Evolutionary Algorithms + Graphical Models = Scalable Black-Box Optimization

Martin Pelikan,
Kumara Sastry,
and David E. Goldberg

IlliGAL Report No. 2001029
November 2001

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

Evolutionary Algorithms + Graphical Models = Scalable Black-Box Optimization

Martin Pelikan, Kumara Sastry, and David E. Goldberg
Illinois Genetic Algorithms Laboratory
104 S. Mathews Avenue, Urbana, IL 61801
University of Illinois at Urbana-Champaign
Phone/FAX: (217) 333-2346, (217) 244-5705
{pelikan,sastry,deg}@illigal.ge.uiuc.edu

Abstract

To solve a wide range of different problems, the research in black-box optimization faces several important challenges. One of the most important challenges is the design of methods capable of automatically discovering the regularities in the problem and utilizing these to ensure efficient and reliable search. This paper discusses the Bayesian optimization algorithm (BOA) that uses Bayesian networks to model promising solutions and guide exploration of the search space. Using Bayesian networks in combination with population-based genetic and evolutionary search allows the algorithm to discover and utilize regularities in the form of a problem decomposition. The paper analyzes the applicability of the methods for learning Bayesian networks in context of genetic and evolutionary search. In particular, the population sizing ensuring that BOA learns a proper decomposition of the problem is analyzed. The paper concludes that the combination of the two approaches in BOA yields a robust, efficient, and accurate search.

1 Introduction

One of the most important challenges of black-box optimization is the design of methods capable of automatic discovery of the regularities in the problem landscape. A proper utilization of the discovered regularities ensures efficient and reliable search for the optimum. Genetic and evolutionary computation (Holland, 1975; Goldberg, 1989a; Rechenberg, 1973) offers a class of methods that are capable of exploiting regularities in the form of a decomposition of the problem by combining bits and pieces of promising solutions found so far and perturbing the solutions slightly.

Many important real-world problems can be efficiently and reliably optimized by using the decompositional bias. However, most of the commonly used genetic and evolutionary algorithms use simple operators that are incapable of *learning* the regularities in the problem. Moreover, many operators are simply not expressive enough to solve difficult real-world problems efficiently and reliably. Much effort has been put into designing competent genetic and

evolutionary algorithms capable of both discovery and utilization of the problem decomposition. One of the most promising lines of research in this area focuses on using probabilistic models to model promising solutions found so far and guide further exploration of the search space (Pelikan, Goldberg, & Lobo, 2002).

This paper discusses the Bayesian optimization algorithm that uses methods for learning and simulation of Bayesian networks to learn and exploit a proper decomposition of the problem defined over a fixed number of discrete variables over some finite domain. We provide the basic definition of the algorithm and the semantics of Bayesian networks within the framework of black-box optimization. The paper addresses the issue of learning a proper decomposition and analyzes the scalability of the algorithm. The results indicate that there is a large range of parameters that ensure efficient and reliable search. Additionally, it is shown that by assuming a bounded difficulty of the subproblems in the decomposition, the number of evaluations of the objective function required by the Bayesian optimization algorithm grows subquadratically or quadratically with respect to the number of variables in the problem.

The paper starts by motivating the use of probabilistic models to guide the search in genetic and evolutionary computation. A brief overview of what we call probabilistic model-building genetic algorithms is given. Section 3 describes the Bayesian optimization algorithm and provides basic background of methods for learning and sampling Bayesian networks. Additionally, the section discusses the semantics of Bayesian networks when used in combination with other operators of genetic and evolutionary algorithms. Section 4 theoretically analyzes the bias of the algorithms for constructing the model based on the Bayesian information criterion toward models with interactions and its impact on the scalability of the Bayesian optimization algorithm and other approaches based on a similar principle. Additionally, the section presents the results of the simulation for the K2 metric based on Bayesian statistics and compares the behavior of the K2 metric to the Bayesian information criterion. Section 5 puts all the pieces of the scalability puzzle together and discusses the result. Finally, Section 6 concludes the paper.

The primary focus of this paper is on the methods for optimizing discrete problems where each variable can obtain one out of a finite set of possible values. In the theoretical part of the paper, only binary variables that can obtain either 0 or 1 are considered. However, many ideas should extend to other domains such as vectors of real numbers.

2 Probabilistic Model-Building Genetic Algorithms

Genetic and evolutionary algorithms (Holland, 1975; Goldberg, 1989a; Rechenberg, 1973) evolve a population of candidate solutions to a given problem by repeatedly applying two key operators: (1) selection, and (2) variation. Selection operator derives an inspiration from Darwin's survival of the fittest by making more copies of good solutions and eliminating the bad solutions. By doing this, the search is biased toward the regions that look promising based on the current state of knowledge. The quality of each solution is measured by the fitness function which assigns each solution a particular value determining the quality of the solution. The task is to find a solution with a maximum fitness.

Variation operators ensure exploration of new regions in the search space. There are two

commonly used types of variation operators: (1) recombination/crossover and (2) mutation. Recombination combines bits and pieces of promising solutions found so far in order to create their novel combinations. Mutation perturbs the solutions slightly in order to explore their close neighborhood. This paper focuses on genetic algorithms (GAs) (Holland, 1975; Goldberg, 1989a), where recombination is the major operator driving the search. Hybrid methods combining genetic algorithms with local search can be used to further refine the candidate solutions in each generation or in the final stage of the run and increase the efficiency of the search ().

2.1 Decomposition in Optimization and Graphical Models

If the problem is decomposable into subproblems of bounded difficulty and proper selection and recombination operators are used, genetic algorithms can find the optimum in the time that grows only polynomially with the number of variables although the number of possible solutions is at least exponential in the number of variables. Of course, not all problems satisfy this property. A good example of such a problem is the needle in the haystack where all solutions are equal except for the optimum that outperforms any other solution. In the needle in the haystack, one simply cannot find the optimum without enumerating one solution after another until the optimum is found. It is also quite straightforward to create problems that deceive a particular optimization algorithm using the bias used by the algorithm against it (see Deb and Goldberg (1991) and Deb, Horn, and Goldberg (1992) for functions deceiving the algorithms based on the decompositional bias). However, it is known that many difficult problems (including many instances of NP-complete problems) can be efficiently solved by decomposition.

A lot of work has been done on the limits of applying decomposability in optimization and other domains. The key question is whether we can design methods that can solve decomposable problems efficiently and reliably without requiring an intensive problem analysis. This involves

1. learning the structure of the problem in the form of its decomposition, and
2. effective utilization of the decomposition.

In machine learning, the field of graphical models has been developed to provide a unified framework for utilizing decomposition in various problem domains involving multivariate statistical analysis, estimation of probability distributions, and probabilistic inference. Many good results have been produced in learning graphical models given training data, sampling graphical models to generate future instances of time series, or fast probabilistic inference in diagnostic systems. Some of the real-world applications include diagnosis, forecasting, automated vision, sensor fusion, and control (Heckerman, Mamdani, & Wellman, 1995). Because graphical models allow the use of decomposition in a quite general fashion, they might be a good candidate for approaching the problem of discovering the decompositional regularities in the search space and utilizing these.

Probabilistic model-building genetic algorithms (PMBGAs) build on this idea and replace traditionally used genetic operators of genetic algorithms with the methods for building and

using graphical probabilistic models. A model of the promising solutions found so far is first constructed. The built model is then sampled to generate new solutions. The dependencies encoded by the built model allow treating some groups of variables as an intact component of the solution. The independencies allow a proper decomposition of the problem that is necessary for efficient optimization.

2.2 General PMBGA

Starting from a random initial population of solutions, each iteration of PMBGAs consists of four steps. First, promising solutions are selected by using some of the selection operators used in traditional genetic algorithms. In the second step, a distribution of the promising solutions is estimated. In the third step, new solutions are generated according to the estimated distribution. Finally, the new solutions replace some of the solutions in the original population or all of them. Slight variations from the above scheme can be found in practice, but the basic idea of the algorithm remains the same.

A number of different PMBGAs has been proposed over the years. In this paper, we focus on PMBGAs for optimization of discrete problems where each candidate solution contains a fixed number of variables, each of which can obtain a value from a finite set. Unless mentioned otherwise, we are going to discuss such problems. The remainder of this section overviews PMBGAs for discrete problems. Most of the discussed methods have been originally proposed for binary variables, although a generalization to any finite alphabet is straightforward. For more information on the application of PMBGAs to the domain of continuous problems, see Bosman and Thierens (2000), Larranaga, Etxeberria, Lozano, and Pena (2000), and Pelikan, Goldberg, and Tsutsui (2000). See Pelikan, Goldberg, and Lobo (2002) for a more complete survey of PMBGAs.

2.3 From Univariate Distributions to Bayesian Networks

First PMBGAs were based on the assumption that all variables in the problem are independent. The resulting model was thus composed of a product of local models for each variable in the problem. Population-based incremental learning (PBIL) algorithm (Baluja, 1994), compact genetic algorithm (cGA) (Harik, Lobo, & Goldberg, 1998), and univariate marginal distribution algorithm (UMDA) (Mühlenbein & Paaß, 1996), are all examples of this approach.

However, assuming that each variable can be treated independently is often a strong oversimplification of the underlying problem and only a very restricted class of problems can be solved using models with no interactions. The research in PMBGAs continued with the design of methods that are capable of covering pairwise interactions by using models in the form of a chain, tree, or a forest of trees. Examples of these approaches are mutual-information-maximization input clustering (MIMIC) of De Bonet, Isbell, and Viola (1997), PBIL extension using dependency trees of Baluja and Davies (1997), and the bivariate marginal distribution algorithm (BMDA) of Pelikan and Mühlenbein (1999).

Using pairwise interactions in the used probabilistic models improved the performance of PMBGAs on difficult problems with correlated variables. However, using some pairwise interactions has still shown to be insufficient to tackle any problem decomposable into subproblems

of bounded difficulty. The algorithm should be able to discover the order of dependencies that must be covered and use an appropriate model to utilize such decomposition.

Factorized distribution algorithm (FDA) (Mühlenbein, Mahnig, & Rodriguez, 1999) provides a general way of dealing with the dependencies and independencies that specify the decomposition of the problem by factorizing the probability distribution into a product of multivariate conditional and marginal probabilities. However, FDA is not capable of learning the factorization on the fly and requires a proper factorization of the problem as input.

Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1998) and the estimation of Bayesian networks algorithm (EBNA) (Etxeberria & Larrañaga, 1999) use Bayesian networks to model and exploit the decomposition of the problem. Both BOA and EBNA use methods for learning Bayesian networks given information in the form of the promising solutions found so far in order to learn the decomposition of the problem as the optimization proceeds. The learned models are then used to guide further exploration of the search space.

Bayesian networks can encode the same class of distributions as FDA (Pelikan, Goldberg, & Cantú-Paz, 1998). However, using Bayesian networks allows the algorithms to learn the decomposition on the fly and therefore is more suitable for black-box optimization, where the structure of the problem is unknown. Recall that there are two key parts of the successful utilization of decompositional bias in search. While FDA focuses on the utilization of the decomposition and the efficiency of the search when a proper decomposition is used, BOA and EBNA include the learning component to broaden the applicability of these ideas to black-box optimization.

The next section describes BOA and the methods used in BOA to learn and sample the distributions in the form of general Bayesian networks in somewhat more detail. The section also motivates the important research issues that must be considered for a successful application of BOA to the problems decomposable into subproblems of bounded difficulty.

3 Bayesian Optimization Algorithm

The Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1998; Pelikan, Goldberg, & Cantú-Paz, 2000b) uses Bayesian networks to model promising solutions and subsequently guide the exploration of the search space. BOA generates the first population of strings at random with a uniform distribution but the initial population can also be biased to a particular region in the search space (Schwarz & Ocenasek, 2000).

The population is updated for a number of iterations (generations), each consisting of four steps. From the current population, the better strings are first selected using one of the popular selection methods, such as tournament and truncation selection. In the second step, a Bayesian network that fits the selected set of strings is constructed. In the third step, new strings are generated according to the joint distribution encoded by the constructed network. Finally, the new strings are incorporated into the original population, replacing some of the old ones or all of them.

The above four steps are repeated until some termination criteria are met. For instance, the run can be terminated when the population converges to a singleton, the population

contains a good enough solution, or a bound on the number of iterations has been reached. The pseudo-code of BOA is shown in Figure 1.

There are a number of alternative ways to perform each step. Selection can be performed by using any popular selection method. Various algorithms can be used to build the model and the method for measuring quality of each network structure can also be chosen arbitrarily. Additionally, the measure of quality of each network structure can incorporate prior information about the problem to enhance the estimation and subsequently improve the efficiency. Some of the alternatives are discussed in our previous work or studied by other authors.

The Bayesian Optimization Algorithm (BOA)

(1) set $t \leftarrow 0$
randomly generate initial population $P(0)$

(2) select a set of promising strings $S(t)$ from $P(t)$

(3) construct the network B using a chosen metric and constraints

(4) generate a set of new strings $O(t)$ according to the joint distribution encoded by B

(5) create a new population $P(t+1)$ by replacing some strings from $P(t)$ with $O(t)$
set $t \leftarrow t+1$

(6) if the termination criteria are not met, go to (2)

Figure 1: The pseudo-code of the Bayesian optimization algorithm.

The remainder of this section describes basic principles of learning and utilization of Bayesian networks. We discuss how local structures can be used to make the representation of the model more efficient and briefly describe a simple greedy algorithm for network construction. Finally, we provide a simple method for sampling the built model to generate new solutions.

3.1 Bayesian Networks

A Bayesian network (Howard & Matheson, 1981; 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) and the edges corresponding to conditional dependencies. Mathematically, a Bayesian network encodes a joint probability distribution given by

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

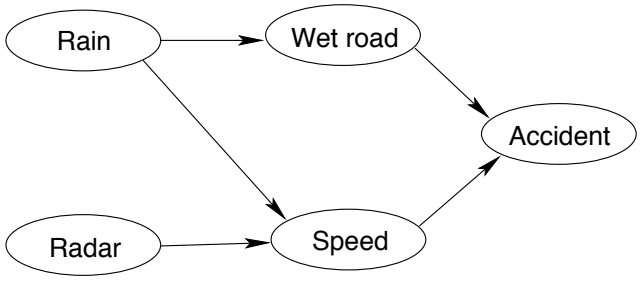


Figure 2: An example Bayesian network.

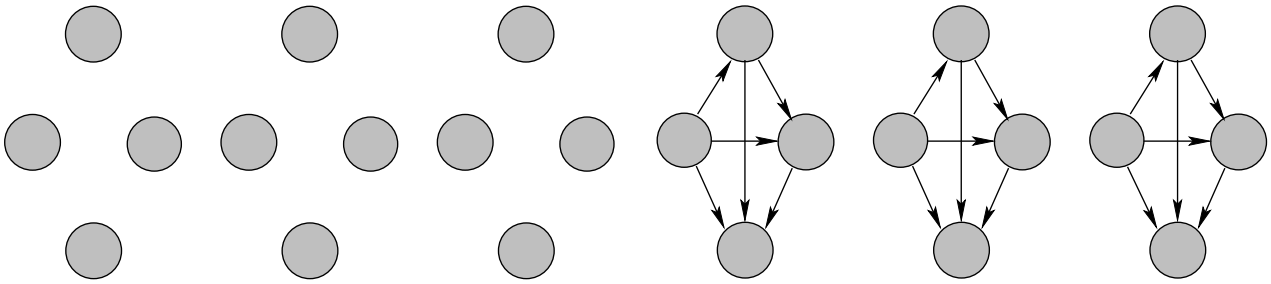
where $X = (X_1, \ldots, X_n)$ is a vector of all the variables in the problem; Π_{X_i} is the set of parents of X_i in the network (the set of nodes from which there exists an edge to X_i); and $p(X_i|\Pi_{X_i})$ is the conditional probability of X_i given its parents Π_{X_i}.

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 the parents of the variable.

A simple example Bayesian network is shown in Figure 2. The example network includes a number of conditional dependencies. For instance, the speed of the car depends on whether it is raining and/or radar is enforced. The road is most likely wet if it is raining. Additionally, the network encodes a number of simple and conditional independence assumptions. For instance, the radar enforcement is independent of whether it is raining or not. A more complex conditional independence assumption is that the probability of an accident is independent of whether the radar is enforced, given a particular speed and condition of the road.

It is important to understand the semantics of Bayesian networks in the framework of PMBGAs. The conditional dependencies represent relationships among different variables and will cause the involved variables remain in the configurations seen in the selected population of promising solutions. The conditional independence assumptions lead to mixing of the bits and pieces from the set of promising solutions. If the problem were linear, a good network would be the one without any dependencies/edges (see Figure 3(a)). If the problem were composed of separable "undecomposable" subproblems of order k, the network should be composed of fully connected sets of k nodes with no edges between the different groups (see Figure 3(b)). More complex problems lead to more complex network structures, although many problems can be solved with quite simple networks in spite of the presence of non-linear interactions of high order.

The following section discusses how the Bayesian networks can be learned given a data set (in our case, the set of selected solutions). Subsequently, we provide a simple method called forward simulation, which can be used for sampling the distribution encoded by a particular Bayesian network and its parameters.



(a) No interactions.

(b) Separable blocks of order 4.

Figure 3: A good model for the case of a problem with no interactions and a problem with separable subproblems of order 4.

3.2 Learning Bayesian Networks

The efficiency and reliability of BOA depend on how well the network reflects the dependencies and independencies in the problem that decompose the problem properly. There are two major subtasks of learning the Bayesian network.

1. Learn the structure (what edges to put in the network).
2. Learn the parameters (values of conditional probabilities).

In our case, learning the parameters of the model is not very difficult because we have complete data where the value of each variable is specified in every instance. To maximize the likelihood of the model with a fixed structure and complete data, the probabilities should be set according to the relative frequencies observed in the modeled data (in our case, the selected set of promising solutions). Thus, the parameters can be learned by iterating through all selected solutions and computing the relative frequencies of the different pieces of these solutions. Learning the structure is a much more difficult problem. The remainder of this section discusses the problems and possible solutions involved in the learning of the structure of Bayesian networks. For more information, see Heckerman, Geiger, and Chickering (1994).

Construction of the network is a very difficult combinatorial problem. In fact, it has been shown that finding the best network is NP-complete for most methods of measuring the quality of each network (Chickering, Geiger, & Heckerman, 1994) and therefore there is no known algorithm for finding the best network in a polynomial time. However, a simple greedy algorithm (Heckerman, Geiger, & Chickering, 1994) often performs very well and has been successfully used in a number of difficult machine learning tasks. The greedy algorithm performs elementary graph operations that improve the quality of the current network the most, starting from an empty network or a network from a different source. Each operation either (1) incorporates a new dependency by adding an edge into the current network or (2) adds a new (or stronger) independency by deleting an edge in the current network.

To measure quality of each network structure, various scoring metrics can be used. Recently, we have used the Bayesian-Dirichlet metric, the minimum description length (MDL) metric, and a metric that is a combination of the Bayesian-Dirichlet and MDL metric. The

use of various scoring metrics in the BOA is discussed in Pelikan et al. (2000b) and Pelikan et al. (2001).

The following two subsections discuss two classes of metrics that can be used to measure quality of competing networks. Subsequently, the use of local structures for efficient representation and learning of Bayesian networks is described. The section closes by describing a method for sampling the learned Bayesian network.

3.2.1 Bayesian metrics

Bayesian metrics (Cooper & Herskovits, 1992; Heckerman, Geiger, & Chickering, 1994) account for the uncertainty of the network structure and its parameters by using the Bayes rule and assigning prior distributions to both the network structures as well as the parameters of each structure. Quality of a particular structure is measured by computing the marginal likelihood of the structure with respect to the given data. The marginal likelihood is computed by averaging the likelihood of the models conditioned on the observed data according to a prior distribution over all possible conditional probabilities in the model:

$$p(B|D) = \frac{p(B) \int_\theta p(\theta|B) p(D|B,\theta)\, d\theta}{p(D)}, \tag{2}$$

where B is the evaluated Bayesian network structure (without particular parameters), D is the data set, and θ denotes a possible set of parameters specifying the conditional probabilities in the network B. Furthermore, $p(B)$ is the prior distribution over the network structures, $p(\theta|B)$ is the prior distribution of the parameters (conditional probabilities) of the particular network structure, and $p(D|B,\theta)$ denotes the probability of D given the network structure and its parameters. Since the probability of data denoted in the last equation by $p(D)$ is the same for all network structures, this term is usually omitted.

The computation of the marginal likelihood is intractable in the general case. The Bayesian-Dirichlet metric (BD) (Cooper & Herskovits, 1992; Heckerman, Geiger, & Chickering, 1994) assumes that the conditional probabilities follow the Dirichlet distribution and makes a number of additional assumptions, yielding the following score:

$$BD(B) = p(B) \prod_{i=1}^{n} \prod_{\pi_i} \frac{\Gamma(m'(\pi_i))}{\Gamma\left(m'(\pi_i) + m(\pi_i)\right)} \prod_{x_i} \frac{\Gamma\left(m'(x_i,\pi_i) + m(x_i,\pi_i)\right)}{\Gamma(m'(x_i,\pi_i))}, \tag{3}$$

where $p(B)$ is the prior probability of the network B; the product over x_i runs over all instances of x_i (in binary case these are 0 and 1); the product over π_i runs over all instances of the parents Π_i of X_i (all possible combinations of values of Π_i); $m(\pi_i)$ is the number of instances with the parents Π_i set to the particular values given by π_i; and $m(x_i,\pi_i)$ is the number of instances with $X_i = i$ and $\Pi_i = \pi_i$. Terms $m'(\pi_i)$ and $m'(x_i,\pi_i)$ denote prior information about the values of the corresponding parameters $m(\pi_i)$ and $m(x_i,\pi_i)$, respectively. In this paper we consider the K2 metric which uses an uninformative prior that assigns $m'(x_i,\pi_i) = 1$ and $m'(\pi_i) = \sum_{x_i} m'(x_i,\pi_i)$.

The prior distribution over the networks specified by term $p(B)$ can bias the construction toward particular structures by assigning higher prior probabilities to these networks. If

one has a good idea about the structure, one can assign higher prior probabilities to the networks similar to the structure believed to be close to the correct one (Heckerman, Geiger, & Chickering, 1994). One may also bias the search toward simpler models by assigning a higher prior probability for models with fewer parameters (Chickering, Heckerman, & Meek, 1997; Friedman & Goldszmidt, 1999; Pelikan, Goldberg, & Sastry, 2001). If there is no prior information about the network structure, the probabilities $p(B)$ are set to a constant and omitted in the construction (uniform prior).

For further details on the Bayesian metrics, please refer to Cooper and Herskovits (1992), and Heckerman, Geiger, and Chickering (1994).

3.2.2 Minimum Description Length Metrics

Minimum description length metrics (Rissanen, 1978; Rissanen, 1989; Rissanen, 1996) are based on the assumption that the number of regularities in the data encoded by the model is somehow proportional to the amount of compression of the data allowed by the model. The model that results in the highest compression should therefore encode the most regularities. There are two major approaches to the design of MDL metrics. The first approach is based on a two-part coding where the score is negatively proportional to the sum of the number of bits required to store the (1) model, and (2) data compressed according to the model. The universal code normalizes the probability of the data given the particular model by the sum of the probabilities of all data sequences given the particular model. The normalized probability of the data is used as the basis for computing the number of bits required to compress the data.

In this paper, we analyze one of the MDL-based metrics, called the Bayesian Information Criterion (BIC) (Schwarz, 1978) used previously in the estimation of Bayesian networks algorithm (EBNA) (Etxeberria & Larrañaga, 1999). The reason for choosing BIC metric for our analysis is that it is much easier to analyze than Bayesian-Dirichlet and other Bayesian metrics are. In a binary case, BIC assigns the network structure a score according to

$$BIC(B) = \sum_{i=1}^{n} \left(-H(X_i|\Pi_i)N - 2^{|\Pi_i|}\frac{\log_2(N)}{2} \right), \tag{4}$$

where $H(X_i|\Pi_i)$ is the conditional entropy of X_i given its parents Π_i, n is the number of variables in the problem, and N is the population size (the size of the training data set). The conditional entropy $H(X_i|\Pi_i)$ is given by

$$H(X_i|\Pi_i) = -\sum_{x_i,\pi_i} p(x_i,\pi_i)\log_2 p(x_i|\pi_i). \tag{5}$$

$H(X_i|\Pi_i)$ denotes the average number of bits required to store the value of X_i given information about the value of Π_i. The entropy is multiplied by the population size in order to reflect the number of bits required to store the entire population. The term $\log_2(N)$ denotes the number of bits required to store one parameter of the model (one probability or frequency). The number of bits required to store each parameter is divided by two because only half of the bits really matter in practice. The term with the conditional entropy ensures that the more

the information about the parents of a variable enables to compress the values of the variable, the higher the value of the BIC metric. The term with the $\log_2(N)$ introduces a pressure toward simpler models by decreasing the metric proportionally to the number of parameters required to fully specify the network.

For further details on the minimum description length metrics, please see Rissanen (1996) and Grünwald (1998).

3.2.3 Using Local Structures

To encode the conditional probabilities corresponding to the nodes of the network, one can use a simple probability table listing the probabilities of all possible instances of a variable and its parents. The probabilities of one particular value of each variable can be eliminated and computed using the remaining ones because the probabilities sum to one. More sophisticated structures such as default tables (Friedman & Goldszmidt, 1999), decision graphs (Chickering, Heckerman, & Meek, 1997), or decision trees (Friedman & Goldszmidt, 1999; Chickering, Heckerman, & Meek, 1997) can also be used.

Using local structures reduces the space required to store the network and is therefore important for an efficient implementation of complex models as well as for increasing the accuracy of MDL metrics. Additionally, using local structures improves the quality of the learning of the network structure (Chickering, Heckerman, & Meek, 1997) by allowing the model to encode and learn more specialized dependencies and independencies. Finally, the local structures allow increasing the expressiveness of the model without compromising its complexity.

3.2.4 Simulation of Bayesian Networks

Given the network structure and the conditional probabilities in the structure, the new solutions are generated according to the distribution encoded by the model. The sampling proceeds in two steps. The first step orders the nodes so that each node is preceded by its parents. The basic idea is to generate the variables in a certain sequence so that the values of the parents of each variable are generated prior to the generation of the variable itself.

In the second step, the values of all variables of a new individual are generated according to the computed ordering. Since the algorithm generates the variables according to the ordering determined in the first phase, when the algorithm attempts to generate a value of each new variable, the parents of the variable are already generated and the probabilities of different values for the variable are thus given by the corresponding elements in the probability tables or the local structures encoding these probability tables. The second step is repeated for each new individual.

The next section focuses on the correspondence between the problem at hand and the learned model. The important issue is whether the commonly used selection methods allow us to learn a correct model that represents the interactions among the variables that are nonlinearly correlated and assumes independence of the variables whose contributions are indeed independent.

4 Dependencies: The Good, The Bad, and The Ugly

There are two important tasks required for a successful application of the methods for learning and utilizing Bayesian networks in black-box optimization. The methods for learning the structure of the network must be capable of identifying nonlinear interactions among the different variables in the problem. Additionally, the learned network (assuming it encodes correct relationships) must lead to efficient optimization. There has been work done in justifying the use of a correct model. Some of the results focus on the theoretical analysis of GAs and their working (Harik, Cantú-Paz, Goldberg, & Miller, 1997; Mühlenbein & Paaß, 1996), while others consider the use of graphical models in particular (Bosman & Thierens, 1999; Mühlenbein, Mahnig, & Rodriguez, 1999; Pelikan, Goldberg, & Cantú-Paz, 1998). However, only little work has been done to justify the success of PMBGAs in learning a good model (Pelikan, Goldberg, & Cantú-Paz, 2000a). Moreover, most of this work has been empirical in its nature.

There are two basic pressures toward models with interactions. The first pressure originates in the fitness function and is brought about by the selection operator, where we can expect certain configurations of correlated variables be preferred at the expense of others. This leads to the discovery of the good, non-linear interactions in the problem that, in turn, lead to efficient optimization. However, there is also a pressure toward models with interactions coming from selection only, even if the fitness contributions of the considered variables are independent (Thierens, 1995). These interactions negatively affect the efficiency of the search by increasing the requirements on the population sizing and time to convergence. For a successful application of PMBGAs to difficult decomposable problems, we must ensure that the nonlinear interactions in the problem are discovered while the linear interactions brought about by the selection operator only are ignored.

Note that while learning graphical models attempts to learn a model assuming the data represents the actual dependencies in the target distribution, in our case the selection operator introduces two kinds of dependencies, one of which we want to discover while the other one we want to ignore. Although the scoring metrics are quite robust and capable of finding a reasonable compromise between the models with and without interactions, the analysis of the two pressures is very important for the justification of the use of graphical models in BOA and other PMBGAs.

This section focuses on the issue of learning the good model and shows that the BIC metric is capable of discovering correct interactions without being misled by the nonlinearities introduced by the selection operator. The interactions between two variables are only considered, but the results of the simulation indicate that we can expect similar behavior in a more general setting.

The section starts by listing the assumptions used in the remainder of the section. Most assumptions follow directly from the decomposability and bounded difficulty of the problem at hand. Section 4.3 quantifies the effect of the binary tournament selection on the probabilities and relates the resulting probabilities to the fitness. We show that the probabilities for any two variables in the considered block of variables can be expressed in terms of a simpler two-bit block and a particular two-bit fitness function. Section 4.4 analyzes the two-bit case in two separate cases: (1) nonlinearly correlated variables and (2) independent variables. We

show that to discover non-linear interaction the population size should grow linearly with the problem size. On the other hand, to be misled by interactions among non-correlated (independent) variables, the population size grows quadratically with the problem size and is orders of magnitude larger than in the non-linear case. This leads to a conclusion that there is a large enough "sweet spot" for the correct setting of the population size ensuring that we do discover nonlinear interactions but still assume independency of uncorrelated variables. Along the way, theory is compared to the empirical results of our simulation of model learning in the asymptotical and actual cases.

4.1 Assumptions

To simplify the analysis, we make a couple of assumptions about the problem. First, we assume that the fitness function is defined as the sum of subfunctions applied to disjoint subsets of variables of order k and that all the subfunctions are the same. However, the conclusions of the analysis also hold in the case of non-separable functions where the contribution of the considered block of k bits is independent of the remaining solution and where the fitness contributions from the remaining solution can be approximated by the normal distribution with a variance growing linearly with the problem size.

Without loss of generality, we denote the considered block of variables (bits) by $X = (X_1, \ldots, X_k)$ or $Y = (Y_1, \ldots, Y_k)$ and the fitness contribution of this block by $f_{BB}(X_1, \ldots, X_k) = f_{BB}(X)$ or $f_{BB}(Y_1, \ldots, Y_k) = f_{BB}(Y)$. We denote the fitness of the solutions with a block $X = (X_1, \ldots, X_k)$ by $f(X)$ and we assume that the contributions of the remaining solutions (variables $X_{k+1}, X_{k+2}, \ldots, X_n$) can be modeled by a normal distribution with the variance proportional to the size of the problem:

$$f(X) \sim N(\bar{f}(X), \sigma_N^2), \tag{6}$$

where $\bar{f}(X) = \bar{f}(X_1, \ldots, X_k)$ is the average fitness of the solutions with fixed variables $(X_1, \ldots, X_k)$, and σ_N^2 is the variance of the noise coming from the remaining variables in the solutions. The above assumption can be justified by the central limit theorem for all decomposable problems of bounded difficulty where the fitness contribution of each subproblem is of the same magnitude and the order of the subproblems is bounded by a constant.

By $p(X) = p(X_1, \ldots, X_k)$, we denote the probability of the particular block of variables. The probability is computed as the relative frequency of this solution given a population of solutions. The probability of other subsets of variables is denoted in a similar fashion. For example, we use $p(X_1, X_2)$ to denote the probability of the first two variables of the block X, and $p(X_2)$ to denote the probability of the second variable in the block X.

Additionally, we assume that the probabilities follow their expected behavior in the case of an infinite population, although in practice we can expect additional noise due to the finite size of the population. In this fashion, we can eliminate the effects of finite sampling and focus on the asymptotic behavior of the BIC metric itself. The section compares the theoretical results to the practical case of finite populations.

We only consider the decision making between adding or not adding the *first* parent of the particular node. That means that we assume that the node has no parents so far. Finally, all

variables are assumed to be binary.

To provide some empirical results, we tested two fitness functions. The first function is onemax, defined as

$$f_{onemax}(X) = \sum_{i=1}^{n} X_i, \tag{7}$$

where $X = (X_1, \ldots, X_n)$ is the input string. In onemax, all variables are independent. The size of the block of variables in onemax does not affect the correctness of our assumptions and therefore any block size should lead to the same result.

The second test function is the trap function of order 5, defined as

$$f_{trap5}(X) = \sum_{i=0}^{n/5-1} trap_5(X_{5i+1} + X_{5i+2} + X_{5i+3} + X_{5i+4} + X_{5i+5}), \tag{8}$$

where $trap_5(u)$ is the trap function of order 5 defined as a function of unitation (it depends only on the number of ones in the considered block of order k). The function $trap_5(u)$ is defined as

$$trap_5(u) = \begin{cases} 4-u & \text{if } u < 5 \\ 5 & \text{otherwise} \end{cases}, \tag{9}$$

where u is the sum of the variables in the block of 5 variables. In the trap function, a block of at least 5 variables must be considered for our assumptions to be correct and the fitness of this block is given by Equation 9. The important feature of the trap function is that the blocks of 5 variables cannot be further decomposed without increasing the population-size requirements of the algorithm exponentially (Thierens & Goldberg, 1993).

4.2 Critical Population Size

To decide whether the model-building procedure should put an edge from X_2 to X_1, we must compare the values of the used scoring metric for the current network with and without the edge and pick the better alternative. Since both MDL and Bayesian metrics consider the contribution of each variable separately from the contributions of other variables, it is sufficient to look at the term corresponding to the node X_1.

The score assigned by BIC to X_1 without the edge from X_2 to X_1 is given by

$$BIC(X_1) = -H(X_1)N - \frac{\log_2 N}{2}, \tag{10}$$

where $H(X_1)$ is the entropy of the variable X_1, and N is the number of selected solutions (selected population size). After adding an edge from X_2 to X_1, the new score for X_1 is given by

$$BIC(X_1 \leftarrow X_2) = -H(X_1|X_2)N - \log_2 N, \tag{11}$$

where $H(X_1|X_2)$ is the conditional entropy of the variable X_1 given the information about the value of the variable X_2. For an addition of the edge $X_2 \rightarrow X_1$, the following inequality

must be satisfied:

$$BIC(X_1 \leftarrow X_2) > BIC(X_1). \tag{12}$$

By substituting the equations 10 and 11 into the last equation, we get

$$(H(X_1) - H(X_1|X_2))\, N - \frac{\log_2 N}{2} > 0. \tag{13}$$

Let us denote the difference between the marginal and conditional entropy of X_1 by D:

$$\begin{aligned} D &= H(X_1) - H(X_1|X_2) \\ &= -\sum_{X_1} p(X_1) \log_2 p(X_1) + \sum_{X_1,X_2} p(X_1, X_2) \log_2 p(X_1|X_2) \\ &= \sum_{X_1,X_2} p(X_1, X_2) \log_2 p(X_1, X_2) - \sum_{X_1} p(X_1) \log_2 p(X_1) - \sum_{X_2} p(X_2) \log_2 p(X_2) \end{aligned} \tag{14}$$

If the variables X_1 and X_2 are not independent, D is strictly positive. Since this is the case in the remainder of this section (even in the case of onemax) and the linear term grows faster than the logarithmical one, Equation 13 will be satisfied for a large enough N. Intuitively, when the two variables are not independent, for a big enough population size, the dependency should be discovered. We call the sufficient population size for the discovery of the dependency $X_2 \rightarrow X_1$ the *critical population size* and denote it by N_{crit}. To determine N_{crit}, the following equation must be solved for N:

$$N - \frac{\log_2 N}{2D} = 0. \tag{15}$$

The above equation has two solutions but there is no closed form for either of these solution. The dependency is discovered for the population sizes lower than the first (lower) solution or greater than the second (greater) one. The first solution is approximately equal to $1 + 2D \ln 2$. However, since even for small problems the value of D is very small, this solution is of no interest in our case. N_{crit} is therefore defined as the larger of the two solutions of the last equation.

If the ratio $\frac{1}{2D}$ is large enough, the larger of the two solutions of the above equation follows a power law is thus of the form $\alpha \left(\frac{1}{2D}\right)^{\beta}$, where $\beta \sim 1.05$. Therefore,

$$\log N_{crit} \approx 1.05 \log \left(\frac{1}{2D}\right) + 2.12. \tag{16}$$

Since for an increasing problem size the magnitude of D decreases inversely proportional to the number of decision variables in the problem, or even faster, the above approximation can be directly used to determine N_{crit}. Figure 4 shows the numerical solution and its approximation using Equation 16.

In order to apply our results to the scale-up behavior of BOA with BIC metric, we are interested in the *growth* of N_{crit} with respect to the size of the problem (i.e., the number of decision variables). Using the approximation given by Equation 16, it can be shown that the growth of N_{crit} is proportional to the growth of $\frac{1}{2D}$ and therefore to determine the growth of

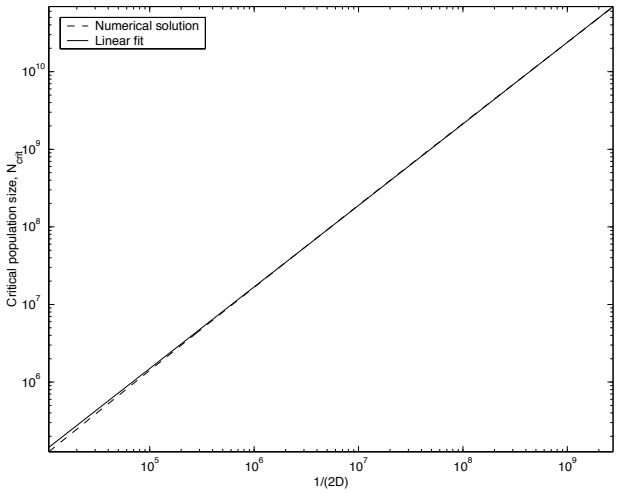


Figure 4: Critical population size with respect to the ratio $\frac{1}{2D}$ for BIC metric.

N_{crit} with respect to the size of the problem, it is sufficient to compute the growth of $\frac{1}{2D}$ with respect to the same parameter.

The following section analyzes the dynamics of frequencies $p(X_1, X_2)$ after applying binary tournament selection to a uniformly distributed population of random solutions. The section implies that the dynamics of these frequencies does not depend on the order of the subfunction involving both X_1 and X_2 denoted by k and it is therefore sufficient to focus on the subfunctions of order 2. Subsequently, we present a detailed analysis of the general two-bit case and its implications on the scale-up behavior and correctness of BOA with BIC metric.

4.3 Block Probabilities After Binary Tournament

The initial population is generated at random with a uniform distribution and therefore the probability of any instantiation of the variables in the considered block of k binary variables is given by

$$p_{init}(X) = \frac{1}{2^k}\cdot, \tag{17}$$

where $X = (X_1, \ldots, X_k)$ denotes the set of variables in the considered block.

Binary tournament selects two parents at random and chooses the one with the better fitness. Let us denote the probability of a tournament between the competing blocks X and Y (both denoting the same block of variables of order k). The order of the selection of the two competing blocks does not affect the result of the selection and therefore the probability of X after applying binary tournament selection is given by

$$p(X) = \sum_{Y_1,\ldots,Y_k} 2p_{tourn}p(f(X) > f(Y)), \tag{18}$$

where $Y = (Y_1, \ldots, Y_k)$ denotes the block competing with X in the tournament, p_{tourn} denotes the probability of choosing X and Y for the tournament, and $f(X)$ and $f(Y)$ denote the distribution of fitness values for the blocks X and Y, respectively (see Equation 6). Therefore,

$p(f(X) > f(Y))$ denotes the probability that the solution with the block X wins the tournament over the solution with the block Y. Using Equation 17, the probability of a tournament of X and Y is given by

$$p_{tourn} = p_{init}(X)p_{init}(Y) = \frac{1}{2^{2k}}. \tag{19}$$

The probability of X winning the tournament over Y can be rewritten as

$$p(f(X) > f(Y)) = p(f(X) - f(Y) > 0). \tag{20}$$

Since $f(X)$ and $f(Y)$ follow a normal distribution (see Equation 6), their difference also follows a normal distribution with the mean equal to the difference of the individual means of $f(X)$ and $f(Y)$, and the variance equal to the sum of the variances of the two distributions. The difference of the mean fitness of $f(X)$ and $f(Y)$ is equal to the difference of their contributions to the overall fitness denoted by $f_{BB}(X)$ and $f_{BB}(Y)$, because the contributions of the remaining solutions cancel out. Thus,

$$f(X) - f(Y) \sim N(f_{BB}(X) - f_{BB}(Y), 2\sigma_N^2). \tag{21}$$

This yields

$$p(f(X) > f(Y)) = F\left(\frac{f_{BB}(X) - f_{BB}(Y)}{\sqrt{2}\sigma_N}\right), \tag{22}$$

where $F(x)$ denotes a cumulative probability density function for a zero-mean normal distribution with the standard deviation of 1. The resulting probability of X after the binary tournament selection is thus given by

$$p(X) = \sum_{Y} \frac{1}{2^{2k-1}} F\left(\frac{f_{BB}(X) - f_{BB}(Y)}{\sqrt{2}\sigma_N}\right). \tag{23}$$

Since the ratio of the fitness differences to noise decreases with the problem size and is usually a very small number for moderate–to–large sized problems, we can use a linear approximation of the cumulative density function where $F(x) = \frac{1}{2} + \frac{x}{\sqrt{2\pi}}$, yielding

$$p(X_1, \ldots, X_k) = \sum_{Y_1,\ldots,Y_k} \frac{1}{2^{2k-1}} \left(\frac{1}{2} + \frac{f_{BB}(X) - f_{BB}(Y)}{2\sqrt{\pi}\sigma_N}\right). \tag{24}$$

To compute D, we must now determine the marginal probabilities of the first two variables

X_1 and X_2. These can be computed by summing $p(X_1, \ldots, X_k)$ over the variables X_3 to X_k:

$$\begin{aligned} p(X_1, X_2) &= \sum_{X_3,\ldots,X_k} p(X) \\ &= \sum_{X_3,\ldots,X_k} \sum_{Y_1,\ldots,Y_k} \frac{1}{2^{2k-1}} \left(\frac{1}{2} + \frac{f_{BB}(X) - f_{BB}(Y)}{2\sqrt{\pi}\sigma_N} \right) \\ &= \frac{1}{2^{2k-1}} \left(2^k \sum_{X_3,\ldots,X_k} \frac{f_{BB}(X)}{2\sqrt{\pi}\sigma_N} + 2^{k-2} \sum_{Y_1,\ldots,Y_k} \left(\frac{1}{2} - \frac{f(Y)}{2\sqrt{(\pi)}\sigma_N} \right) \right) \\ &= \frac{1}{4} \left(1 + \frac{\bar{f}_{BB}(X_1, X_2) - \bar{f}_{BB}}{\sqrt{\pi}\sigma_N} \right), \end{aligned} \tag{25}$$

where $\bar{f}_{BB}(X_1, X_2)$ denotes the average fitness contribution of X with fixed values for variables X_1 and X_2, and $\bar{f}_{BB}$ denotes the average block fitness f_{BB}, i.e.

$$\bar{f}_{BB}(X_1, X_2) = \frac{1}{2^{k-2}} \sum_{X_3,\ldots,X_k} f_{BB}(X), \tag{26}$$

and

$$\bar{f}_{BB} = \frac{1}{2^k} \sum_{X_1,\ldots,X_k} f_{BB}(X). \tag{27}$$

The pairwise frequencies therefore depend only on the average block fitnesses of the corresponding two bits. The behavior for any block of order k can be accurately approximated by a special case of a two-bit building block with the fitness defined according to the average building-block fitnesses $\bar{f}_{BB}(X_1, X_2)$. The next section computes D for the general function defined over two binary variables. Additionally, the section provides a couple of examples confirming that the approximations made in our analysis are reasonable and that the relationship between the two-bit and k-bit is correct indeed.

4.4 General Two-Bit Case

In the general two-bit case, the fitness of the block $X = (X_1, X_2)$ can be written in the following form:

$$f_{BB}(X_1, X_2) = a_0 + a_1 X_1 + a_2 X_2 + a_{12} X_1 X_2, \tag{28}$$

where a_0, a_1, a_2, and a_{12} are constants. In the above equation, if the contribution of X_1 is independent of X_2, the constant $a_{12} = 0$. On the other hand, if $a_{12} \neq 0$, the contributions of the variables X_1 and X_2 are correlated.

The frequencies of two bits X_1 and X_2 after applying the tournament selection can be

computed using Equation 25, yielding

$$\begin{aligned} p(X_1, X_2) &= \frac{1}{4}\left(1 + 2\frac{f_{BB}(X_1, X_2) - \bar{f}_{BB}}{2\sqrt{\pi}\sigma_N}\right) \\ &= \frac{1}{4}\left(1 + 2\frac{a_0 + a_1 X_1 + a_2 X_2 + a_{12} X_1 X_2 - \frac{2a_1+2a_2+a_{12}}{4}}{2\sqrt{\pi}\sigma_N}\right) \\ &= \frac{1}{4}\left(1 + \frac{a_1(4X_1 - 2) + a_2(4X_2 - 2) + a_{12}(4X_1X_2 - 1)}{4\sqrt{\pi}\sigma_N}\right) \end{aligned} \tag{29}$$

By summing the above equations over X_1 and X_2, respectively, we get

$$\begin{aligned} p(X_1) &= \frac{1}{2}\left(1 + \frac{a_1(4X_1 - 2) + a_{12}(2X_1 - 1)}{4\sqrt{\pi}\sigma_N^2}\right) \\ p(X_2) &= \frac{1}{2}\left(1 + \frac{a_2(4X_2 - 2) + a_{12}(2X_2 - 1)}{4\sqrt{\pi}\sigma_N^2}\right) \end{aligned} \tag{30}$$

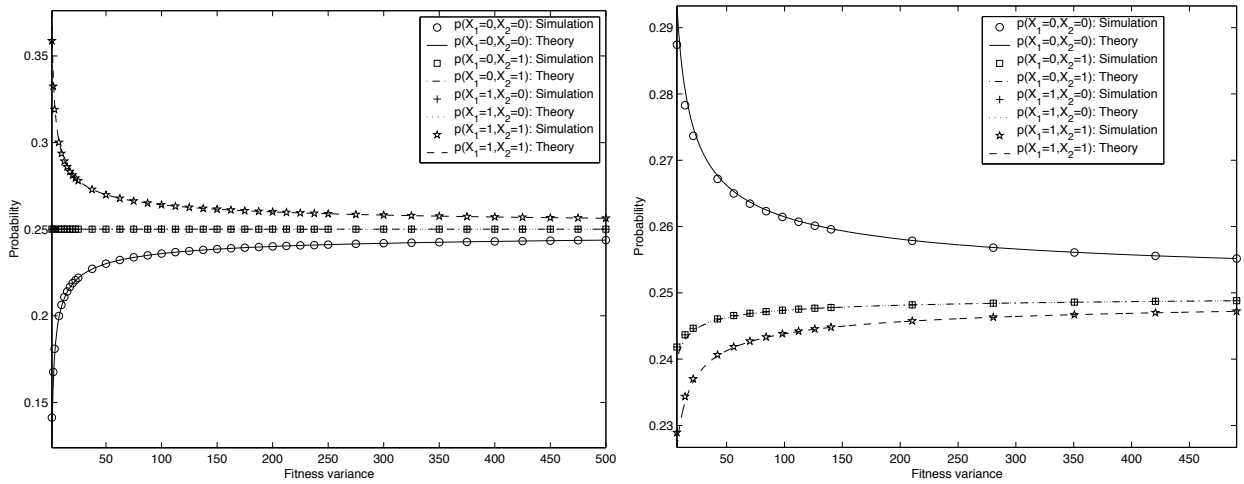
The above equations can be used to compute the frequencies of any instantiation of X_1 and X_2, yielding the following set of equations:

$$\begin{aligned} p(X_1 = 0, X_2 = 0) &= \frac{1}{4}\left(1 + \frac{-2a_1 - 2a_2 - a_{12}}{4\sqrt{\pi}\sigma_N}\right) = \frac{1}{4}(1 + \chi_{00}) \\ p(X_1 = 0, X_2 = 1) &= \frac{1}{4}\left(1 + \frac{-2a_1 + 2a_2 - a_{12}}{4\sqrt{\pi}\sigma_N}\right) = \frac{1}{4}(1 + \chi_{01}) \\ p(X_1 = 1, X_2 = 0) &= \frac{1}{4}\left(1 + \frac{2a_1 - 2a_2 - a_{12}}{4\sqrt{\pi}\sigma_N}\right) = \frac{1}{4}(1 + \chi_{10}) \\ p(X_1 = 1, X_2 = 1) &= \frac{1}{4}\left(1 + \frac{2a_1 + 2a_2 + 3a_{12}}{4\sqrt{\pi}\sigma_N}\right) = \frac{1}{4}(1 + \chi_{11}) \end{aligned} \tag{31}$$

Additionally, the probabilities of single variables X_1 and X_2, can be computed as follows:

$$\begin{aligned} p(X_1 = 0) &= \frac{1}{2}\left(1 + \frac{-2a_1 - a_{12}}{4\sqrt{\pi}\sigma_N^2}\right) = \frac{1}{2}(1 - \chi_1) \\ p(X_1 = 1) &= \frac{1}{2}\left(1 + \frac{2a_1 + a_{12}}{4\sqrt{\pi}\sigma_N^2}\right) = \frac{1}{2}(1 + \chi_1) \\ p(X_2 = 0) &= \frac{1}{2}\left(1 + \frac{-2a_2 - a_{12}}{4\sqrt{\pi}\sigma_N^2}\right) = \frac{1}{2}(1 - \chi_2) \\ p(X_2 = 1) &= \frac{1}{2}\left(1 + \frac{2a_2 + a_{12}}{4\sqrt{\pi}\sigma_N^2}\right) = \frac{1}{2}(1 + \chi_2) \end{aligned} \tag{32}$$

Before substituting the above frequencies into the equation for D, we compare the approximation with an accurate simulation for two example fitness functions. Figure 5(a) shows the dynamics of the pairwise frequencies for the onemax case (see Equation 7) where $a_1 = a_2 = 1$ and $a_0 = a_{12} = 0$. Figure 5(b) shows the dynamics of the pairwise frequencies for the first two bits of the trap function of order 5 (see Equation 8) where the two-bit approximation



(a) Onemax.

(b) Trap of order 5.

Figure 5: Pairwise frequencies versus their approximation with respect to noise.

yields $a_0 = 2.5$, $a_1 = a_2 = -1$, and $a_{12} = 0.75$. In both examples, the approximations as well as the empirical results for $p(X_1 = 0, X_2 = 1)$ and $p(X_1 = 1, X_2 = 0)$ are the same and, therefore, they overlap. The true expected values of $p(X_1, X_2)$ were determined by a simulation performed according to the accurate model of the frequency dynamics for both functions (without any approximations).

To determine the final form of D for the general two-bit case, the two sets of equations (equations 31 and 32) can be substituted into Equation 14 for D. As discussed above, the general two-bit case generalizes to a k-bit block in a straightforward manner. The order of terms required for an accurate approximation of D depends on whether $a_{12} = 0$ or $a_{12} \neq 0$. Therefore, we split the computation of D into two cases. The first case considers $a_{12} \neq 0$ where the fitness contribution of X_1 and X_2 are not independent (good dependency). In the second case, $a_{12} = 0$ and therefore the fitness contributions of X_1 and X_2 are independent (bad dependency). The following sections discuss each of the two cases. Subsequently, the overall result is discussed.

4.4.1 Dependent Case: $a_{12} \neq 0$

First, let us compute an approximation of the entropy of X_1 defined as

$$H(X_1) = -\sum_{X_1} p(X_1) \log_2 p(X_1). \tag{33}$$

Using the set of equations 32, the entropy of X_1 can then be computed as

$$\begin{aligned} H(X_1) &= -\frac{1}{2}(1-\chi_1)\log_2\frac{1}{2}(1-\chi_1) - \frac{1}{2}(1+\chi_1)\log_2\frac{1}{2}(1+\chi_1) \\ &= -\frac{1}{2}\left(\log_2(1-\chi_1^2) + \chi_1(\log_2(1+\chi_1) - \log_2(1-\chi_1))\right) + 1 \end{aligned} \tag{34}$$

Since χ_1 is very small for moderate–to–large sized problems (it approaches zero as σ_N approaches infinity), we can use a linear approximation of the logarithm near 1. In particular,

$$\begin{aligned} \log_2(1-\chi_1^2) &\approx \frac{-\chi_1^2}{\ln 2}, \\ \chi_1\log_2(1+\chi_1) &\approx \frac{\chi_1^2}{\ln 2}, \\ \chi_1\log_2(1-\chi_1) &\approx -\frac{\chi_1^2}{\ln 2}. \end{aligned} \tag{35}$$

Thus,

$$\begin{aligned} H(X_1) &= -\frac{\chi_1^2}{2\ln 2} + 1 \\ &= -\frac{4a_1^2 + 4a_1a_{12} + a_{12}^2}{32\pi\sigma_N^2\ln 2} + 1. \end{aligned} \tag{36}$$

The conditional entropy $H(X_1|X_2)$ is given by

$$H(X_1|X_2) = H(X_1, X_2) - H(X_2),$$

where $H(X_1, X_2)$ is the joint entropy of X_1 and X_2, and $H(X_2)$ is the entropy of X_2. $H(X_2)$ can be computed analogously to $H(X_1)$, yielding

$$H(X_2) = -\frac{4a_2^2 + 4a_2a_{12} + a_{12}^2}{32\pi\sigma_N^2\ln 2} + 1.$$

The joint entropy $H(X_1, X_2)$ is given by

$$H(X_1, X_2) = -\sum_{X_1, X_2} p(X_1, X_2)\log_2 p(X_1, X_2) = -(A_{00} + A_{01} + A_{10} + A_{11}),$$

where

$$A_{ij} = p(X_1 = i, X_2 = j)\log_2 p(X_1 = i, X_2 = j). \tag{37}$$

The terms A_{ij} can be approximated as follows:

$$\begin{aligned} A_{ij} &= \frac{1}{4}\left((1+\chi_{ij})\log_2\frac{1}{4}(1+\chi_{ij})\right) \\ &= \frac{1}{4}\left(\log_2(1+\chi_{ij}) + x_{ij}\log_2(1+\chi_{ij}) - 2(1+\chi_{ij})\right) \end{aligned} \tag{38}$$

Since χ_{ij} is very small, we can use the following approximations to simplify the above equation:

$$\begin{aligned} \log_2(1+\chi_{ij}) &\approx \frac{2\chi_{ij}-\chi_{ij}^2}{2\ln 2}, \\ \chi_{ij}\log_2(1+\chi_{ij}) &\approx \frac{\chi_{ij}^2}{\ln 2}. \end{aligned} \tag{39}$$

Thus,

$$A_{ij} = \frac{1}{4}\left(\frac{2\chi_{ij}+\chi_{ij}^2}{2\ln 2} - 2(1+\chi_{ij})\right) \tag{40}$$

By substituting the approximations of A_{ij} and the equations for χ_{ij}, we get

$$H(X_1,X_2) = -\frac{1}{8}\left(\frac{16a_1^2+16a_2^2+12a_{12}^2+16a_1a_{12}+16a_2a_{12}}{16\pi\sigma_N^2\ln 2}\right)+2 \tag{41}$$

Thus, the difference D between the marginal and conditional entropies can be approximated by

$$\begin{aligned} D &= H(X_1) - H(X_1|X_2) \\ &= \frac{a_{12}^2}{32\pi\sigma_N^2\ln 2}. \end{aligned} \tag{42}$$

The above equation can be rewritten as

$$\log D = -2\log\sigma_N + \log\frac{a_{12}^2}{32\pi\ln 2}. \tag{43}$$

For the trap function of order 5 defined in Equation 8, $a_{12}=0.75$, and therefore D is given by

$$\log D = -2\log\sigma_N + \log\frac{a_{12}^2}{32\pi\ln 2}.$$

The final approximation of D for the case of trap function of order 5 given by the last equation is verified with the actual results in Figure 6.

By substituting Equation 42 into Equation 16, we can infer that for the fitness variance large enough,

$$N_{crit} = O(\sigma_N^{2.1}). \tag{44}$$

Using the assumption that $\sigma_N^2 \propto n$ where n is the number of variables in the problem, we can imply that

$$N_{crit} = O(n^{1.05}). \tag{45}$$

In other words, the critical population size for discovering a dependency between the two variables that are non-linearly correlated grows approximately linearly with the size of the problem.

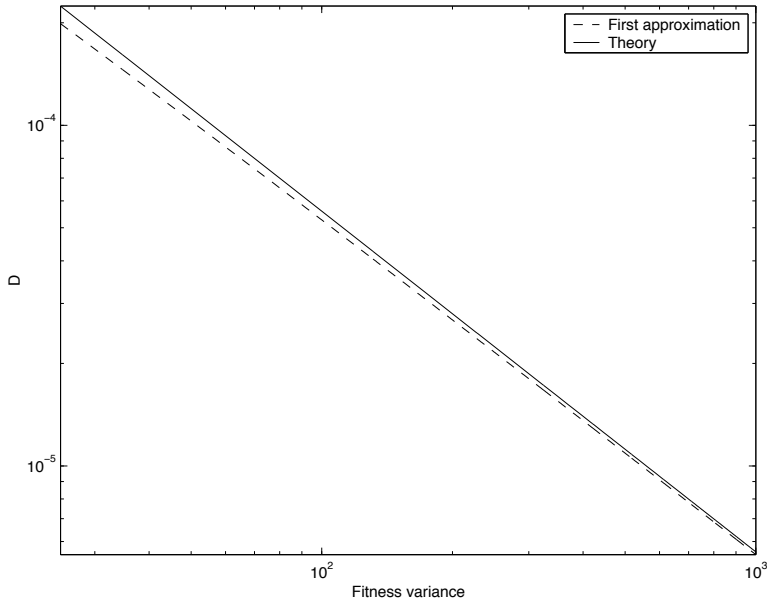


Figure 6: D versus σ_N^2 for trap function of order 5.

4.4.2 Independent Case: $a_{12} = 0$

Note that for this case $a_{12} = 0$. Therefore,

$$\chi_1 = \frac{a_1}{2\sqrt{\pi}\sigma_N}, \quad \chi_2 = \frac{a_2}{2\sqrt{\pi}\sigma_N}. \tag{46}$$

We can now write x_2 in terms of x_1 as

$$\chi_2 = \frac{a_2}{a_1}\chi_1 = b\chi_1, \tag{47}$$

where $b = a_2/a_1$. We know that,

$$D = H(X_1) + H(X_2) - H(X_1, X_2). \tag{48}$$

Furthermore, we know that

$$\begin{aligned} H(X_i) &= -\frac{1}{2}\left[(1+\chi_i)\log_2\left(\frac{1+\chi_i}{2}\right) + (1-\chi_i)\log_2\left(\frac{1-\chi_i}{2}\right)\right] \\ &= -\frac{1}{2}\left[(1+\chi_i)\log_2(1+\chi_i) + (1-\chi_i)\log_2(1-\chi_i) - 2\right] \\ &= -\frac{1}{2}\left[\log_2((1+\chi_i)(1-\chi_i)) + \chi_i\log_2\left(\frac{1+\chi_i}{1-\chi_i}\right) - 2\right]. \end{aligned} \tag{49}$$

Since χ_i is very small, the logarithms in Equation 49 can be approximated by the following series representations:

$$\log_2\left((1+\chi_i)(1-\chi_i)\right) \approx -\frac{2\chi_i^2+\chi_i^4}{2\ln 2}, \tag{50}$$

$$\log_2\left(\frac{1+\chi_i}{1-\chi_i}\right) \approx \frac{6\chi_i+2\chi_i^3}{3\ln 2}. \tag{51}$$

Using these series approximations, the entropy $H(X_i)$ is given by

$$H(X_i) = -\frac{1}{2}\left[\frac{6\chi_i^2+\chi_i^4}{6\ln 2} - 2\right]. \tag{52}$$

We can now compute the relations for the entropies $H(X_1)$ and $H(X_2)$:

$$H(X_1) = -\frac{6\chi_1^2+\chi_1^4}{12\ln 2} + 1. \tag{53}$$

$$\begin{aligned} H(X_2) &= -\frac{6\chi_2^2+\chi_2^4}{12\ln 2} + 1, \\ &= -\frac{6b^2\chi_1^2+b^4\chi_1^4}{12\ln 2} + 1. \end{aligned} \tag{54}$$

The only other term remaining to compute the entropy difference D is the joint entropy, $H(X_1, X_2)$ given by

$$H(X_1, X_2) = -\left[A_{00} + A_{01} + A_{10} + A_{11}\right],$$

where

$$\begin{aligned} \chi_{00} &= -(1+b)\chi_1, \\ \chi_{01} &= -(1-b)\chi_1, \\ \chi_{10} &= (1-b)\chi_1, \\ \chi_{11} &= (1+b)\chi_1, \end{aligned} \tag{56}$$

and

$$\begin{aligned} A_{ij} &= \frac{1}{4}\left[(1+\chi_{ij})\log_2\left(\frac{1+\chi_{ij}}{4}\right)\right] \\ &= \frac{1}{4}\left[\log_2(1+\chi_{ij}) + \chi_{ij}\log_2(1+\chi_{ij}) - 2(1+\chi_{ij})\right]. \end{aligned} \tag{57}$$

From the above equation, we can write

$$\begin{aligned}
h\left(\chi_{00}\right) &= \frac{1}{4}\left[\log_2\left(1-(1+b)\chi_1\right)-(1+b)\chi_1\log_2\left(1-(1+b)\chi_1\right)-2\left(1-(1+b)\chi_1\right)\right],\\
h\left(\chi_{01}\right) &= \frac{1}{4}\left[\log_2\left(1-(1-b)\chi_1\right)-(1-b)\chi_1\log_2\left(1-(1-b)\chi_1\right)-2\left(1-(1-b)\chi_1\right)\right],\\
h\left(\chi_{01}\right) &= \frac{1}{4}\left[\log_2\left(1+(1-b)\chi_1\right)+(1-b)\chi_1\log_2\left(1+(1-b)\chi_1\right)-2\left(1+(1-b)\chi_1\right)\right],\\
h\left(\chi_{00}\right) &= \frac{1}{4}\left[\log_2\left(1+(1+b)\chi_1\right)+(1+b)\chi_1\log_2\left(1+(1+b)\chi_1\right)-2\left(1+(1+b)\chi_1\right)\right].
\end{aligned}$$

Summing the above equations gives us $H(X_1, X_2)$:

$$\begin{aligned}
H(X_1,X_2) &= -\frac{1}{4}\left[\log_2\left((1-(1+b)^2\chi_1^2)(1-(1-b)^2\chi_1^2)\right)+(1+b)\chi_1\log_2\left(\frac{1+(1+b)\chi_1}{1-(1+b)\chi_1}\right)\right.\\
&\quad \left.+(1-b)\chi_1\log_2\left(\frac{1+(1-b)\chi_1}{1-(1-b)\chi_1}\right)-8\right]. \qquad (58)
\end{aligned}$$

Using the approximations from equations 50, and 51,

$$\begin{aligned}
H(X_1,X_2) &= -\frac{1}{4\ln 2}\left[(1+b)^2\chi_1^2+\frac{1}{6}(1+b)^4\chi_1^4+(1-b)^2\chi_1^2+\frac{1}{6}(1-b)^4\chi_1^4\right]+2,\\
&= -\frac{1}{4\ln 2}\left[\left((1+b)^2+(1-b)^2\right)\chi_1^2+\frac{1}{6}\left((1+b)^4+(1-b)^4\right)\chi_1^4\right]+2,\\
&= -\frac{1}{2\ln 2}\left(1+b^2\right)\chi^2-\frac{1}{12\ln 2}\left(1+6b^2+b^4\right)\chi_1^4+2. \qquad (59)
\end{aligned}$$

The entropy difference D can be computed by substituting equations 53, 54, and 59 into Equation 48:

$$\begin{aligned}
D &= \frac{1}{2\ln 2}\left(1+b^2\right)\chi^2+\frac{1}{12\ln 2}\left(1+6b^2+b^4\right)\chi^4-\frac{1}{2\ln 2}\left(1+b^2\right)\chi^2-\frac{1}{12\ln 2}\left(1+b^4\right)\chi^4,\\
&= \frac{b^2}{2\ln 2}\chi_1^4. \qquad (60)
\end{aligned}$$

Recall that $\chi_1 = \frac{a_1}{2\sqrt{\pi}\sigma_N}$, and $b = a_2/a_1$. Therefore, the above equation can be written in terms of the fitness variance coming from remaining solutions σ_N^2 as

$$D = \frac{1}{32\ln 2}\left(\frac{a_1a_2}{\pi}\right)^2\sigma_N^{-4}. \qquad (61)$$

The above equation can also be written as

$$\log D = -4\log\sigma_N + 2\log\left(\frac{a_1a_2}{4\pi\sqrt{2\ln 2}}\right). \qquad (62)$$

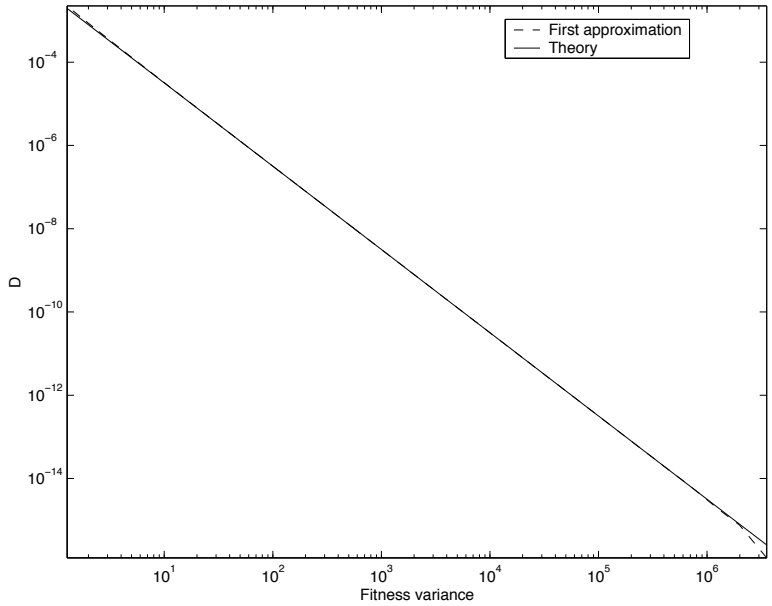


Figure 7: D versus σ_N^2 for onemax.

For the onemax problem, $a_1 = a_2 = 1$ and the above equation reduces to

$$\log D = -4 \log \sigma_N - 2 \log \left(4\pi \sqrt{2 \ln 2} \right) . \tag{63}$$

The above equation is verified with actual values for D in the case of onemax problem in Figure 7.

By substituting Equation 61 into Equation 16, we can imply that if $a_{12} = 0$,

$$N_{crit} = O(\sigma_N^{4.2}), \tag{64}$$

which yields

$$N_{crit} = O(n^{2.1}). \tag{65}$$

In other words, the population size to discover the dependencies between linear variables, which are independent with respect to the fitness function, grows approximately quadratically with the problem size.

4.5 Empirical Results

Figure 8(a) shows the critical population size for onemax fitness function defined in Equation 7. Both the simulation for infinite populations (based on the exact theoretical model for the frequencies using an infinite population) as well as the final approximate result are shown. We can see that the match between the theory and accurate simulation is very good and that the critical population size for discovering a dependency between independent variables increases approximately quadratically with the fitness variance that is proportional to the size

of the problem.

Figure 8(b) shows the critical population size for the trap function of order 5 compared to the empirical results for the simulation with finite and infinite populations. The correlated bits are both selected from one of the trap subfunctions, while the independent bits are selected from two different subfunctions. The infinite-population simulation was again performed according to the accurate theoretical model. The simulation for a finite population was performed by simulating the actual binary tournament selection on a finite population and increasing the population size until the probability of discovering the dependency was more than 95% in 100 independent runs. The exact theoretical results and the approximations match very well. We can also see that the use of a finite population introduces additional noise that increases the population-sizing requirements for a reliable detection of the correct dependencies, but that the growth of the appropriate population size is still approximately linear.

The results in Figure 8(b) also indicate that there is a large range of population sizes that result in a reliable discovery of nonlinear dependencies but still do not introduce unnecessary dependencies between independent variables. Moreover, the range further increases as we increase the problem size.

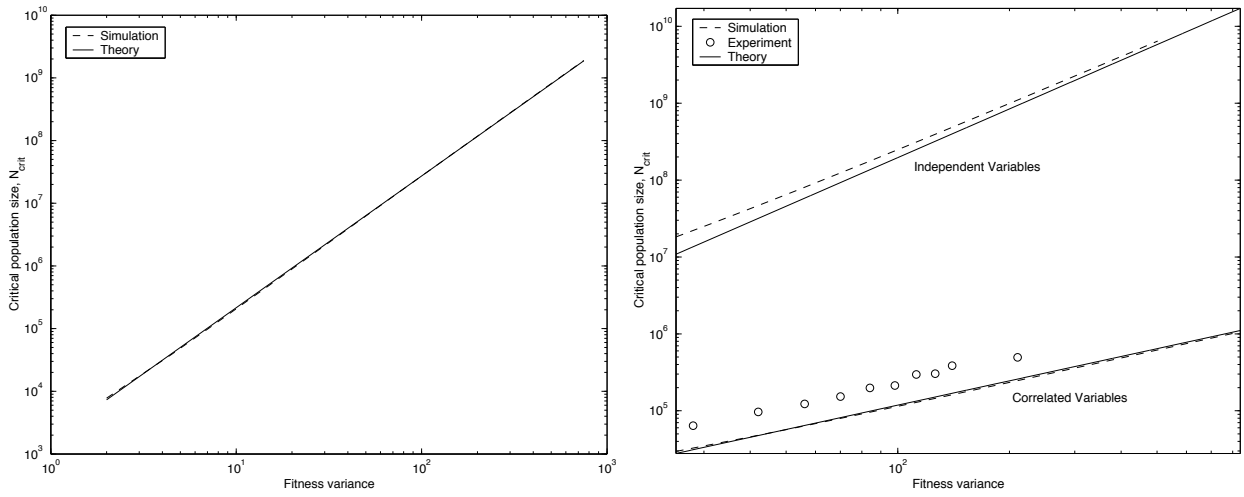
Our theoretical analysis considered only the binary tournament selection. Figure 9(a) indicates that the range of population sizes leading to the discovery of good dependencies but ensuring that the algorithm is not misled by the bad dependencies grows with the selection pressure. The bad news is that the growth of the required population sizes grows slightly faster with increased selection pressures. For the tournament size of $s = 2$, the actual growth of the population size is approximately $O(n^{1.035})$. For the tournament size of $s = 16$, the growth increases to $O(n^{1.242})$. On the other hand, the order of the growth of the population size required to discover the bad dependencies decreases from 1.974 for $s = 2$ to 1.572 for $s = 16$. Nonetheless, as mentioned above, the range of adequate population sizes still increases with the selection pressure.

Figure 9(b) shows that increasing the tournament size up to $s = 16$ decreases the critical population size even for the case of a finite population. However, the positive effects of the selection pressure can be expected to decrease and actually harm the performance of the algorithm in practice due to the premature convergence.

We observed similar results regarding the discovery of dependencies of higher order for both the onemax and trap function. The population sizes required to discover the dependencies of higher order seem to grow even slower than the dependencies of order 2, but we believe that this is merely a consequence of our choice of the fitness functions. For other functions, the behavior may change, but the order of the growth should still remain the same.

4.6 Behavior of K2 Metric

It has been shown that the behavior of Bayesian and MDL metrics is asymptotically the same although Bayesian metrics are known to introduce unnecessary dependencies in practice. These dependencies are often eliminated by either (1) restricting the maximum order of interactions in the model or (2) biasing the prior probabilities of the networks to favor the simpler ones.



(a) Onemax.

(b) Trap of order 5.

Figure 8: Critical population size for onemax and trap function of order 5 for BIC metric.

The theoretical analysis in this paper considered only BIC metric. It is interesting that the behavior of the K2 metric (see Equation 3) is very similar according to the accurate simulation with an infinite population. However, the noise in the considered frequencies due to the finite populations results in many unnecessary dependencies in the real case as we observed in our experiments with BOA with K2 metric (Pelikan, Goldberg, & Sastry, 2001).

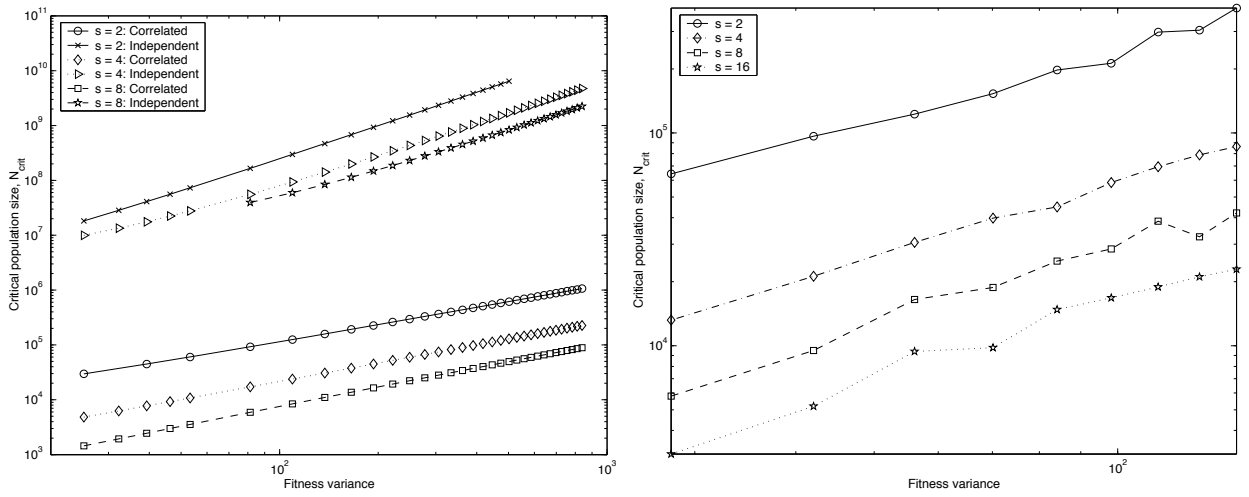
Figure 10 shows the critical population size with respect to the variance of the problem for K2 metric on both onemax and trap problems with binary tournament selection. Comparing the results for K2 metric indicates that K2 metric requires lower population sizes to discover the dependencies. This can be seen as one reason for the preference of more complex models compared to the MDL metric. Another reason for this behavior is the response of the K2 metric to the noise in data.

5 Ramifications for the Scalability of BOA

Let us now combine the results of the above section with the ones presented in Pelikan, Goldberg, and Cantú-Paz (2000a) to determine how BOA scales up on the decomposable problems where the fitness contributions of all the subproblems are scaled the same. We first analyze the results of the previous section to determine the growth of an adequate population size with respect to the size of the problem. Subsequently, we estimate the overall time to convergence by using the convergence-time approximation for BOA presented in Pelikan, Goldberg, and Cantú-Paz (2000a).

There are two important results of the previous section:

1. **Good dependencies.** To discover the good dependencies, the population size grows linearly with the problem size.



(a) Simulation with infinite population.

(b) Experiment with finite population.

Figure 9: The effects of increasing the selection pressure on the critical population size.

2. **Bad dependencies.** To be misled by the bad dependencies, the population size grows quadratically with the problem size.

Additionally, the sufficient population size for discovering the bad dependencies is orders of magnitude bigger than the population size required for discovering the good dependencies. That is a fundamentally important result. First of all, the linear growth of the population size for a good dependency indicates that even if we were to discover all the good dependencies in the first generation of BOA, the growth of the population size would still be linear. Second, there is a large range of adequate population sizes so that the search is still efficient enough and no superfluous dependencies are incorporated in the used model.

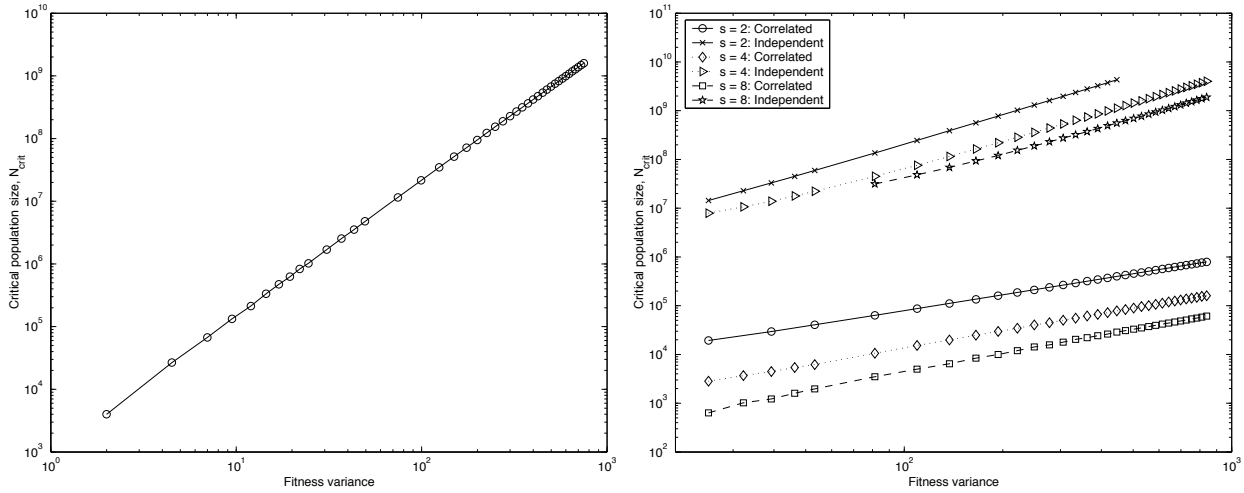
The above result can be used to determine how the adequate population size to solve a problem reliably and accurately grows with the size of the problem. There are three important factors influencing the population sizing in BOA (Pelikan, Goldberg, & Cantú-Paz, 2000a):

1. **Initial supply.** The population must be large enough to ensure that there is a sufficient supply of alternative solutions for each subproblem.

2. **Decision making.** The population must be large enough to ensure that the decision making between the alternative solutions to each subproblem is not misled by the noise from the remaining solutions and that the best partial solution indeed wins.

3. **Model building.** The population must be large enough to ensure that the learned model is correct.

The first two factors are known from the analysis of the population sizing in GAs (Holland, 1975; Goldberg, 1989b; Goldberg, Deb, & Clark, 1992; Harik, Cantú-Paz, Goldberg, & Miller, 1997; Goldberg, Sastry, & Latoza, 2001). However, in all those models the crossover was



(a) Onemax.

(b) Trap of order 5.

Figure 10: Critical population size for onemax and trap function of order 5 for K2 metric.

assumed to mix solutions properly or—in the terminology of BOA—the model was assumed to be correct. The last factor is introduced to ensure that BOA is capable of discovering such a good model and, among other things, the theory of the other factors can be applied.

The population size required for an adequate initial supply remains constant independently of the problem size (Harik, Cantú-Paz, Goldberg, & Miller, 1997). The growth of the population size for the good decision making grows with the square root of the problem size (Harik, Cantú-Paz, Goldberg, & Miller, 1997). Therefore, the population size required to build an accurate model is the primary factor affecting the population sizing in BOA (Pelikan, Goldberg, & Cantú-Paz, 2000a). Therefore, the above theoretical results for the discovery of dependencies among nonlinearly correlated variables can be used to approximate an adequate population size, yielding a bound on the sufficient population size:

$$N_{opt} = O(n^{1.05}), \tag{66}$$

where N_{opt} denotes the sufficient population size, and n is the size of the problem.

However, the population sizing is not the only important factor influencing the scalability of BOA. The total number of evaluations required to find the optimum is given by

$$N_{evals} = cN \times G, \tag{67}$$

where c is the proportion of the population replaced by the offspring at each generation, N is the population size, and G is the total number of generations.

The last piece we must collect to complete the puzzle of BOA scalability is the time to convergence. Fortunately, the number of generations until convergence in BOA can be modeled analogously to the case with the onemax problem and a perfect model for the onemax (the network with no interactions, see Figure 3(a)). In that case, Mühlenbein and Schlierkamp-

Voosen (1993) showed that the number of generations until convergence grows as

$$G = \left(\frac{\pi}{2} - \arcsin\left(2p - 1\right)\right) \frac{\sqrt{n}}{I}, \tag{68}$$

where p is the proportion of ones on each position in the initial generation, n is the problem size, and I is the selection intensity. The selection intensity in generation t is given by

$$I(t) = \frac{\overline{f}(t+1) - \overline{f}(t)}{\sigma(t)}, \tag{69}$$

where $\overline{f}(t+1)$ is the average fitness in the population in generation $t+1$, $\overline{f}(t)$ is the average fitness in generation t, and $\sigma(t)$ is the standard deviation of the fitness values in generation t. For most commonly used selection methods, such as tournament or truncation selection, the selection intensity is constant and the number of generations is therefore bounded by

$$G = O(\sqrt{n}). \tag{70}$$

Although the exact approximation of G given in Equation 68 is correct only for a simple model with no interactions applied to the onemax case, the model can be used to accurately model the convergence time of BOA on many other decomposable problems where the order of each subproblem is bounded by a constant and the contributions of all the subproblems are scaled the same (see Miller and Goldberg (1996) and Pelikan, Goldberg, and Cantú-Paz (2000a)). When the dynamics of the fitness variance is similar to the onemax case, the actual approximation given by Equation 68 approximates the time to convergence very well. Even if this is not the case, the time to convergence can still be accurately approximated by fitting G according to Equation 70. See Figure 11 (from Pelikan, Goldberg, and Cantú-Paz (2000a)) for the time to convergence using BOA with truncation selection with threshold $\tau = 50\%$ that selects the best half of the population in each generation. The empirical results on the trap and onemax fitness functions are compared to the prediction according to Equation 68. In both cases, the results match the original approximation for the onemax with the model with no interactions.

Using equations 66, 67, and 70, the total number of evaluations until convergence to the optimum can therefore be bounded as follows:

$$N_{evals} = O(n^{1.55}), \tag{71}$$

where n is the size of the problem. In other words, the number of evaluations required by BOA to converge to the optimum grows subquadratically with the problem size.

In the above text, we assumed that the contributions of all the subproblems are scaled the same. In practice, there are two extreme cases of scaling the contributions of the different subproblems: (1) uniform scaling, and (2) exponential scaling. How does the situation change in the case of exponential scaling? The important feature of exponential problems is that at any point in time there is at most a few subproblems that matter. Until these subproblems converge, the signal coming from the remaining solutions is negligible. Consequently, the model building becomes somewhat easier because the entire population will be used to

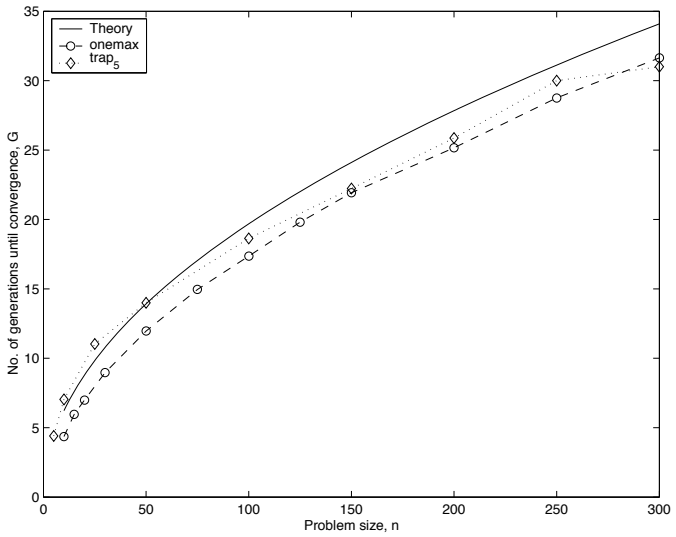


Figure 11: Number of generations until convergence of the BOA with a onemax function, deceptive function of order 3, and a trap function of order 5. The problem sizes range from 0 to 300 bits.

construct a model for a small subset of variables. Yet another consequence of the small signal from the remaining solutions is that the distribution of the variables that do not matter at the moment will be preserved. Therefore, if the variables were independent, they will remain independent. If the variables were correlated, they will remain correlated.

The major issue for sizing the populations in the exponential case thus becomes the loss of diversity in the solutions that do not matter right now but are still subject to a genetic drift. We must ensure that the population size is large enough to preserve enough copies of each possible solution to the least salient problem until the rest of the solution is converged or close to converge. It has been shown that the population sizes ensuring sufficient preservation of the least salient partial solutions grow linearly with the problem size (Goldberg & Segrest, 1987). Thus, the exponential scaling should not have any negative effect on the order of growth of the population size in BOA.

Furthermore, the exponential scaling of the different subproblems causes the number of generation to grow linearly with the number of the subproblems in the decomposition (Thierens, Goldberg, & Pereira, 1998). Assuming that the number of subproblems in the decomposition grows linearly with the problem size, this leads to the following bound on the required number of evaluations:

$$N_{evals}^{X} = O(n^2). \tag{72}$$

To summarize both results from equations 71 and 72, the overall number of evaluations until convergence on decomposable problems of bounded difficulty is expected to be somewhere between $O(n^{1.5})$ to $O(n^2)$, where n is the size of the problem. If the difficulty and order of the subproblems increases with the problem size, the above estimates may further increase.

6 Summary and Conclusions

The paper first discussed the importance of bias for making the search tractable. The primary focus was on what we call the decompositional bias that decomposes the problem into a number of quasi-independent subproblems and utilizes the decomposition to ensure efficient search. The probabilistic model-building genetic algorithms were discussed as a promising line of research toward methods capable of both learning and utilization of the decompositional bias.

Bayesian optimization algorithm (BOA) was described. BOA uses general Bayesian networks to model promising solutions found so far and guide further search. The learned model can encode conditional dependencies and independencies. The dependencies lead to preservation of alternative partial solutions to the different subproblems in the decomposition. The independencies ensure efficient search by decomposing the entire problem into a number of quasi-independent subproblems.

The methods for learning Bayesian networks were discussed in context of optimization in BOA. The paper analyzed the effect of the tournament selection operator on the discovery of the good dependencies covering nonlinear interactions in the problem as well as the bad dependencies between independent variables. The results showed that the population sizes required for discovering the true dependencies grow approximately linearly with the problem size while the population sizes that mislead the model building and lead to the discovery of dependencies between independent variables grow approximately quadratically and are orders of magnitude higher. Consequently, there is a large range for setting a proper population size.

The different facets of the BOA and GA theory were combined to approximate the scalability of BOA on decomposable problems of bounded difficulty with uniformly scaled subproblems. The extension of the results to the problems with nonuniformly scaled subproblems was discussed. The final result claims that the number of evaluations until convergence should grow subquadratically or quadratically with the problem size, depending on the scaling of the subproblems. If the difficulty or order of the involved subproblems were to increase with the problem size, the overall complexity of BOA could grow faster.

By combining the achievements of genetic and evolutionary computation with the advanced methods of machine learning and probabilistic modeling, BOA offers a method to solve difficult optimization problems quickly, accurately, and reliably. The number of evaluations until reliable convergence to the optimum grows subquadratically or quadratically with the size of the problem. There is a very large sweet spot for setting a proper population size for different selection pressures to ensure fast and reliable convergence to the optimum. Although the practitioner may not know all the parameters involved in the computation of the adequate population size, the algorithm is robust enough to accommodate for the uncertainty in estimating those parameters.

Acknowledgments

The authors would like to thank David M. Chickering, Peter Grünwald, Henry Tirri, and Erick Cantú-Paz for interesting insights and discussions.

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel

Command, USAF, under grant F49620-00-1-0163. Research funding for this work was also provided by 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. Martin Pelikan was partially supported by grant VEGA 1/7654/20 of the Slovak Grant Agency. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

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

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning* (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA: Carnegie Mellon University.

Baluja, S., & Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proc. 14th International Conference on Machine Learning* (pp. 30–38). Morgan Kaufmann.

Bosman, P. A., & Thierens, D. (2000). *Mixed ideas* (Utrecht University Technical Report UU-CS-2000-45). Utrecht, Netherlands: Utrecht University.

Bosman, P. A. N., & Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, Volume I (pp. 60–67). Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA.

Chickering, D. M., Geiger, D., & Heckerman, D. (1994). *Learning Bayesian networks is NP-hard* (Technical Report MSR-TR-94-17). Redmond, WA: Microsoft Research.

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

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

De Bonet, J. S., Isbell, C. L., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In Mozer, M. C., Jordan, M. I., & Petsche, T. (Eds.), *Advances in neural information processing systems*, Volume 9 (pp. 424). The MIT Press, Cambridge.

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

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

Etxeberria, R., & Larrañaga, P. (1999). Global optimization using Bayesian networks. In Rodriguez, A. A. O., Ortiz, M. R. S., & Hermida, R. S. (Eds.), *Second Symposium on Artificial Intelligence (CIMAF-99)* (pp. 332–339). Habana, Cuba: Institude of Cybernetics, Mathematics, and Physics and Ministry of Science, Technology and Environment.

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

Goldberg, D. E. (1989b). Sizing populations for serial and parallel genetic algorithms. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 70–79). San Mateo, CA: Morgan Kaufmann. (Also IlliGAL Report No. 88004).

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

Goldberg, D. E., Sastry, K., & Latoza, T. (2001). On the supply of building blocks. In Spector, L., et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 336–342). San Francisco, CA: Morgan Kaufmann. (Also IlliGAL Report No. 2001015).

Goldberg, D. E., & Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. In Grefenstette, J. J. (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 1–8). Hillsdale, NJ: Lawrence Erlbaum Associates.

Grünwald, P. (1998). *The minimum description length principle and reasoning under uncertainty.* Doctoral dissertation, University of Amsterdam, Amsterdam, Netherlands.

Harik, G. R., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In *Proceedings of the International Conference on Evolutionary Computation 1997 (ICEC '97)* (pp. 7–12). Piscataway, NJ: IEEE Press. Also IlliGAL Report 96004.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. In *Proceedings of the International Conference on Evolutionary Computation 1998 (ICEC '98)* (pp. 523–528). Piscataway, NJ: IEEE Press.

Heckerman, D., Geiger, D., & Chickering, D. M. (1994). *Learning Bayesian networks: The combination of knowledge and statistical data* (Technical Report MSR-TR-94-09). Redmond, WA: Microsoft Research.

Heckerman, D., Mamdani, A., & Wellman, M. P. (1995). Real-world applications of Bayesian networks. *Communications of the ACM*, *38*(3), 24–30.

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

Howard, R. A., & Matheson, J. E. (1981). Influence diagrams. In Howard, R. A., & Matheson, J. E. (Eds.), *Readings on the principles and applications of decision analysis*, Volume II (pp. 721–762). Menlo Park, CA: Strategic Decisions Group.

Larranaga, P., Etxeberria, R., Lozano, J. A., & Pena, J. M. (2000). Optimization in continuous domains by learning and simulation of Gaussian networks. In Wu, A. (Ed.), *Workshop proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000* (pp. 201–204). San Fransisco, CA: Morgan Kaufmann.

Miller, B. L., & Goldberg, D. E. (1996). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, *4* (2), 113–131.

Mühlenbein, H., Mahnig, T., & Rodriguez, A. O. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, *5*, 215–247.

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

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

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

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. (2000a, 10-12 July). Bayesian optimization algorithm, population sizing, and time to convergence. In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., & Beyer, H.-G. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)* (pp. 275–282). Las Vegas, Nevada: Morgan Kaufmann. Also IlliGAL Report No. 2000001.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000b). Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation*, *8* (3), 311–341. Also IlliGAL Report No. 98013.

Pelikan, M., Goldberg, D. E., & Lobo, F. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, *21* (1), 5–20. Also IlliGAL Report No. 99018.

Pelikan, M., Goldberg, D. E., & Sastry, K. (2001, 7-11 July). Bayesian optimization algorithm, decision graphs, and Occam's razor. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., & Burke, E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 519–526). San Francisco, CA: Morgan Kaufmann.

Pelikan, M., Goldberg, D. E., & Tsutsui, S. (2000). *Combining the strengths of the Bayesian optimization algorithm and adaptive evolution strategies* (IlliGAL Report No. 2001023). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In Roy, R., Furuhashi, T., & Chawdhry, P. K. (Eds.), *Advances in Soft Computing - Engineering Design and Manufacturing* (pp. 521–535). London: Springer-Verlag.

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

Rissanen, J. J. (1978). Modelling by shortest data description. *Automatica*, *14*, 465–471.

Rissanen, J. J. (1989). *Stochastic complexity in statistical inquiry*. Singapore: World Scientific Publishing Co.

Rissanen, J. J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, *42*(1), 40–47.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*, 461–464.

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

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

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.

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