size of the problem is the strongest niching one can afford without increasing population sizing requirements.

5 TEST PROBLEMS

In order to analyze the performance of hierarchical BOA on difficult hierarchical problems, most test problems are hierarchical. The remainder of this section describes test problems used in our experiments.

5.1 HIERARCHICALLY DECOMPOSABLE FUNCTIONS

Hierarchically decomposable functions (HDFs) (Watson, Hornby, & Pollack, 1998; Pelikan & Goldberg, 2000b) are a subclass of general additively decomposable functions (Pelikan, Goldberg, & Cantú-Paz, 1998). HDFs are defined on multiple levels where the input to each level is based on the solutions found on lower levels. The *fitness contribution* of each building block is separated from its *interpretation* (meaning) when it is used as a building block for constructing the solutions on a higher level. The overall fitness is computed as the sum of fitness contributions of each building block.

In spite of bounded difficulty of HDFs on each level, a hierarchical function can contain interactions of order equal to the size of the problem. Bounded difficulty on each level of the hierarchy makes HDFs solvable in polynomial time even though the problem is very difficult when viewed on a single level. It is important to note that hierarchical problems of bounded difficulty are a strictly more difficult class of problems than problems of bounded difficulty on a single level.

A hierarchically decomposable function is defined by its structure in the form of a tree with one-to-one mapping between the leaves and the variables in a problem, and two sets of functions: (1) the interpretation functions and (2) the contribution functions. The structure defines which blocks of interpretations to interpret to the next level and how, and which blocks contribute to the overall fitness on this level. The interpretation functions define how we interpret solutions from lower

levels to become inputs of the contribution and interpretation functions on a higher level. The contribution functions define how much do blocks of interpretations on each level contribute to the overall fitness.

The difficulty of hierarchical functions depends on the underlying structure as well as the contribution and interpretation functions. The hierarchical if-and-only-if (HIFF) function (Watson et al., 1998) uses the "if and only if" function on each level. More difficult functions have been proposed (Goldberg, 1997; Goldberg, 1998; Pelikan & Goldberg, 2000b), where functions deceive the algorithms to a local optimum on each level. Only at the top level it becomes clear which optimum is the global one.

In this paper, only simple structures such as balanced binary and ternary trees are used. The contribution of each subfunction on each level is scaled so that the contributions on all levels are of the same magnitude.

5.1.1 Hierarchical If-and-Only-If (HIFF)

The structure of the HIFF is a balanced binary tree. By $height(x)$ we denote the distance from the node x in the tree to one of its descendant leaves. Since the tree is balanced, the height is well-defined. Each leaf contributes to the fitness by 1. Each parent node x contributes to the overall fitness by $2^{height(x)}$ if and only if the interpretations of its children are both either 0 or 1. Otherwise, the contribution is 0. The two symbols are interpreted to their parent on the next level as 0 in case they are both 0's, 1 in case they are both 1's, and '-' otherwise. As input, the leaves of the tree get the input string with no change.

5.1.2 Hierarchical Trap Functions

Hierarchical traps use a balanced k-ary tree as the underlying structure, where $k \geq 3$. The interpretation functions interpret blocks of all 0's and 1's to 0 and 1, respectively, similarly to the HIFF. Everything else is interpreted into '-'.

Each contribution function is a trap function of order k. A trap function is a function of unitation, i.e. its value depends only on the number of ones in the input string. See Figure 1 for a graph of the trap function of order k. If there is a '-' in the input to this function, it simply returns 0.

The values of f_{high} and f_{low} define the heights of the two peaks. The trap function is fully deceptive whenever f_{high} is greater than f_{low} within some proportion depending on the order k of the function. See Deb and Goldberg (1994) for sufficient conditions of deception. If the function is deceptive or $f_{low} > f_{high}$, any schemata of order lower than k bias the search to the string all zeroes.

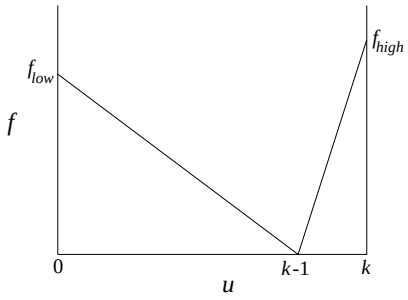


Figure 1: Trap function of order k.

In both functions used in our experiments, the underlying structure is a ternary tree ($k = 3$) and the leaves do not directly contribute to the overall fitness. For all non-leaf nodes x of the first hierarchical trap except for the root, the contribution is computed by a trap with equal peaks $f_{high} = f_{low} = 1$ multiplied by $3^{height(x)}$. The contribution of the root node is given by a trap with $f_{high} = 1$ and $f_{low} = 0.9$ multiplied by $3^{height(root)}$. In this fashion, the function biases the search to the solution of all zeroes on each but the top level. However, the optimum is in the string of all ones. The top level is also deceptive which makes the problem even harder. The above function is denoted by H-Trap1 in further text.

In the second function the bias toward solutions with many zeroes is made even stronger by making the peak f_{low} higher than the other peak everywhere except for the root. To keep the global optimum in the string of all ones, we set $f_{high} = 1$ and $f_{low} = 1 + 0.1/k$ for all non-root levels. This function is denoted by H-Trap2.

The HIFF function does not bias the search toward either global optimum. Unlike the HIFF, both hierarchical trap functions H-Trap1 and H-Trap2 bias the search toward the solution with all zeroes on all levels. However, the actual global optimum is in the string of all ones. Therefore, the functions are very difficult to solve and without effective linkage learning required to preserve the local optima on each level and niching required to preserve alternative partial solutions until solving the problem on the highest level, the algorithm cannot reach the global optimum. For a more detailed description of the test functions, see Pelikan and Goldberg (2001).

5.2 BIPOLAR FUNCTION

The bipolar deceptive function of order 6 is constructed by concatenating a number of bipolar subfunctions of order 6 (Deb, Horn, & Goldberg, 1992). See Pelikan and Goldberg (2001) for a full definition of the function. The bipolar function of size n has $2^{n/6}$

global and $20^{n/6}$ local optima. For $n = 30$, there are 32 global and $3,200,000$ local optima.

6 RESULTS

To show how hierarchical BOA scales up on difficult hierarchical problems, we performed tests on each function with varying problem size. For each problem size, we required that the algorithm find the global optimum in all 30 independent runs. The performance was measured by an average number of fitness evaluations until the optimum was found. The population size was determined empirically to minimize the number of fitness evaluations until the optimum was found. A window size was set to the problem size, i.e. $w = n$. We used decision trees to represent conditional probabilities in the model and construct the model. Prior distribution of models was biased toward simpler models (Pelikan et al., 2000).

The results of our experiments on the HIFF and H-Trap1 functions are shown in Figure 2. In all three cases the algorithm scales up subquadratically. On the left-hand side of the figure, the graphs in arithmetic scale display the growth of the number of fitness evaluations with respect to the size of the problem for the HIFF and H-Trap1 problems. Results on H-Trap2 were within 8% of the results on H-Trap1 and due to the lack of space we do not present them here (see Pelikan and Goldberg (2001)).

Theory of population sizing and time to convergence for the BOA on separable problems of bounded difficulty (Pelikan et al., 2000) can be used to estimate time to convergence of hierarchical BOA on hierarchical problems. Theory suggests that a single level of the hierarchical problem can be solved in about $O(n^{1.5})$ fitness evaluations. The number of levels in all three hierarchical problems grows as $O(\log n)$. Thus, the overall time to convergence should grow as $O(n^{1.5} \log n)$. The fit is very good and matches also the slopes in the log-log scaled graphs very accurately.

On the right-hand side, the log-log scaled graphs including the slopes between neighboring points are shown. A linear function in this scale is a polynomial of the degree equal to the slope of the curve. To show that the number of fitness evaluations grows at most polynomially with the problem size, the points must lie on a straight line. In our experiments, we see that the slopes in fact *decrease* with the problem size. This is the effect of the logarithm in the expected number of fitness evaluations.

The simple genetic algorithm with fixed crossover is not able to optimize hierarchical functions without making sure that interacting genes are close to each other. Under the assumption of tight linkage, the simple genetic algorithm with good niching should work quite well. The algorithm presented in Watson (2000) is able to solve the HIFF problem even for interacting genes spread throughout the strings. However, Watson's algorithm requires $O(n^2 \log n)$ fitness evaluations for a problem of size n which is more than is required by our algorithm.

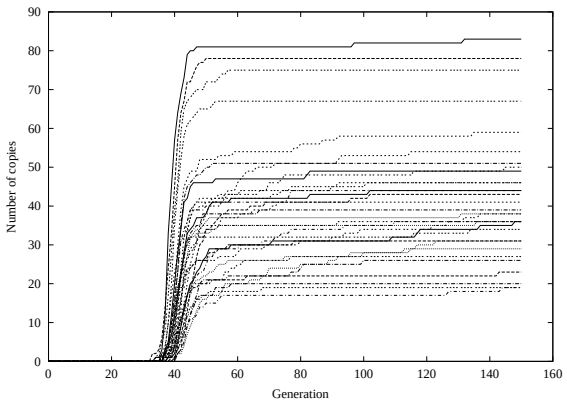
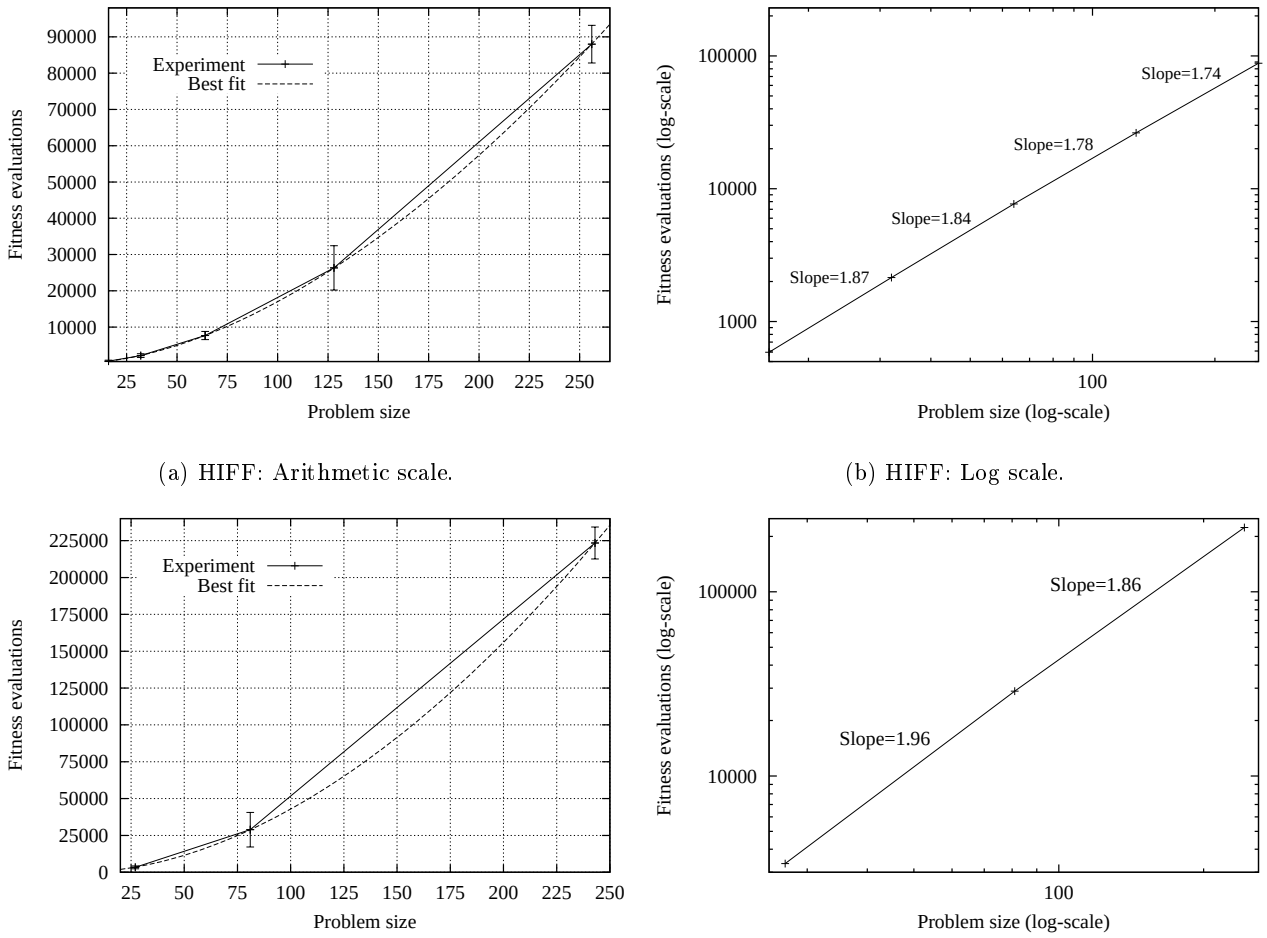


Figure 3: Number of copies of different global optima of the bipolar function. There are 32 optima in this function and all 32 are multiply represented at the end of the run.

To show the ability of hierarchical BOA to discover multiple optima, we also performed a single run on a bipolar function of size $n = 30$ with a sufficiently big population and recorded the number of copies of each global optimum in the population (see Figure 3). We have performed a number of experiments with varying parameters with a very similar result. The algorithm was able to discover and maintain all global optima which soon took over the entire population. However, the optima were not equally distributed, ranging from about 1.27% to about 5.53% of the population. This confirmed the intuition that, unlike fitness sharing, the methods based on crowding are not very sensitive to the fitness values. They are able to maintain a number of alternatives but the total space occupied by each alternative is not proportional to its fitness.

7 CONCLUSIONS

The paper takes another important step toward increasingly competent genetic algorithms by providing an algorithm that is able to solve problems on a single level as well as multiple levels. It emphasizes the importance of solving separable problems on a single level by showing that we need not modify much to successfully move from a single level to hierarchies. To solve



(a) HIFF: Arithmetic scale.

(b) HIFF: Log scale.

(c) H-Trap1: Arithmetic scale.

(d) H-Trap1: Log scale.

Figure 2: Results on the hierarchical functions.

hierarchically decomposable problems quickly, accurately, and reliably, a combination of niching, linkage learning, and efficient representation of partial solutions is necessary.

To learn the linkage, hierarchical BOA uses Bayesian networks to model promising solutions and to generate the new ones. To efficiently represent partial solutions, decision graphs are used to represent local densities in a model. To assure powerful niching, the restricted tournament replacement is used.

Separable deceptive problems of bounded difficulty are extended to multiple levels. The designed hierarchical trap problems that are deceptive on each level are intractable by local search methods and can be used as a benchmark for other optimization algorithms. Hierarchical BOA can solve these problems very efficiently and reliably and it scales up subquadratically with the problem size. Population sizing and convergence theory can be used to approximate the behavior of the algorithm both on single-level and hierarchical problems.

Hierarchical BOA should be applicable to real-world problems without problem specific knowledge ahead of time. This takes us closer to the promised land of robustness, that has long been associated with GAs but rarely delivered.

Acknowledgments

The authors would like to thank Martin Butz, Erick Cantú-Paz, Clarissa Van Hoyweghen, Fernando Lobo, Franz Rothlauf, and Kumara Sastry for many useful discussions and valuable comments.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-94-1-0103, F49620-95-1-0338, F49620-97-1-0050, and F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The US Government

is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Martin Pelikan was partially supported by grant VEGA 1/7654/20 of Slovak Grant Agency. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U. S. Army, or the U.S. Government.

References

Cavicchio, Jr., D. J. (1970). *Adaptive search using simulated evolution.* Doctoral dissertation, University of Michigan, Ann Arbor, MI. (University Microfilms No. 25-0199).

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.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Doctoral dissertation, University of Michigan, Ann Arbor. (University Microfilms No. 76-9381).

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

Deb, K., Horn, J., & Goldberg, D. E. (1992). *Multimodal deceptive functions* (IlliGAL Report No. 92003). Urbana, IL: University of Illinois at Urbana-Champaign.

Friedman, N., & Goldszmidt, M. (1999). Learning Bayesian networks with local structure. In Jordan, M. I. (Ed.), *Graphical models* (1 ed.). (pp. 421–459). Cambridge, MA: MIT Press.

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

Goldberg, D. E. (1997, November). *The design of innovation: Lessons from genetic algorithms.* Unpublished manuscript.

Goldberg, D. E. (1998, June 15). Four keys to understanding building-block difficulty. Presented in Projet FRACTALES Seminar at I.N.R.I.A. Rocquencourt, Le Chesnay, Cedex.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J. (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 41–49). Hillsdale, NJ: Lawrence Erlbaum Associates.

Harik, G. (1994, May). *Finding multiple solutions in problems of bounded difficulty* (IlliGAL Report No. 94002). Urbana, IL: University of Illinois at Urbana-Champaign.

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

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

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

Pelikan, M., & Goldberg, D. E. (2000a). *Genetic algorithms, clustering, and the breaking of symmetry* (IlliGAL Report No. 2000013). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., & Goldberg, D. E. (2000b). *Hierarchical problem solving by the Bayesian optimization algorithm* (IlliGAL Report No. 2000002). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., & Goldberg, D. E. (2001). *Escaping hierarchical traps with competent genetic algorithms* (IlliGAL Report No. 2001003). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1998). *Linkage problem, distribution estimation, and Bayesian networks* (IlliGAL Report No. 98013). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). *Bayesian optimization algorithm, population sizing, and time to convergence* (IlliGAL Report No. 2000001). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., Goldberg, D. E., & Lobo, F. (2000). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications.* In press.

Pelikan, M., Goldberg, D. E., & Sastry, K. (2000). *Bayesian optimization algorithm, decision graphs, and Occam's razor* (IlliGAL Report No. 2000020). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. In Forrest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 38–45). San Mateo, CA: Morgan Kaufmann.

Watson, R. A. (2000). Analysis of recombinative algorithms on a non-separable building-block problem. *Foundations of Genetic Algorithms.* In printing.

Watson, R. A., Hornby, G. S., & Pollack, J. B. (1998). Modeling building-block interdependency. In Eiben, A. E., et al. (Eds.), *Parallel Problem Solving from Nature, PPSN V* (pp. 97–106). Berlin: Springer Verlag.