Practical, Robust, and Scalable Black-Box Optimization with BOA and hBOA

Martin Pelikan

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

Abstract. To design practical black-box optimizers, one of the primary goals is to minimize the amount of work that must be done by the user while ensuring that a high-quality solution will be found quickly and reliably. This paper shows that probabilistic model-building genetic algorithms (PMBGAs) provide a great framework for designing practical and powerful black-box optimizers. The paper focuses on two algorithms that are among the most powerful PMBGAs: The Bayesian optimization algorithm (BOA) and the hierarchical BOA (hBOA).

1 Introduction

Design of practical, robust, and scalable black-box optimization algorithms is an important challenge. There are several design issues that are of primary importance. First of all, the optimization algorithm should not require the user to modify representation of candidate solutions much or to map the original optimization problem to yet another problem of different structure. Instead, the optimization algorithm itself should be able to manipulate representation or map the problem if necessary. Second, one should minimize the number of parameters that the practitioner must set to ensure successful search for the optimum (e.g. thresholds, step size, or the number of iterations). Third, it should be possible to incorporate all tools available to further enhance efficiency of the optimizer (e.g. heuristics, constraints, prior knowledge). Fourth, the optimizer should be robust and it should be able to solve broad classes of problems, so that the practitioner would not have to consider hundreds optimizers for each new optimization problem but he should get satisfactory results by choosing from a small number of robust optimizers. The optimization method should be able to deal with multiple objectives and noise in evaluation. Finally, scalability is important so that the results on toy problems are relevant also for solving more challenging real-world problems.

The purpose of this paper is to show that most of these important issues can be approached with great success using probabilistic model-building genetic algorithms (PMBGAs). The paper focuses on two of the most powerful PMBGAs: The Bayesian optimization algorithm (BOA) [1] and the hierarchical

BOA (hBOA) [2, 3], and discusses features that make these algorithms especially useful for optimization practitioners. In the remainder of this paper we assume that candidate solutions are represented by fixed-length binary strings, but other finite alphabets can be used for each position in solution strings in a straightforward manner.

The paper starts with a brief description of PMBGAs, BOA, and hBOA. Section 3 discusses problems that can be solved using BOA and hBOA and presents main results of BOA and hBOA scalability theory. Section 4 focuses on methods for enhancing efficiency of BOA and hBOA, including parallelization, fitness estimation, prior knowledge utilization, and hybridization. Section 5 discusses how parameters can be eliminated from BOA and hBOA. Finally, Section 6 summarizes and concludes the paper.

2 Hierarchical Bayesian optimization algorithm (hBOA)

Probabilistic model-building genetic algorithms (PMBGAs) [4, 5] 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 [2, 3].

For an overview of PMBGAs, please see references [4] and [5]. PMBGAs are also known as estimation of distribution algorithms (EDAs) [6] and iterated density-estimation algorithms (IDEAs) [7]. The remainder of this section describes the Bayesian optimization algorithm (BOA) and its hierarchical extension, the hierarchical BOA (hBOA).

2.1 Bayesian optimization algorithm (BOA)

The Bayesian optimization algorithm (BOA) [1] 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 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. BOA uses Bayesian networks [8] to model promising solutions. The variation operator then proceeds by sampling the probabilistic model to generate new solutions, which then replace the original population of candidate solutions or its part.

The run is terminated when termination criteria are satisfied; for example, the run can be 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.

```
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).

2.2 Hierarchical BOA (hBOA)

The hierarchical Bayesian optimization algorithm (hBOA) [2, 3] extends BOA by using Bayesian networks with local structures [8] to model promising solutions and by using the restricted tournament replacement (RTR) [9] to incorporate new solutions into the original population. Local structures in Bayesian networks ensure efficient representation of high order interactions, whereas RTR ensures that useful diversity in the population is maintained for long periods of time. Figure 1 shows the pseudocode of hBOA. For a more detailed description of hBOA, see [10].

The following section discusses the classes of problems that can be solved by BOA and hBOA and it overviews basic theoretical and empirical results.

3 Scalability of BOA and hBOA

This section discusses the class of problems that hBOA can solve, and it summarizes important theoretical and empirical results related to scalability of hBOA.

3.1 Decomposition in optimization

From no free lunch theorem for optimization [11], it is known that all optimization algorithms perform the same if their performance is averaged over all possible optimization problems. However, it is a different story when we look at algorithm performance for some more specific classes of problems, such as linear problems (e.g. onemax) or separable problems of bounded difficulty [12]. So before talking about scalability and efficient performance, one must answer the following important question: "What problems do we want a particular algorithm to solve?"

BOA attempts to solve problems that can be decomposed into subproblems of bounded order on a single level. hBOA extends the applicability of BOA to problems that can be decomposed over multiple levels of difficulty into a hierarchy. The reason for using decomposability as the basic concept for robust and scalable solution to broad classes of problems is that many complex real-world

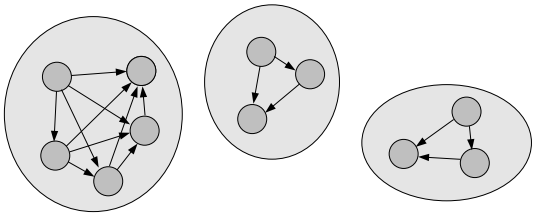


Fig. 2. An illustrative example of problem decomposition of a problem with 11 variables into 3 subproblems. Edges between the variables denote connections in a Bayesian network.

systems can be decomposed into simpler subsystems so that interactions within each subsystem are of much higher magnitude than the interactions between the subsystems. Decomposition is one of the primary concepts used by humans to build models of complex real-world systems, get to understand these systems, and design artificial complex systems [13, 12]. That is why decomposition as a tool for cracking tough optimization problems holds a big promise.

To use decomposition in optimization, genetic and evolutionary algorithms represent a good candidate, because they allow a practical and flexible population-based search where most variation operators are based on problem decomposition. However, if the variation operator exploits wrong decomposition, evolutionary algorithms may take exponential time to solve the problem [14, 15]. That is why it is necessary that both (1) the problem is decomposed properly, and (2) the learned decomposition is used well. PMBGAs address both these issues by using methods of machine learning and statistics as is discussed next.

3.2 Solving decomposable problems using BOA and hBOA

In BOA and hBOA, identifying important subproblems corresponds to learning a Bayesian network [16, 17] that captures the structure of the problem. Loosely said, a good network should connect variables within each subproblem and it should have no connections between the variables in different subproblems as is illustrated in Figure 2. For more complex problems where subproblems interact in some way, the groups of variables corresponding to different subproblems can be connected as well. For an exact definition of a proper distribution factorization for the use with decomposable functions, see [18]. Adequate population sizing ensures that there exist a reasonable number of copies of the solution to each subproblem [19]. Solutions to the different subproblems are then juxtaposed by sampling the Bayesian network.

Decomposition can be used on one or multiple levels. To juxtapose solutions on several levels of difficulty and enable the algorithm to exploit hierarchical decomposition, replacing traditional variation of genetic algorithms by building and sampling Bayesian networks is insufficient and additional features must be incorporated as well [2, 3]. First of all, noninferior partial solutions must be preserved to enable that there is enough material to combine solutions on the next

higher level. Useful diversity preservation thus becomes important. Second, an algorithm must be capable of working with large partial solutions and expressing dependencies between and within these larger partial solutions. By using RTR for diversity preservation and local structures in Bayesian networks for efficient representation of dependencies of large order, hBOA becomes capable of learning and exploiting decomposition over multiple levels of difficulty. For more details on hierarchical optimization with hBOA, please see [2, 3].

To enable solution to challenging optimization problems, it is necessary to ensure that time complexity of these two algorithms grows with a low-order polynomial with respect to the number of variables in the problem (string positions). There are two basic approaches to showing that the optimizer scales up well on the target class of problems:

- Design artificial problems to test the algorithm on the boundary of its design envelope as is common in engineering design. The problems should challenge the optimizer along different dimensions of problem difficulty that are characteristic for the target class of problems (e.g., interactions between decision variables, external noise, and scaling).
- Develop theory that captures dynamics of the optimization algorithm on the target class of problems.

The remainder of this section overviews important results in the above two areas.

3.3 Scalability of BOA

There are three primary sources of computational complexity in BOA and hBOA:

(1) evaluation of candidate solutions,
(2) model building and sampling, and
(3) restricted tournament replacement (RTR).

The overhead related to the remaining operation of BOA and hBOA is negligible compared to the above two sources of complexity. Additionally, although the asymptotic computational complexity of RTR can be the same as that for model building and sampling, the actual time spent in this operator is usually orders of magnitude smaller than that spent in model building. For some problems, where the number of decision variables to optimize is small but each evaluation takes long time, evaluation of candidate solutions is the bottleneck. For other problems, where each evaluation can be done quickly but the number of bits is large, model building becomes the bottleneck.

All the aforementioned sources of complexity depend on the same three factors:

(1) G, the number of iterations (generations) until convergence,
(2) N, the population size, and
(3) k, the order of statistics necessary to solve the problem.

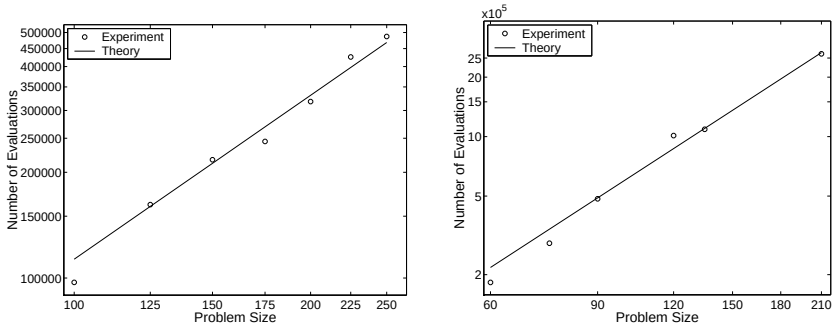
For good scalability it is necessary that k is bounded by a constant or that it grows at most logarithmically with the size of the problem, because the population size grows faster than proportionally to 2^k due to the initial supply [19]. Nonetheless, there is nothing BOA and hBOA can do to influence k, which is directly related to the definition and complexity of the problem. Here we focus on classes of problems with k upper-bounded by a constant, called also problems decomposable problems of bounded difficulty [12].

For any constant upper bound on k, it is important to make sure that both the population size and the number of iterations grow with a low-order polynomial with respect to the number of decision variables in the problem. Next, we review theory for population sizing and the number of generations until convergence applicable to BOA. These are then adopted to hBOA. These results are then verified with a number of experiments on artificial problems designed to test BOA and hBOA on the boundary of their design envelopes.

Population sizing in BOA There are 4 facets to consider to estimate an adequate population size, N, for hBOA [12, 10]:

(1) Initial supply of building blocks [19]. To ensure that BOA finds the optimum, it is necessary that each subproblem in a proper problem decomposition receives a sufficient number of copies of all possible instances that the variables included in this subproblem can obtain. According to this facet, $N \propto 2^k \log n$.
(2) Delayed supply of building blocks [20]. If subproblems are scaled differently, they converge in a sequential manner where at each iteration, only one or several subproblems influence selection. The population size must be large enough to ensure that supply of alternative instances for subproblems that converge later in the run is not eliminated due to the stochastic nature of search operators in BOA. According to this facet, $N = O(n)$.
(3) Decision making between competing building blocks [21]. For each subproblem, decisions between alternative instantiations of this subproblem are done statistically across the entire population. Although the influence of other positions should average out over the population, the population must be large enough to ensure that good instantiations of variables in each subproblem are indeed going to increase in proportion. According to this facet, $N \propto \sqrt{n} \log n$.
(4) Building an adequate model [22]. It is necessary that the population is large enough to be able to learn an adequate Bayesian network. That means that (1) the dependencies must be identified properly, and (2) the parameters of the resulting Bayesian network structure must be estimated accurately. According to this facet, $N = O(n^{1.05})$.

The first three factors are adopted from GA theory [12], whereas the last factor is related to building a Bayesian network that captures appropriate problem decomposition. The last factor is related directly to BOA and the necessity to use an appropriate problem decomposition. Putting all factors together results in an upper bound on the population size of $O(n^{1.05})$.



(a) Uniform scaling (b) Exponential scaling

Fig. 3. Experimental verification of BOA scalability theory

Number of iterations until convergence To estimate the number of generations until convergence, two extreme cases can be considered. For uniformly scaled problems (all subproblems are scaled similarly), the number of generations until convergence grows as $O(\sqrt{n})$ [6]. This model captures population dynamics for a Bayesian network with no edges (probability vector), onemax (task is to maximize the number of ones), and infinite population. Although these assumptions are usually not satisfied in practice, the actual dynamics for adequate population sizes is very close to this estimate assuming that problem can indeed be decomposed into subproblems of order k and that an adequate population size is used. For small population sizes the number of generations can grow faster, but it will never grow faster than linearly with the number of decision variables in the problem [23]. For exponentially scaled subproblems, subproblems converge in a sequential fashion and thus the number of generations until convergence can be upper bounded by $O(n)$.

Experimental verification Given the above theory, the number of evaluations should grow between $O(n^{1.55})$ to $O(n^2)$, depending on scaling of subproblems. Figure 3 (from [10]) verifies this result for concatenated trap functions of order 5, which are created by decomposing the problem into disjoint groups of 5 bits and applying a trap function to each subproblem. In all experiments, an average number of evaluations is computed for 30 independent runs, where the population size is set to the minimum population size ensuring that 30 independent runs converge to the optimum. The figure also shows an exponentially scaled problem where subproblems of order 3 are considered where each subproblem uses a deceptive function of order 3.

The quadratic or subquadratic growth of the number of evaluations independently of k is qualitatively better than the growth proportional to n^k for algorithms based on local variation operators [24], and the exponential growth for genetic algorithms without any additional assumptions about problem encoding [15, 12].

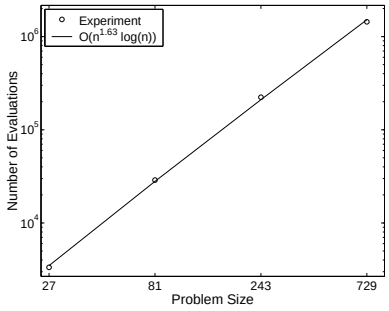


Fig. 4. Experimental verification of hBOA scalability theory.

3.4 Scalability of hBOA

hBOA must solve the problem on several levels. The number of evaluations for BOA should thus be multiplied by the number of levels. Figure 4 shows the number of evaluations for hBOA on a hierarchical trap problem, where the number of levels grows as a logarithm of problem size. The complexity should thus be bounded by $O(n^{1.55} \log n)$. The figure confirms this result. For hierarchical traps, both local search operators as well as traditional genetic algorithms scale exponentially with the number of variables in the problem.

4 Efficiency enhancement of BOA and hBOA

Solving nearly decomposable and hierarchical problems in quadratic or subquadratic time ensures good scalability, but even quadratic performance can be prohibitive if there are thousands or tens of thousands of variables, or if one evaluation takes tens of seconds, minutes, or even hours. There are several ways in which performance of BOA and hBOA can be improved. The following list extends the IlliGAL efficiency enhancement techniques decomposition [12] to incorporate new ways of enhancing efficiency of BOA and hBOA:

1. Parallelization,
2. hybridization,
3. time continuation,
4. fitness evaluation relaxation,
5. prior knowledge utilization,
6. incremental and sporadic model building, and
7. learning from problem-specific experience.

The following list overviews important results in several of the above techniques for efficiency enhancement:

Parallelization. There are two primary sources of computational complexity in BOA and hBOA. The first is the model building and sampling, which can be computationally expensive if many variables are included in the model.

Promising results were reported for parallelization of the model building and sampling in BOA and hBOA [25–27]. The second source is evaluation of candidate solutions. Parallelization of evaluation can be done similarly as in conventional genetic algorithms [28]. Both the model building and evaluation of candidate solutions can thus be distributed among multiple processors.

Hybridization. Promising results obtained with a hybrid method consisting of hBOA and a simple deterministic local searcher were reported for the Ising spin glass problem [29], which is a class of highly multimodal optimization problems with high order of interactions between problem variables. The results indicated that using local search can significantly reduce population size. A simplex method was successfully incorporated into ECGA for silicon cluster optimization [30], where the number of evaluations until convergence was significantly decreased when using the hybrid algorithm. Finally, LFDA (also a PMBGA based on Bayesian network models) for graph partitioning was improved by incorporating Kernighan-Lin algorithm to improve solutions locally [31]. To summarize, incorporating a local searcher into BOA and hBOA is as straightforward as for any other genetic and evolutionary algorithm, and it appears to be beneficial especially for difficult and highly multimodal problems.

Fitness evaluation relaxation. Several approaches were presented to extending the probabilistic model in PMBGAs by incorporating statistics about the fitness [32–34]. Modeling fitness can be especially useful for problems with enormously expensive fitness evaluation (e.g. problems that involve a finite element analysis or a stochastic simulation), where a portion of fitness evaluations can be replaced with a sample from the model. Speed-ups of over 30 (with respect to the number of fitness evaluations) were reported for BOA on decomposable problems of bounded difficulty [34].

Prior knowledge utilization. There are several ways in which prior information about the problem can be incorporated into PMBGAs. First, model building can be modified to bias the search toward models of particular form. Second, promising candidate solutions or partial candidate solutions can be incorporated into the population.

For example, Schwarz and Ocenasek [35] bias the initial population of BOA on graph partitioning using a simple local searcher. Additionally, they used edges in the input graph to bias model building so that dependencies that agree with the edges in the input graph were favored. Results indicated that using prior knowledge improves performance of BOA. Prior knowledge was also used for efficient cluster optimization in ECGA [30], where the initial population was biased to optimal clusters of smaller size. Finally, prior knowledge was used to restrict the model of LFDA to contain only dependencies that agree with the edges of the input graph in graph partitioning [31]. In all cases, prior knowledge improved performance of BOA and hBOA, although no results were reported that would indicate that eliminating prior knowledge would lead to a qualitative decrease in performance of PMBGAs (by more than a constant factor).

5 Eliminating parameters of hBOA

To ensure that BOA or hBOA finds an optimum accurately and reliably, it is necessary that its parameters are set adequately. Nonetheless, there is only one crucial parameter—population size—because all other parameters influence convergence only by a constant factor. Even the population size can be eliminated using the population sizing scheme of the parameter-less genetic algorithm [36]. Using the aforementioned population sizing approach, a series of populations of different sizes are evolved in parallel. The approach ensures that both the optimum will be found reliably and that the overhead with respect to the number of function evaluations will be of at most a logarithmic factor [37]; initial results with a fully parameter-less hBOA confirm this [38].

As a result, BOA and hBOA can be used without setting any parameters, and they should still ensure low-order polynomial scalability on the class of nearly decomposable problems of bounded difficulty.

6 Summary and conclusions

This paper argued that BOA and hBOA provide the practitioner with a powerful and flexible tool for solving difficult optimization problems that can be applied without much prior knowledge about the problem or the algorithms themselves. BOA and hBOA solve a broad class of nearly decomposable and hierarchical problems in a scalable manner, without the necessity to set any parameters. A number of methods for enhancing efficiency of BOA and hBOA are available to further improve performance of these algorithms.

Results overviewed in this paper show that incorporating advanced machine learning techniques, such as methods for learning and sampling Bayesian networks, is proving to be one of the most promising and dynamic areas of genetic and evolutionary computation but also computational optimization at large.

Acknowledgments

Martin Pelikan was supported by the Research Award at the University of Missouri at St. Louis and the Research Board at the University of Missouri at Columbia.

References

1. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99) **I** (1999) 525–532 Also IlliGAL Report No. 99003.
2. 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 Also IlliGAL Report No. 2000020.
3. Pelikan, M., Goldberg, D.E.: A hierarchy machine: Learning to optimize from nature and humans. Complexity **8** (2003) 36–45

4. 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 Also IlliGAL Report No. 99018.
5. Larrañaga, P., Lozano, J.A., eds.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Boston, MA (2002)
6. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. Parallel Problem Solving from Nature (1996) 178–187
7. 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
8. 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)
9. Harik, G.R.: Finding multimodal solutions using restricted tournament selection. Proceedings of the International Conference on Genetic Algorithms (ICGA-95) (1995) 24–31
10. Pelikan, M.: Bayesian optimization algorithm: From single level to hierarchy. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL (2002) Also IlliGAL Report No. 2002023.
11. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67–82
12. Goldberg, D.E.: The design of innovation: Lessons from and for competent genetic algorithms. Volume 7 of Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers (2002)
13. Simon, H.A.: The Sciences of the Artificial. The MIT Press, Cambridge, MA (1968)
14. Thierens, D., Goldberg, D.E.: Mixing in genetic algorithms. Proceedings of the International Conference on Genetic Algorithms (ICGA-93) (1993) 38–45
15. Thierens, D.: Analysis and design of genetic algorithms. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1995)
16. Buntine, W.L.: Theory refinement of Bayesian networks. Proceedings of the Uncertainty in Artificial Intelligence (UAI-91) (1991) 52–60
17. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research, Redmond, WA (1994)
18. Mühlenbein, H., Mahnig, T., Rodriguez, A.O.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics **5** (1999) 215–247
19. Goldberg, D.E., Sastry, K., Latoza, T.: On the supply of building blocks. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 336–342 Also IlliGAL Report No. 2001015.
20. Thierens, D., Goldberg, D.E., Pereira, A.G.: Domino convergence, drift, and the temporal-salience structure of problems. Proceedings of the International Conference on Evolutionary Computation (ICEC-98) (1998) 535–540
21. Harik, G.R., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. Proceedings of the International Conference on Evolutionary Computation (ICEC-97) (1997) 7–12
22. Pelikan, M., Sastry, K., Goldberg, D.E.: Scalability of the Bayesian optimization algorithm. International Journal of Approximate Reasoning **31** (2002) 221–258

23. 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)
24. Mühlenbein, H.: How genetic algorithms really work: I.Mutation and Hillclimbing. Parallel Problem Solving from Nature (1992) 15–25
25. Ocenasek, J., Schwarz, J.: The parallel Bayesian optimization algorithm. In: Proceedings of the European Symposium on Computational Inteligence, Physica-Verlag (2000) 61–67
26. Lozano, J.A., Sagarna, R., Larrañaga, P.: Parallel estimation of distribution algorithms. In: Estimation of distribution algorithm: A new tool for evolutionary computation. Kluwer (2001)
27. Ocenasek, J., Schwarz, J., Pelikan, M.: Design of multithreaded estimation of distribution algorithms. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003) (2003) 1247–1258
28. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer, Boston, MA (2000)
29. Pelikan, M., Ocenasek, J., Trebst, S., Troyer, M., Alet, F.: Computational complexity and simulation of rare events of Ising spin glasses. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) (2004)
30. Sastry, K.: Efficient atomic cluster optimization using a hybrid extended compact genetic algorithm with seeded population. IlliGAL Report No. 2001018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2001)
31. Mühlenbein, H., Mahnig, T.: Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. Journal of Approximate Reasoning **31** (2002) 157–192
32. Sastry, K., Goldberg, D.E., Pelikan, M.: Don't evaluate, inherit. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 551–558 Also IlliGAL Report No. 2001013.
33. Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation:. IlliGAL Report No. 2004010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL (2004)
34. Pelikan, M., Sastry, K.: Fitness inheritance in the Bayesian optimization algorithm. IlliGAL Report No. 2004009, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL (2004)
35. Schwarz, J., Ocenasek, J.: A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning. In: Proceedings of the Fourth Joint Conference on Knowledge-Based Software Engineering, Brno, Czech Republic, IO Press (2000) 51–58
36. Harik, G., Lobo, F.: A parameter-less genetic algorithm. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99) **I** (1999) 258–265
37. 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)
38. Pelikan, M., Lin, T.K.: Parameter-less hierarchical BOA. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) (2004)