Parameter-Less Hierarchical BOA

Martin Pelikan and Tz-Kai Lin

Dept. of Math. and Computer Science, 320 CCB
University of Missouri at St. Louis
8001 Natural Bridge Rd., St. Louis, MO 63121
pelikan@cs.umsl.edu
tlkq4@studentmail.umsl.edu

Abstract. The parameter-less hierarchical Bayesian optimization algorithm (hBOA) enables the use of hBOA without the need for tuning parameters for solving each problem instance. There are three crucial parameters in hBOA: (1) the selection pressure, (2) the window size for restricted tournaments, and (3) the population size. Although both the selection pressure and the window size influence hBOA performance, performance should remain low-order polynomial with standard choices of these two parameters. However, there is no standard population size that would work for all problems of interest and the population size must thus be eliminated in a different way. To eliminate the population size, the parameter-less hBOA adopts the population-sizing technique of the parameter-less genetic algorithm. Based on the existing theory, the parameter-less hBOA should be able to solve nearly decomposable and hierarchical problems in quadratic or subquadratic number of function evaluations without the need for setting any parameters whatsoever. A number of experiments are presented to verify scalability of the parameter-less hBOA.

1 Introduction

When the hierarchical Bayesian optimization algorithm (hBOA) was designed [1, 2], it was argued that hBOA can solve difficult nearly decomposable and hierarchical problems without the need for setting any parameters. As a result, to solve a new black-box optimization problem, it should be sufficient to plug the new problem into hBOA, press the start button, and wait until hBOA figures out where the optimum is. It was argued that all hBOA parameters except for the population size can be set to their default values without affecting good scalability of hBOA [3]. However, choosing an adequate population size was argued to be crucial [3] and in all experiments the user was assumed to set the population size optimally to obtain best performance while retaining reliable convergence. That is why the dream of having a fully parameter-less optimizer for the entire class of nearly decomposable and hierarchical problems remained one parameter away from reality.

The purpose of this paper is to propose a fully parameter-less hBOA by implementing the parameter-less population-sizing scheme of the parameter-less

genetic algorithm (GA) [4] in hBOA. The parameter-less hBOA simulates a collection of populations of different sizes, starting with a small base population size. Each next population is twice as large as the previous one. To enable parallel simulation of a number of populations when the number of populations that must be simulated is not known in advance, each population is forced to proceed with at most the same speed as any smaller population, where speed is considered with respect to the number of function evaluations. This ensures that although there is no upper bound on the number of populations simulated in parallel and on their size, the overall computational overhead is still reasonable compared to the case with an optimal population size. In fact, theory exists that shows that the parameter-less population sizing scheme in the parameter-less GA does not increase the number of function evaluations until convergence by more than a logarithmic factor [5]. As a result, hBOA can be expected to perform within a logarithmic factor from the case with the optimal population size. This is verified with a number of experiments on nearly decomposable and hierarchical problems.

The paper starts by discussing probabilistic model-building genetic algorithms (PMBGAs) and the hierarchical BOA (hBOA). Section 3 discusses the parameter-less genetic algorithm, which serves as the primary source of inspiration for designing the parameter-less hBOA. Section 4 describes the parameter-less hBOA. Section 5 describes experiments performed and discusses empirical results. Finally, Section 6 summarizes and concludes the paper.

2   Hierarchical Bayesian Optimization Algorithm (hBOA)

Probabilistic model-building genetic algorithms (PMBGAs) [6,7] replace traditional variation operators of genetic and evolutionary algorithms by a two-step procedure. In the first step, a probabilistic model is built for promising solutions after selection. Next, the probabilistic model is sampled to generate new solutions. By replacing variation operators inspired by genetics with machine learning techniques that allow automatic discovery of problem regularities from populations of promising solutions, PMBGAs provide quick, accurate, and reliable solution to broad classes of difficult problems, many of which are intractable using other optimizers [1,2].

For an overview of PMBGAs, please see references [6] and [7]. PMBGAs are also known as estimation of distribution algorithms (EDAs) [8] and iterated density-estimation algorithms (IDEAs) [9]. The remainder of this section describes the hierarchical Bayesian optimization algorithm, which is one of the most advanced and powerful PMBGAs.

2.1   Hierarchical BOA (hBOA)

The hierarchical Bayesian optimization algorithm (hBOA) [1,2] evolves a population of candidate solutions to the given problem. The first population of candidate solutions is usually generated at random. The population is updated

```
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 local struct. 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;
  };
```

Fig. 1. The pseudocode of the hierarchical BOA (hBOA).

for a number of iterations using two basic operators: (1) selection, and (2) variation. The selection operator selects better solutions at the expense of the worse ones from the current population, yielding a population of promising candidates. The variation operator starts by learning a probabilistic model of the selected solutions. hBOA uses Bayesian networks with local structures [10] to model promising solutions. The variation operator then proceeds by sampling the probabilistic model to generate new solutions, which are incorporated into the original population using the restricted tournament replacement (RTR) [11]. RTR ensures that useful diversity in the population is maintained for long periods of time. The run is terminated when a good enough solution has been found, when the population has not improved for a long time, or when the number of generations has exceeded a given upper bound. Figure 1 shows the pseudocode of hBOA. For a more detailed description of hBOA, see [12].

3   Parameter-Less Genetic Algorithm

The parameter-less genetic algorithm (GA) [4] eliminates the need for setting parameters—such as the population size, selection pressure, crossover rate, and mutation rate—in genetic algorithms. The crossover rate and selection pressure are set to ensure consistent growth of building blocks based on the schema theorem. The mutation rate can be eliminated in a similar manner. The population size is eliminated by simulating a collection of populations of different sizes. In the context of hBOA, eliminating the population size is the most important part of the parameter-less GA, because the selection pressure influences performance of hBOA by only a constant factor and there are no crossover or mutation rates in hBOA.

3.1   Eliminating Selection Pressure and Crossover Rate

The parameter-less GA assumes that selection and recombination are primary search operators. The choice of selection pressure and crossover rate should consider the following facts. Selection must be strong enough to ensure consistent

growth of superior building blocks of the optimum, but it must not be too strong because otherwise diversity in the population might be lost prematurely. On the other hand, the crossover rate must be large enough to ensure sufficient exploration of the search space but if blind crossover operators (e.g., one-point and uniform crossover) are used, each application of crossover can break an important building block and thus crossover must not be applied too frequently.

Additionally, there is a tradeoff between selection and crossover. Greater selection pressures allow greater crossover rates. Analogically, smaller selection rates allow only smaller crossover rates. The tradeoff can be formalized using a simplification of the schema theorem, which claims that the expected number of copies of each partial solution or schema after selection and crossover is given by $s(1 - \epsilon)$, where s characterizes the selection strength as a factor by which the number of best solutions will grow, and ϵ represents disruption of the schema by crossover. By ignoring mutation and making a conservative assumption that crossover always disrupts the schema, it is easy to show that $\epsilon = p_c$ and that setting $s = 4$ and $p_c = 0.5$ ensures net growth of 2 for the schemata contained in the best solution.

The original parameter-less GA did not consider mutation. To incorporate mutation, the probability of disrupting a schema due to mutation would have to be incorporated into ϵ. For a bit-flip mutation and binary strings, where each bit of a solution is flipped with a fixed probability p_m, a bounding case could assume that the schema under consideration spans across the entire solution. In that case, the probability of disrupting the schema due to mutation can be computed as $1 - (1 - p_m)^n$, where n is the total number of bits in the solution.

For interacting variables where traditional variation operators fail, even the parameter-less GA is going to suffer from excessive disruption and ineffective mixing of building blocks, as the negative effects of blind variation cannot be eliminated by tweaking GA parameters, but only by modifying the operators themselves.

3.2   Eliminating Population Size

To eliminate the population size, the parameter-less GA simulates a collection of populations of different size [4]. It is important that for any population size N, there exists a population of size greater or equal than N in the collection. Otherwise, problems that require population size greater or equal than N could not be solved. Consequently, the collection must contain infinitely many populations, the size of which cannot be upper bounded.

The parameter-less GA arranges the collection of populations as a *sequence*. The size of the first population in the sequence is set to a small constant called the base population size. The size of the second population is twice the size of the first population. In general, each next population is twice the size of the previous population and the population size thus grows exponentially starting with the base population size.

Each population in the collection is allowed to run one generation for each k generations of the population twice as large, where $k \geq 2$ is an integer con-

stant. The original parameter-less GA considered $k = 4$. That means, that the smaller population was allowed to proceed at twice the speed of the next larger population, where speed is again measured with respect to fitness evaluations. It can be shown that for any $k \geq 2$, the infinite collection of populations can be simulated tractably without increasing the number of function evaluations until convergence by more than a logarithmic factor with respect to the optimal population size [5].

3.3   How Many Generations for Each Population?

Based on convergence theory for both large and small populations [13,14,15], the number of generations can be assumed to be upper-bounded by a constant that does not depend on the population size. Since small populations process generations faster than larger populations, it seems reasonable to terminate simulation of each population at some point and use computational resources more efficiently.

The parameter-less GA terminates a population in the collection if either of the following criteria is satisfied:

- The population converges and it consists of many copies of a single solution. In this case, it can be expected that no more improvement will take place anymore or that the search will become very inefficient. Clearly, this criterion is not going to have much effect if niching is used.
- A larger population has a greater average fitness. Since larger populations converge generally at most as fast as smaller populations, this situation indicates that the smaller population got stuck in a local optimum and it can thus be terminated.

It is important to note that the logarithmic overhead computed in [5] does not consider the termination criterion and the parameter-less GA should thus perform well even without terminating any populations in the collection.

4   Parameter-Less hBOA

The parameter-less hBOA incorporates the population-sizing technique of the parameter-less GA into hBOA. Since hBOA ensures growth and mixing of important building blocks via learning and sampling a probabilistic model of promising solutions, there is no reason to restrict crossover rate and all offspring can be created by sampling the probabilistic model of promising solutions. Additionally, any selection pressure that favors best candidate solutions can be used without changing scalability of hBOA by more than a constant factor.

The parameter-less hBOA simulates a collection of populations

$$P = \{P_0, P_1, \ldots\}.$$

The size of the first population P_0 is denoted by N_0 and it is called the base population size. The size of the population P_i is denoted by N_i and it can be obtained by multiplying the base population size by a factor of 2^i:

$$N_i = N_0 2^i.$$

For all $i \in \{1, 2, \ldots\}$, one generation of P_i is executed after executing $k \geq 2$ generations of P_{i-1}. Here we use $k = 2$ so that all populations proceed at the same speed with respect to the number of evaluations. Each population is initialized just before its first iteration is executed. The pseudocode that can be used to simulate the collection of populations described here is shown in Figure 2. This implementation is slightly different from the one based on a k-ary counter described in the first parameter-less GA study [4].

```
Parameter-less hBOA
  initialize P[0];
  generation[0]=0;
  max_initialized=0;
  i=0;
  while (not done) {
    simulate one generation of P[i];
    generation[i] = generation[i]+1;
    if (generation[i] mod k = 0) {
      i = i + 1;
      if (i>max_initialized) {
        initialize P[i];
        max_initialized=i;
      }
    }
    else
      i = 0;
  };
```

Fig. 2. The pseudocode of the parameter-less hBOA.

4.1   When to Terminate a Population?

The same termination criteria as in the parameter-less GA can be used in the parameter-less hBOA. However, since hBOA uses niching, no population in the collection can be expected to converge for a long time. Additionally, we terminate each population after it executes for a number of generations equal to the number of bits in the input string. According to our experience, enabling each population to run for more generations does not improve performance further, whereas decreasing the limit on the number of generations might endanger convergence on exponentially scaled and hierarchical problems. Similarly as in the parameter-less GA, in the parameter-less hBOA populations can also be run indefinitely

without increasing the worst-case overhead compared to the case with an optimal population size as predicted by theory.

5   Experiments

This section describes experimental methodology, test problems, and experimental results.

5.1   Experimental Methodology

The parameterless hBOA was applied to artificial hierarchical and nearly decomposable problems and 2D $\pm J$ spin glasses with nearest neighbor interactions and periodic boundary conditions. For artificial problems, problem size was varied to examine scalability of the parameter-less hBOA. The performance of the parameter-less hBOA was compared to that of hBOA with optimal population size. For spin glasses, systems of different size were tested and 100 random instances were examined for each problem size to ensure that the results would provide insight into hBOA performance on a wide range of spin glass instances. To improve hBOA performance on spin glasses, a deterministic local search that flips each bit in a candidate solution until the solution cannot be improved anymore is used to improve each candidate solution before it is evaluated. The local searcher favors the best change at each iteration [12]. In all experiments, the base population size $N_0 = 10$ is used.

For each problem instance and each problem size, the parameter-less hBOA is first run to find the optimum in 100 independent runs and the total number of evaluations in every run is recorded. The average number of function evaluations is then displayed. These results are compared to the results for hBOA with the minimum population size that ensures that 30 independent runs converge to the optimum. The minimum population size was determined using bisection until the width of the resulting interval is at most 10% of the lower bound.

The remainder of this section discusses test problems and experimental results.

5.2   Deceptive Function of Order 3

In the deceptive function of order 3 [16,17], the input string is first partitioned into independent groups of 3 bits each. This partitioning should be unknown to the algorithm, but it should 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. The 3-bit deceptive function is defined as follows:

$$dec_3(u) = \begin{cases} 1 & \text{if } u = 3 \\ 0 & \text{if } u = 2 \\ 0.8 & \text{if } u = 1 \\ 0.9 & \text{if } u = 0 \end{cases}, \tag{1}$$
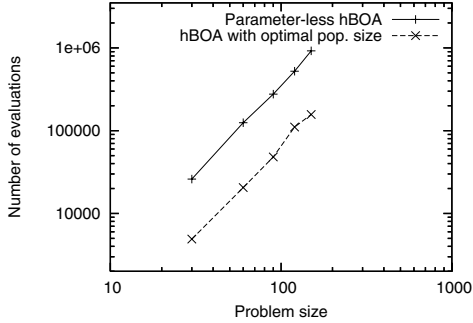
Fig. 3. Parameter-less hBOA and hBOA with the optimal population size on the deceptive function of order 3.

where u is the number of 1s in the input string of 3 bits. The 3-bit deceptive function is fully deceptive [17], which means that variation operators should not break interactions between bits in each group, because all statistics of lower order lead the algorithm away from the optimum. That is why most crossover operators as well as the model in UMDA will fail at solving this problem faster than in exponential number of evaluations, which is just as bad as with random search. Since deceptive functions bound a broad class of decomposable problems, performance of the parameter-less hBOA on this class of problems should indicate what performance can be expected on other decomposable problems.

Figure 3 shows the number of evaluations of the hierarchical BOA with optimal population size determined by the bisection method and the parameter-less hBOA on order-3 deceptive functions of $n = 30$ to $n = 150$ bits. The results indicate that the number of function evaluations for decomposable problems increases only by a near constant factor, and eliminating the parameters of hBOA thus does not affect scalability of hBOA on decomposable problems qualitatively.

5.3   Hierarchical Traps of Order 3

In an order-3 hierarchical trap with L levels, the total number of bits in a candidate solution is assumed to be an integer power of 3, $n = 3^L$. A candidate solution is evaluated on multiple levels and the overall value of the hierarchical trap is computed as the sum of contributions on all levels. On each level, a basis function similar to the deceptive function of order 3 is used:

$$basis(u, f_{lo}, f_{hi}) = \begin{cases} f_{hi} & \text{if } u = 3 \\ f_{lo} - u\frac{f_{lo}}{u-1} & \text{otherwise} \end{cases}, \tag{2}$$

On the lowest level, the candidate solution is partitioned into independent groups of 3 bits each, and each group is evaluated using the basis function with $f_{lo} = f_{hi} = 1$. The contributions of 000 and 111 are thus equally good, and 000 and

111 are superior to other combinations of 3 bits in any group. The contributions of all these groups are added together to form the overall contribution of the first level. Each group is then mapped (or interpreted) into a single symbol on the next level using the following interpretation function:

$$map(X) = \begin{cases} 0 & \text{if } X = 000 \\ 1 & \text{if } X = 111 \\ - & \text{otherwise} \end{cases}, \qquad (3)$$

where X denotes the input 3 bits or symbols, and "$-$" is a null symbol. The second level thus contains $\frac{n}{3}$ symbols from $\{0, 1, -\}$.

Symbols on the second level are also partitioned into independent groups of 3 bits each and each group contributes to the fitness on this level using the same basis function as on the first level, where $f_{hi} = f_{lo} = 1$. However, groups that contain the symbol "$-$" do not contribute to the fitness at all. The overall contribution of the second level is multiplied by 3 and added to the contribution of the first level. Each group is then mapped to the third level using the same interpretation function as was used to map the first level to the second one.

The same principle is used to evaluate and map each higher level except for the top level, which contains 3 symbols (the last 3 symbols are not mapped anymore). The contribution of kth level is always multiplied by 3^{k-1} so that the overall contribution of each level is of the same magnitude. The only difference when evaluating the top level of 3 symbols is that $f_{lo} = 0.9$, whereas $f_{hi} = 1$.

The global optimum of hierarchical traps is in the string of all 1s. However, blocks of 0s and 1s seem to be equally good on each level. Furthermore, any evolutionary algorithm is biased to solutions with many 0s, because the neighborhood of solutions with many 1s is inferior. Since hierarchical traps bound a broad class of hierarchical problems, performance of the parameter-less hBOA on this class problems should indicate what performance can be expected on other hierarchical problems.

Figure 4 shows the number of evaluations of the hierarchical BOA with optimal population size determined by the bisection method and the parameter-less hBOA on hierarchical traps of $n = 27, 81$, and 243 bits. The results indicate that the number of function evaluations for hierarchically decomposable problems increases only by a constant factor, and eliminating the parameters of hBOA does not affect scalability of hBOA on this class of problems qualitatively.

5.4   2D Ising $\pm J$ Spin Glasses

A 2D spin-glass system consists of a regular 2D grid containing n nodes. The edges in the grid connect nearest neighbors. Additionally, edges between the first and the last element in each dimension are added to introduce periodic boundary conditions.

With each edge, there is a real-valued constant associated with it called also a coupling constant. Each spin can obtain two values: $+1$ or -1. Given a set
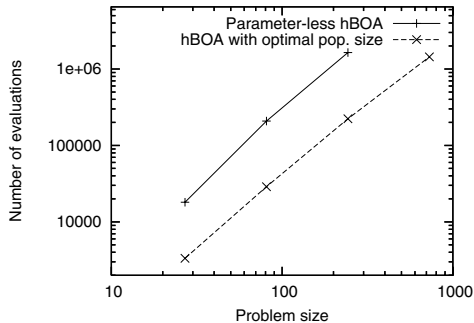
Fig. 4. Parameter-less hBOA and hBOA with the optimal population size on the hierarchical trap.

of coupling constants and spins, the energy of the spin glass system can be computed as

$$E(S) = \sum_{(i,j) \in E} s_i J_{i,j} s_j, \tag{4}$$

where $i \in \{0, 1, \ldots, n-1\}$, $j \in \{0, 1, \ldots, n-1\}$, E is the set of edges in the system, s_i and s_j denote the values of the spins i and j, respectively, and $J_{i,j}$ denotes the coupling constant between the spins i and j.

Here the task is to find the ground state for a 2D spin glass with specified coupling constants, where the ground state is the configuration of spins that minimizes the energy of the system given by Equation 4. Each spin glass configuration is represented by a string of bits where each bit corresponds to one spin: a 0 represents a spin -1, and a 1 represents a spin $+1$. We created 100 random spin glasses of sizes 6×6 (36 spins) to 14×14 (196 spins) by generating coupling constants to be $+1$ or -1 with equal probabilities. Spin glasses with coupling constants restricted to $+1$ and -1 are called $\pm J$ spin glasses.

Figure 5 shows the number of evaluations of the hierarchical BOA with optimal population size determined by the bisection method and the parameter-less hBOA on 2D $\pm J$ spin glasses. The good news is that the number of evaluations until convergence of the parameter-less hBOA still grows as a low-order polynomial with respect to the number of decision variables. The bad news is that, unlike for single-level and hierarchical traps, in this case the parameter-less population sizing does influence scalability of hBOA. More specifically, the order of the polynomial that approximates the number of evaluations increases by about 1.

Clearly, a linear factor by which the number of evaluations increases disagrees with the existing theory, which claims that the factor should be at most logarithmic [5]. Our hypothesis why this happens is that in this case the dynamics of hBOA with local search changes when increasing the population size. Using the hybrid method enables the two searchers—hBOA and the local searcher—
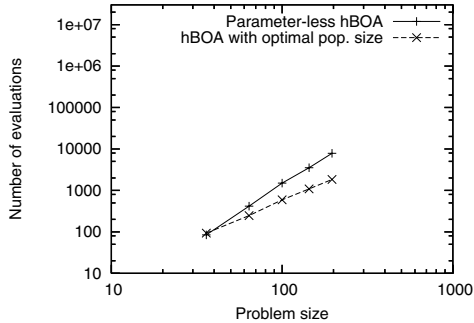
Fig. 5. Parameter-less hBOA and hBOA with the optimal population size on $\pm J$ spin glasses.

to cooperate and solve the problem faster. hBOA allows the local searcher to exit the enormous number of local optima, which make local search by itself intractable. On the other hand, the local searcher decreases the population sizes in hBOA significantly by making the algorithm to focus on regions with local optima. The performance of the hybrid then depends on how well the division of labor between the local and global searcher is done. Increasing the population size in hBOA with local search can influence the division of labor in the hybrid. Since the parameter-less hBOA simulates a number of populations of different sizes, we believe that the division labor is affected and more opportunities are given to hBOA at the expense of the local searcher. We are currently verifying the above hypothesis. The division of labor between local and global searcher can be also tuned by changing the maximum number of generations. However, to ensure that the algorithm will scale also exponentially scaled problems, where the number of generations grows linearly with the size of the problem, the maximum number of generations should not be limited by a function that grows slower than linearly.

6   Summary and Conclusions

This paper described, implemented, and tested the parameter-less hierarchical BOA. The parameter-less hBOA enables the practitioner to simply plug in the problem into hBOA without requiring the practitioner to first estimate an adequate population size or other problem-specific parameters. Despite the parameter-less scheme, low-order polynomial time complexity of hBOA on broad classes of optimization problems is retained. The parameter-less hBOA is thus a true black-box optimization algorithm, which can be applied to hierarchical and nearly decomposable problems without setting any parameters whatsoever. An interesting topic for future work is to develop techniques to improve hBOA performance via automatic tuning of other parameters, such as the maximum order of interactions in the probabilistic model or the window size in RTR.

References

1. Pelikan, M., Goldberg, D.E.: Escaping hierarchical traps with competent genetic algorithms. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 511–518
2. Pelikan, M., Goldberg, D.E.: A hierarchy machine: Learning to optimize from nature and humans. Complexity **8** (2003)
3. Pelikan, M., Sastry, K., Goldberg, D.E.: Scalability of the Bayesian optimization algorithm. International Journal of Approximate Reasoning **31** (2002) 221–258
4. Harik, G., Lobo, F.: A parameter-less genetic algorithm. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99) **I** (1999) 258–265
5. Pelikan, M., Lobo, F.G.: Parameter-less genetic algorithm: A worst-case time and space complexity analysis. IlliGAL Report No. 99014, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
6. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications **21** (2002) 5–20
7. Larrañaga, P., Lozano, J.A., eds.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Boston, MA (2002)
8. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. Parallel Problem Solving from Nature (1996) 178–187
9. Bosman, P.A.N., Thierens, D.: Continuous iterated density estimation evolutionary algorithms within the IDEA framework. Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000) (2000) 197–200
10. Chickering, D.M., Heckerman, D., Meek, C.: A Bayesian approach to learning Bayesian networks with local structure. Technical Report MSR-TR-97-07, Microsoft Research, Redmond, WA (1997)
11. Harik, G.R.: Finding multimodal solutions using restricted tournament selection. Proceedings of the International Conference on Genetic Algorithms (ICGA-95) (1995) 24–31
12. Pelikan, M.: Bayesian optimization algorithm: From single level to hierarchy. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL (2002)
13. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. Evolutionary Computation **1** (1993) 25–49
14. Thierens, D., Goldberg, D.: Convergence models of genetic algorithm selection schemes. Parallel Problem Solving from Nature (1994) 116–121
15. Ceroni, A., Pelikan, M., Goldberg, D.E.: Convergence-time models for the simple genetic algorithm with finite population. IlliGAL Report No. 2001028, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2001)
16. Ackley, D.H.: An empirical study of bit vector function optimization. Genetic Algorithms and Simulated Annealing (1987) 170–204
17. Deb, K., Goldberg, D.E.: Sufficient conditions for deceptive and easy binary functions. Annals of Mathematics and Artificial Intelligence **10** (1994) 385–408