Bivariate Estimation-of-Distribution Algorithms Can Find an Exponential Number of Optima

Benjamin Doerr
Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris
Palaiseau, France
doerr@lix.polytechnique.fr

Martin S. Krejca
Hasso Plattner Institute
Potsdam, Germany
martin.krejca@hpi.de

ABSTRACT

Finding a large set of optima in a multimodal optimization landscape is a challenging task. Classical population-based evolutionary algorithms (EAs) typically converge only to a single solution. While this can be counteracted by applying niching strategies, the number of optima is nonetheless trivially bounded by the population size.

Estimation-of-distribution algorithms (EDAs) are an alternative, maintaining a probabilistic model of the solution space instead of an explicit population. Such a model is able to implicitly represent a solution set that is far larger than any realistic population size.

To support the study of how optimization algorithms handle large sets of optima, we propose the test function EqualBlocksOneMax (EBOM). It has an easy to optimize fitness landscape, however, with an exponential number of optima. We show that the bivariate EDA *mutual-information-maximizing input clustering* (MIMIC), without any problem-specific modification, quickly generates a model that behaves very similarly to a theoretically ideal model for that function, which samples each of the exponentially many optima with the same maximal probability.

CCS CONCEPTS

• **General and reference** → **Empirical studies**; *Experimentation.*

KEYWORDS

Estimation-of-distribution algorithms; probabilistic model building; empirical study

ACM Reference Format:
Benjamin Doerr and Martin S. Krejca. 2020. Bivariate Estimation-of-Distribution Algorithms Can Find an Exponential Number of Optima. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390177

1 INTRODUCTION

A key feature of evolutionary algorithms (EAs) is their applicability to a wide range of optimization problems. EAs require little problem-specific knowledge and generally provide the user with a good solution. Since many real-world optimization problems are multimodal [1, 11, 23], it is desirable for an EA to return multiple solutions. This way, the user also gains precious insight into their problem.

Unfortunately, classical population-based EAs tend to converge to a single solution, due to strong selection operators and due to a long-known phenomenon called *genetic drift* [4]. In order to counteract this behavior, different techniques have been introduced, commonly subsumed under the term *niching* [15, 17, 23]. These techniques maintain diversity in the population and assist in finding and keeping multiple good solutions. While this approach is useful for increasing the number of different solutions, it still limits the insights gained about the underlying problem, as the only information the EA returns is the solutions themselves. As such, it only provides information about areas of the search space that it has visited and does not propose further promising regions.

A different algorithmic approach that aims to additionally incorporate information about the entire search space is the framework of *estimation-of-distribution algorithms* (EDAs; [21]). Instead of an explicit set of solutions, EDAs maintain a probabilistic model of the search space. This model acts as a solution-generating mechanism and reflects information about which parts of the search space seem more favorable than others. An EDA evolves its model based on samples drawn from it. This way, the model is refined such that it generates better solutions with higher probability. In the end, an EDA returns the best solutions found as well as its model.

EDAs are commonly classified by the power of their model [21]. This results in the following trade-off: an EDA with a simple model performs an update quickly but may be badly suited to accurately represent the distribution of good solutions. In contrast, the update of an EDA with a complex model is computationally expensive, but the model is better capable of representing good solutions. The complexity of a model is determined by how many dependencies it can detect among different problem variables. For example, a *univariate* EDA assumes independence of all problem variables, whereas a *bivariate* EDA can represent dependencies among pairs of variables. We go into detail about these types of EDAs in Section 1.1.

While increasing the complexity of an EDA's model is useful for finding optima in a larger class of problems [20], it is not evident that an increased model complexity is also useful for finding *multiple optima* or representing them adequately in the model. In fact, EDAs have been designed specifically with the intention of being used for multimodal optimization. Peña et al. [18] introduce the *unsupervised estimation of Bayesian network algorithm* (UEBNA), which uses unsupervised learning in order to generate the Bayesian network of its model. The algorithm is tested against other EDAs and evaluated

GECCO '20, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390177

(mostly) on bisection problems on graphs with many symmetries that only have a low number of optima (two to six). Interestingly, for the larger problems, even UEBNA is not able to find all optima. Thus, the test functions seem to be hard, and the experiments do not only show how many optima the algorithm can find but also how well it copes with hard problems.

A similar setting has been considered by Chuang and Hsu [3], who introduce an EDA that is also specifically tailored toward multimodal optimization. However, they evaluate their results only on trap functions with a low number of optima (two to four). Thus, the focus of their work is arguably also more on the hardness of the problem than on finding many optima.

Hauschild et al. [10] consider the *hierarchical Bayesian optimization algorithm* and analyze how well its model reflects the problem structure of two hard test functions. They show that the structure is best reflected during the middle of the run and that it is then simplified toward the end. This makes sense, as the model aims to reflect best how to generate *optimal* solutions. This does not need to coincide with how the *entire* structure can be reflected. For example, if the problem has a single solution, it suffices to have a simple model that only generates this solution in a straightforward way. Again, the focus of the authors is rather based on the hardness of the problem (its structure) instead of representing many optima.

Overall, to the best of our knowledge, there are no results dedicated to finding many different optima on a function easy enough so that finding an optimum at all is not already a challenge.

In this work, we introduce the test function EqualBlocksOneMax (EBOM, Section 2.3), which has an exponential number of optimal solutions. It is easy in the sense that all local optima are also global optima. We are interested in how well the underlying structure of the optima can be detected by an algorithm.

Univariate EDAs apparently are not suitable to return a model that represents the exponential number of optima of EBOM. EAs can find at most as many different optima as the size of their population. Depending on the actual size, this may be a large number but will be polynomial for any reasonable run time of the algorithm. This is still insignificant to the total number of optima of EBOM.

We show that *mutual-information-maximizing input clustering* (MIMIC; [2]), arguably the simplest bivariate EDA, is represents the structure of EBOM well. It builds a model that behaves very similarly to an ideal model for EBOM, which creates all optimal solutions with the maximal probability possible. Our experiments (Section 3) show that, for almost all input sizes we consider, MIMIC samples about $1 \cdot 10^4$ to $4.5 \cdot 10^4$ optima per run and never samples an optimum twice. As EBOM can be described by a bivariate model, our results suggest that bivariate EDAs are well suited to reasonably capture the set of all optima for functions they can optimize.

Following, we discuss different types of EDAs in order to explain how common probabilistic models look like and why univariate models are unsuited for representing multiple optima. In Section 2, we present MIMIC, the definition of EBOM, and what an ideal model for EBOM is. In Section 3, we explain our test setup and discuss our results. We conclude our paper in Section 4.

1.1 Types of EDAs

A common way of classifying EDAs is with respect to how they decompose a problem [21]. Such a decomposition is typically based on representing a probability distribution over the search space as a product of various probabilities that may share dependencies. This information is stored compactly by a probabilistic graphical model (PGM; [12]). As the name suggests, PGMs use graphs for storing information about probability distributions, where variables are represented as nodes and dependencies as edges. The arguably best known type of PGM are Bayesian networks (BNs).

A BN can be represented as a directed acyclic graph. A directed edge from x to y represents that y is dependent on (at least) x. Each node stores a probability distribution conditional on the outcomes of all of its predecessors. A solution according to the probability distribution of the BN can be sampled by traversing the graph in a topological order, always determining the outcome of an input variable based on the outcome of its predecessors. The larger the in-degree of a node in a BN can become, the more costly it is to represent the model, as the conditional probability distribution for each node can grow quite large. Thus, the number of dependencies in the models of EDAs are usually restricted.

1.1.1 Univariate EDAs. The BN of a univariate EDA is an independent set. That is, each node represents a probability distribution based solely on a single variable. Hence the name *uni*-variate. Examples of univariate EDAs are the *compact genetic algorithm* [8] and the *univariate marginal distribution algorithm* [16].

When optimizing functions over bit strings, the probability of each binary input variable tends to either 0 or 1 rather quickly [6, 7], forcing the model to put its probability mass onto a single solution. Thus, univariate EDAs are ill-suited to represent multiple solutions at once. For more theoretical investigations on this topic, please refer to a recent survey by Krejca and Witt [13].

1.1.2 Bivariate EDAs. In a bivariate EDA, each problem variable can be dependent on at most one other variable. Examples of bivariate EDAs are *mutual-information-maximizing input clustering* [2] and the *bivariate marginal distribution algorithm* [22].

Recently, Lehre and Nguyen [14] showed that MIMIC may have a huge advantage over univariate EDAs on deceptive functions, but this may be a consequence of a suboptimal parameter choice [5].

Since a bivariate model can store simple dependencies, it is capable to represent multiple solutions at once. Further, the model can still be built somewhat efficiently, as there is at most a quadratic number of possible dependencies to consider when building the model. Thus, we focus on bivariate EDAs in this work.

1.1.3 Multivariate EDAs. This type is used as an umbrella term for any type of EDA that is able to represent some form of dependency. While the models of such EDAs can perform well on deceptive, hard functions, creating a model can be computationally expensive, as potentially many dependencies need to be checked. Examples of multivariate EDAs are the *extended compact genetic algorithm* [9] and the *hierarchical Bayesian optimization algorithm* [19].

2 PRELIMINARIES

In this section, we introduce some notation that we use throughout the paper as well as the algorithm and the test function that we consider in our analysis in Section 3.

2.1 Notation

Let $\mathbb{N}$ denote the set of all natural numbers, including 0. For $a, b \in \mathbb{N}$, let $[a..b] := [a, b] \cap \mathbb{N}$ denote the set of all natural

numbers from a to b (including both bounds). As a special case of that notation, for $b \in \mathbb{N}$, let $[b] := [1..b]$ denote the set of all positive natural numbers up to b. For an $n \in \mathbb{N}$, let id_n denote the identity function over $[n]$.

For a logical proposition P, let $\mathbb{1}\{P\}$ denote the indicator function of the truth value of P, that is, $\mathbb{1}\{P\} = 1$ if P is true, and it is 0 otherwise.

We consider pseudo-Boolean optimization, that is, optimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$. We call such a function *fitness function*. We call a bit string $x \in \{0,1\}^n$ an *individual* and $f(x)$ the *fitness of* x. If not stated otherwise, let f always denote a fitness function, and let n always denote its dimension.

2.2 Mutual-Information-Maximizing Input Clustering (MIMIC)

Mutual-information-maximizing input clustering (MIMIC; [2]) is a bivariate estimation-of-distribution algorithm (EDA). The Bayesian network of the probabilistic model of MIMIC can be represented as a directed path over n nodes, where each of the nodes corresponds to one of the n bit positions of f. Further, MIMIC has two parameters, $\lambda, \mu \in \mathbb{N}$ with $\lambda \geq \mu$, that represent how many individuals are generated and selected each iteration, respectively.

Initially, the model represents the uniform distribution. It is rebuilt each iteration in the following way: first, λ individuals are generated according to the current model, and μ individuals are selected according to some selection mechanism. We call the resulting (multi-)set S. A path is constructed greedily based on the entropy of the distribution of the bits at the different positions in S.

The first node of the new path is a position with the lowest entropy, that is, a position with the largest number of 1s or 0s. Each subsequent node is chosen with respect to the lowest entropy conditional on the distribution of the current last node in the path. This way, the new path represents a model that best reflects the distributions of pairs of positions observed in S. We now go into detail about our implementation of MIMIC (Algorithm 1).

2.2.1 Probabilistic model and sampling. For our implementation of MIMIC, we describe the probabilistic model via a permutation π (over $[n]$) and an $n \times 2$ matrix of probabilities. Bit strings are sampled bit by bit in the order of π. For a position $i \in [2..n]$ and a bit value $b \in \{0,1\}$, an entry $P_{\pi(i),b}$ denotes the probability to sample a 1 at position $\pi(i)$, given that the bit at position $\pi(i-1)$ is b. Note that entries in P always denote the probability to sample a 1. For the position $\pi(1)$ (which does not have a predecessor in π), we set $P_{\pi(1),0} = P_{\pi(1),1}$. Thus, either entry denotes the probability to sample a 1 without a prior.

For a bit string $x \in \{0,1\}^n$, we write $x \sim \mathrm{sample}_\pi(P)$ to denote that x is being sampled with respect to the probabilistic model consisting of π and P. More formally the sampling procedure creates x such that, for any bit string $y \in \{0,1\}^n$,

$$\Pr[x = y] = (P_{\pi(1),0})^{y_{\pi(1)}} \cdot (1 - P_{\pi(1),0})^{1-y_{\pi(1)}} \cdot \prod_{\substack{i \in [2..n]: \\ y_{\pi(i)}=0}} (1 - P_{\pi(i),y_{\pi(i-1)}}) \cdot \prod_{\substack{i \in [2..n]: \\ y_{\pi(i)}=1}} P_{\pi(i),y_{\pi(i-1)}}.$$

2.2.2 Selection. Given a population $O \subseteq \{0,1\}^n$ of individuals and a fitness function f, we write $\mathrm{select}_\mu(O, f)$ to denote a selection mechanism that selects μ individuals from O. In this paper, we use *truncation selection*, that is, we sort the individuals in O by fitness and then select the μ best individuals (breaking ties uniformly at random).

Algorithm 1: MIMIC [2] with parameters μ and λ, $\mu \leq \lambda$, and a selection scheme select_μ, optimizing a fitness function $f\colon \{0,1\}^n \to \mathbb{R}$ with $n \geq 2$.

$t \leftarrow 0$;
$\pi^{(t)} \leftarrow \mathrm{id}_n$;
$P^{(t)} \leftarrow (\frac{1}{2})_{i \in [n], b \in \{0,1\}}$;
repeat
  $O^{(t)} \leftarrow \emptyset$;
  for $i \in [\lambda]$ **do**
    $x^{(i)} \sim \mathrm{sample}_{\pi^{(t)}}(P^{(t)})$;
    $O^{(t)} \leftarrow O^{(t)} \cup \{x^{(i)}\}$;
  $S^{(t)} \leftarrow \mathrm{select}_\mu(O^{(t)}, f)$;
  $I \leftarrow [n]$;
  $\pi^{(t+1)}(1) \leftarrow \arg\min_{i \in I} h[S^{(t)}; i]$;
  $I \leftarrow I \setminus \{\pi^{(t+1)}(1)\}$;
  for $b \in \{0,1\}$ **do** $P^{(t+1)}_{\pi^{(t+1)}(1),b} \leftarrow \gamma_1[S^{(t)}; \pi^{(t+1)}(1)]$;
  for $j \in [2..n]$ **do**
    $\pi^{(t+1)}(j) \leftarrow \arg\min_{i \in I} h[S^{(t)}; i \mid \pi^{(t+1)}(j-1)]$;
    $I \leftarrow I \setminus \{\pi^{(t+1)}(j)\}$;
    for $b \in \{0,1\}$ **do**
      $P^{(t+1)}_{\pi^{(t+1)}(j),b} \leftarrow \gamma_{1b}[S^{(t)}; \pi^{(t+1)}(j) \mid \pi^{(t+1)}(j-1)]$;
  restrict all values of $P^{(t+1)}$ to the interval $[\frac{1}{n}, 1 - \frac{1}{n}]$;
  $t \leftarrow t + 1$;
until termination criterion met;

2.2.3 Building the probabilistic model. When constructing a new probabilistic model, MIMIC makes use of the unconditional and conditional (empirical) entropy of a set of bit strings. These mathematical functions make use of the relative occurrences of bit values. To this end, for a population $S \subseteq \{0,1\}^n$, a position $i \in [n]$, and a bit value $b \in \{0,1\}$, let the frequency of b at position i in S be

$$\gamma_b[S; i] = \frac{1}{|S|} \sum_{x \in S} \mathbb{1}\{x_i = b\}.$$

Further, for a population $S \subseteq \{0,1\}^n$, two positions $i, j \in [n]$, and two bit values $b_1, b_2 \in \{0,1\}$, we define the conditional frequency of b_1 at position i in O conditional on the value b_2 at position j by

$$\gamma_{b_1 b_2}[S; i \mid j] = \begin{cases} \frac{1}{2} & \text{if } \gamma_{b_2}[S; j] = 0, \\ \frac{1}{|S| \cdot \gamma_{b_2}[S;j]} \sum_{x \in S} \mathbb{1}\{x_i = b_1 \wedge x_j = b_2\} & \text{else.} \end{cases}$$

Note that the case $\gamma_{b_2}[S; j] = 0$ means that the event we condition on has a probability of 0, which is not well defined. In order to represent our lack of knowledge in this case, we choose $\frac{1}{2}$ as the value for the respective probability, which corresponds to a uniform distribution.

We now define the (empirical) entropy functions that MIMIC utilizes. To this end, we define that $0 \cdot \log_2(0) = 0$. For a population

$S \subseteq \{0,1\}^n$ and a position $i \in [n]$, the entropy at position i in S is

$$h[S;i] = -\sum_{b \in \{0,1\}} \gamma_b[S;i] \cdot \log_2(\gamma_b[S;i]).$$

Further, for a population $S \subseteq \{0,1\}^n$ and two positions $i, j \in [n]$, the entropy at position i in O conditional on position j is

$$h[S;i \mid j] = -\sum_{(b_1,b_2) \in \{0,1\}^2} \gamma_{b_1 b_2}[S;i \mid j] \cdot \gamma_{b_2}[S;j] \cdot \log_2(\gamma_{b_1 b_2}[S;i \mid j]).$$

Given these definitions and a population $S \subseteq \{0,1\}^n$ of selected individuals, MIMIC builds a new model by constructing a new permutation π' and updating the probabilities in P with respect to π'. The permutation π' is built in the following iterative and greedy fashion, breaking ties uniformly at random: for the first position, an index with the lowest entropy in S is chosen. Each subsequent position is determined by an index with the lowest entropy in S conditional on the previous index in π'.

Each time that a new position i is determined for π', the probabilities $P_{i,0}$ and $P_{i,1}$ are updated. If $i = \pi'(1)$, both $P_{i,0}$ and $P_{i,1}$ are set to the relative number of 1s at position i in S, that is $\gamma_1[S;i]$. If $i \neq \pi'(1)$, that is, there is a preceding position j in π', for a bit value b, the probability $P_{i,b}$ is set to the relative number of 1s at position i in S that also have a value of b at position j. Note that this is equivalent to setting $P_{i,b}$ to $\gamma_{1b}[S;i \mid j]$.

In order to circumvent the model from sampling only 0s or only 1s at some position, we make sure that no probability is 0 or 1. We enforce this after building π' and updating P by increasing probabilities less than $\frac{1}{n}$ to $\frac{1}{n}$ and by decreasing probabilities greater than $1-\frac{1}{n}$ to $1-\frac{1}{n}$. We may also say that we restrict P to the interval $[\frac{1}{n}, 1-\frac{1}{n}]$.

Note that restricting the probabilities makes it necessary to define a value for the first case in the definition of $\gamma_{b_1 b_2}$, since it can happen that $\gamma_{b_2}[S;j] = 0$ in S, but the corresponding probability is not 0, as it is restricted to $[\frac{1}{n}, 1-\frac{1}{n}]$. In such a case, it is possible to sample b_2 with the new model, making it necessary to define the probability P_{i,b_2}.

2.3 EqualBlocksOneMax (EBOM)

Many benchmark functions test an algorithm's capability of finding an optimal solution at all. Hence, they are commonly composed of deceptive or otherwise hard landscapes with many dependencies. In order to reduce the probability of finding an optimal solution by pure chance, the number of optima of such a function is usually small. For EDAs, it is not only interesting how fast they find good solutions but also how well their probabilistic model represents the distribution of good solutions in the search space.

To this end, we introduce the test function EBOM. It represents a fairly simple hill-climbing landscape, similar to that of the well-known OneMax function (the sum of all bit values in an individual), but features an exponential number of optima. Thus, finding a single optimal solution is easy, but exploiting the structure of EBOM and being able to generate a large number of *different* optima is challenging.

2.3.1 Definition. Given a bit string of length n, EBOM operates on blocks of size 2 and returns the number of blocks that are either 00 or 11. Let n be even. For each $j \in [\frac{n}{2}]$, let the pair of positions $2j-1$ and $2j$ denote block j. For an individual $x \in \{0,1\}^n$, we say that block j is *correct* if the bits in block j have identical values. The objective of EBOM is to maximize the number of correct blocks. Formally, for all $x \in \{0,1\}^n$,

$$\text{EBOM}(x) = \sum_{j=1}^{n/2} \mathbb{1}\{x_{2j-1} = x_{2j}\}.$$

Consequently, EBOM has a maximal fitness of $\frac{n}{2}$ and $2^{n/2}$ different optima, since there are two possibilities for each of the $\frac{n}{2}$ blocks to be correct.

2.4 An Ideal Model of MIMIC for EBOM

We are interested in a model of MIMIC that generates each optimal solution of EBOM with the same maximal probability. We call such a model *ideal*.

The permutation π of an ideal model is such that, for each block $j \in [\frac{n}{2}]$ of EBOM, the positions $2j-1$ and $2j$ are adjacent in π (but in any order). In the following, assume without loss of generality that $\pi(2j-1) < \pi(2j)$, that is, position $2j-1$ occurs before $2j$ in π. For the probability matrix P of an ideal model, the probabilities of position $2j-1$ are both $\frac{1}{2}$, and the probabilities of position $2j$ are $1-\frac{1}{n}$ (conditional on a prior 1) and $\frac{1}{n}$ (conditional on a prior 0). Note that, when sampling a solution with an ideal model, the bit sampled at position $2j$ is sampled conditional on the bit at position $2j-1$. Due to the choice of P, this probability is maximized. Choosing $\frac{1}{2}$ as the value of *both* probabilities of position $2j-1$ further ensures two things: (1) The bit at position $2j-1$ is sampled independently of the bit at position $2j-2$.[1] (2) Block j is 00 or 11 with equal probability. Overall, an ideal model has maximal equal probability to sample an optimum. We now discuss features that help in assessing whether a model is close to an ideal model or not.

In an ideal model, the probability that a generated bit string is one of the $2^{n/2}$ optima is $2^{n/2}\left(\frac{1}{2}(1-\frac{1}{n})\right)^{n/2} = \left((1-\frac{1}{n})^n\right)^{1/2}$. Using that $\lim_{n\to\infty}(1-\frac{1}{n})^n = \frac{1}{e}$, the probability of MIMIC to sample an optimal solution, given an optimal model, is roughly $1/\sqrt{e} \approx 60.65\,\%$. However, note that the probability of $1/\sqrt{e}$ of sampling an optimum is, by itself, *not* indicative of an ideal model. This probability is also achieved by any other model which is like an ideal model but has the following difference: for each block j (defined as above), the probabilities at position $2j-1$ are equal but not necessarily $\frac{1}{2}$. Given such a model, the probability to sample *any* optimum is still $1/\sqrt{e}$. However, the probability to sample a *specific* optimum may differ from optimum to optimum. Consequently, we also consider a second indicator for an ideal model.

The property of an ideal model that *each* optimum has the same probability of being sampled makes it unlikely that such a model creates duplicate solutions in $m \in \mathbb{N}^+$ independent tries. More formally, for an optimal model, since each optimum is equally likely, the probability that all optima are distinct when sampling m optimal solutions is $(2^{n/2})!/\left(2^{mn/2} \cdot (2^{n/2}-m)!\right)$, by the birthday paradox. This probability is at least $(1-m/2^{n/2})^m \geq 1-m^2/2^{n/2}$, by Bernoulli's inequality, which is close to 1 as long as $m^2 = o(2^{n/2})$.

[1] For this to hold, it suffices that both probabilities of position $2j-1$ are the same; they do not have to be $\frac{1}{2}$.

We conclude from these insights that a good model of MIMIC should sample optima with a probability of roughly $1/\sqrt{e}$ and that it should not sample duplicates, with high probability.

3 RESULTS

In this section, we show that MIMIC creates models in reasonable time that behave similarly to an ideal model for EBOM. We first explain our setup, then we discuss our results.

3.1 Algorithm Setup

We use MIMIC as seen in Algorithm 1 with truncation selection (with uniform tie-breaking) and with $\lambda = \lfloor 12n \ln n \rfloor$ and $\mu = \lfloor \lambda/8 \rfloor$. Our choice for λ is based on a grid search for the exponent of the n-factor in the interval $[0.5, 1]$ with a step size of 0.1. The exponent of 1 was the first with that MIMIC found an optimum in all runs of our test setup (see also Section 3.2). For μ, we chose a constant fraction of λ, which is common for EDAs.

3.2 Test Setup

We are interested in determining how well MIMIC is capable of generating a probabilistic model that implicitly captures an exponential number of optima of EBOM. Consequently, we use our insights from Section 2.4 in order to determine how good a model of MIMIC is. To this end, we let MIMIC run for a number of iterations I, which we explain below, and we determine

(1) the probabilistic model (that is π and P) in each iteration,
(2) with what probability optimal solutions are created in each iteration, and
(3) how many distinct optima are created in I iterations.

Our choice of I is as follows: let T denote the number of iterations until MIMIC samples an optimum for the first time. Then we let the algorithm run for T more iterations, that is, $I = 2T$. Since MIMIC may fail finding an optimum in a reasonable time, we abort a run if the number of iterations exceeds 50 000 iterations. However, we chose λ and μ such that *all* of our tests were successful. That is, MIMIC always found an optimum, and we let the algorithm run for $2T$ iterations.

We consider MIMIC for values of n from 50 to 200 in steps of 10. For each value of n, we start 100 independent runs. For each run, we record the number of iterations until the first optimum is sampled (that is, T), the set of *all* optima that are found in each of the $2T$ total iterations (which may include duplicates), the number of optima found in each of these iterations, as well as the probabilistic model in each iteration. Note that with this data we are able to compute the information above we are interested in.

3.2.1 Visualization. We depict our results in Figures 1 to 4 and in Tables 1 and 2. In these plots, we visualize:

(1) the total number of iterations and fitness evaluations,
(2) how the probabilistic model evolves during a run,
(3) the number of optima found as well the number of runs that only found distinct optima, and
(4) the probability of sampling an optimum during an iteration.

For each figure, we plot the data of all 100 runs (per n) simultaneously in a concise manner: we depict the median of the data as a point and connect the medians with a solid line. Further, we depict the mid 50 % (that is, ranks 25 to 75 when ordering the runs) as a shaded area bounded by a dotted line. We provide more information about the visualization in the discussion of our results.

Table 1: The probabilities of the first 10 positions (occurring in π) of MIMIC optimizing EBOM for one of the runs with $n = 200$, at iteration $2T$. For a discussion of this table, please refer to Section 3.3.2.

position i	$P_{i,0}$	$P_{i,1}$
17	0.662052	0.662052
18	0.005	0.995
90	0.636872	0.610266
89	0.005	0.995
41	0.649587	0.58435
42	0.005	0.995
49	0.68438	0.546488
50	0.005	0.995
104	0.582677	0.613208
103	0.005	0.995
	⋮	

3.3 Discussion

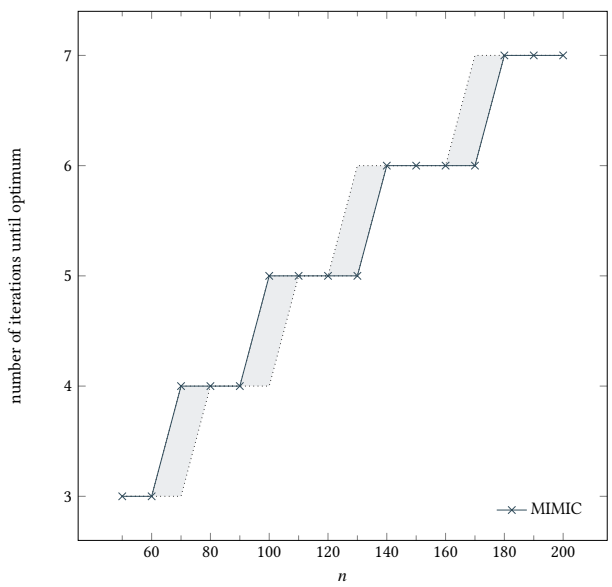
In this section, we discuss the results depicted in Figures 1 to 4 and in Tables 1 and 2.

3.3.1 Run time. Figure 1 shows the run time of each of the 100 runs per n with respect to the number of iterations (Figure 1a) and with respect to the number of fitness function evaluations (Figure 1b).

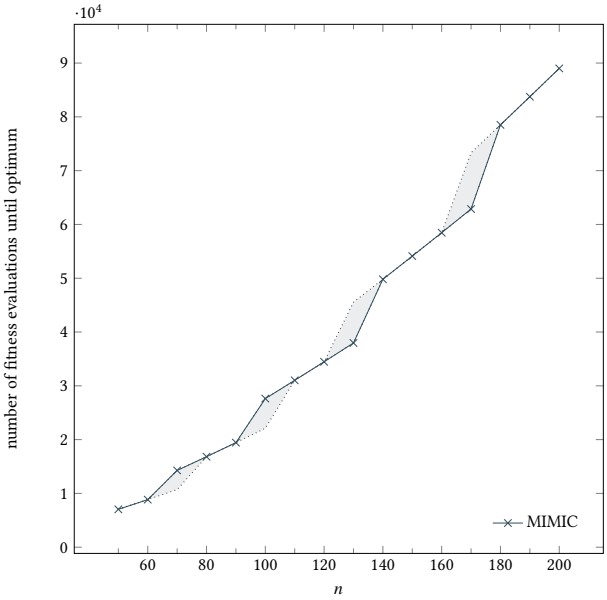
The number of iterations depicted is the number of iterations of each run until an optimum was found for the first time. That is, the number of iterations corresponds to T, as explained in Section 3.2. For each change in the number of iterations (for example, at $n = 70$), there is one value of n that has a high variance (the shaded area), and many runs take either the number of iterations of the previous value of n or an extra iteration. Except for these transitions, the run time of MIMIC is enormously consistent, with the mid 50 % all taking the same number of iterations. Overall, the number of iterations slightly increases with n.

The number of fitness function evaluations provides a better picture on how long MIMIC takes for a run. Note that the numbers shown in Figure 1b are the numbers from Figure 1a times λ, as MIMIC performs λ fitness evaluations in each iteration. The reason that the curve is not constant when the number of iterations stays the same for different values of n is that we chose $\lambda = \lfloor 12n \ln n \rfloor$, which grows in n. Thus, depending on how T grows in n, the total run time of MIMIC on EBOM is at least in the order of $n \ln n$.

3.3.2 Probabilistic model. Figure 2 and Tables 1 and 2 showcase information about the probabilistic model of MIMIC and its quality with respect to an ideal model (see also Section 2.4). For a comparison to make sense, it is important that the permutation π of a model of MIMIC is close to that of an ideal model – ideally, π would correspond to a permutation of an ideal model. To this end, we say that a permutation π is *correct* if, starting from the first position, its positions occur in pairs of two such that (1) the positions in each



(a) The number of iterations it took each of the 100 runs per n until an optimum was found for the first time (that is, T).



(b) The number of fitness evaluations it took each of the 100 runs per n until an optimum was found for the first time (that is, λT).

Figure 1: Two depictions of the run time of MIMIC optimizing EBOM. For information about the type of plot used, please refer to Section 3.2.1. For a discussion of these plots, please refer to Section 3.3.1.

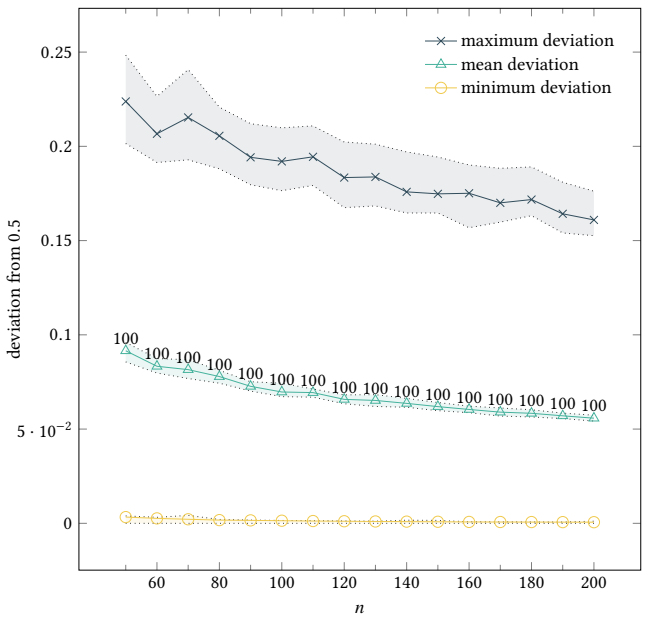


Figure 2: Depicted are the maximum, mean, and minimum of the deviation of the central probabilities in P from 0.5 in iteration $2T$. The numbers over the plot with the triangles denote the number of runs (out of 100) that have a correct permutation in their model. For a discussion of this plot, please refer to Section 3.3.2.

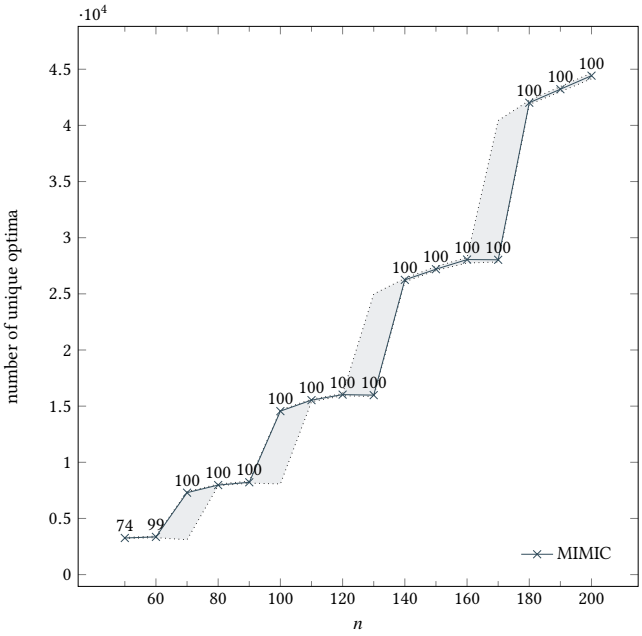


Figure 3: The number of distinct optima that MIMIC found when optimizing EBOM. An optimum is distinct if it was only sampled once during a single run. The number over each data point states how many of the 100 runs sampled *exclusively* distinct optima. For a discussion of these plots, please refer to Section 3.3.3.

Table 2: The probabilities of the positions 1 and 2 (occurring in π) of MIMIC optimizing EBOM for one of the runs with $n = 200$, over all iterations. For a discussion of this table, please refer to Section 3.3.2.

iteration	$P_{1,0}$	$P_{1,1}$	$P_{2,0}$	$P_{2,1}$
1	0.5	0.5	0.5	0.5
2	0.533825	0.454897	0.428928	0.587039
3	0.311688	0.665446	0.542926	0.485564
4	0.51094	0.458128	0.256098	0.789337
5	0.141582	0.828571	0.535533	0.478152
6	0.474968	0.453086	0.115023	0.903664
7	0.465116	0.519018	0.0794045	0.943806
8	0.465385	0.478368	0.0381406	0.98
9	0.450299	0.486737	0.00945626	0.995
10	0.440618	0.480589	0.005	0.995
11	0.005	0.995	0.442663	0.462725
12	0.005	0.995	0.470277	0.424279
13	0.406593	0.460972	0.005	0.995
14	0.455733	0.421111	0.005	0.995

pair differ by exactly 1 and that (2) the maximum of the positions of each pair is an even number. Note that the set of all correct permutations corresponds exactly to that of all ideal models.[2]

In Table 1, we show an excerpt of the model from one out of the 100 runs of MIMIC on EBOM in the last iteration, that is, $2T$. In total, we mention 10 entries from the model (out of 200). The first column depicts the bit positions as they occur in the permutation π. We see that all entries occur as they would in a correct permutation, suggesting that the entire permutation is correct. Note that the order of the positions per pair appears randomly, which makes sense, as the order does not matter for sampling a block in EBOM correctly.

The other two columns of Table 1 show the two probabilities of the position from the first column. We see that, for each pair of positions (as defined above), the first position has its probabilities close to 0.5 and second one has its probabilities at the borders of the interval $[\frac{1}{n}, 1 - \frac{1}{n}]$. Further, the probabilities at the borders are at the correct end for maximizing the probability of sampling a block in EBOM correctly. That is, the probability $P_{i,0}$ is at $\frac{1}{n}$ (making it likely to sample a 0 when the previous position sampled at 0), and the probability $P_{i,1}$ is at $1 - \frac{1}{n}$. Overall, the results from Table 1 already suggest that MIMIC builds a model close to an ideal one.

In Figure 2, we have a closer look at how closely the model of MIMIC in iteration $2T$ resembles an ideal model. In order for such a comparison to make sense, we first analyze how well the permutation of such a model deviates from the permutation of an ideal model. Out of *all* of our runs, *each* run produced a correct permutation in iteration $2T$. We depict these numbers in Figure 2 over the curve in the middle, with the triangles. Thus, the only way for a model of MIMIC to deviate from an ideal model is in how largely the probabilities in P deviate from those of an ideal model.

[2]Property (2) is necessary, since EBOM defines its block with respect to position 1. For example, positions 1 and 2 form a block in EBOM, but positions 2 and 3 do not.

When comparing probabilities of P to that of an ideal model, we group the probabilities into those that should be close to 0.5 (the *central* probabilities) and into those that should be close to the borders (the *border* probabilities). We may also use the respective adjective for a position in order to indicate that both of the probabilities are central or border. We group the probabilities with respect to the blocks in π. In order to determine which position of each block is central and which is border, we look at the probability with the highest deviation from 0.5 (breaking ties uniformly at random). The position with the probability that has the highest deviation is considered border, the other position is considered central.

We then calculate the absolute distance of each probability to its ideal value. For the central probabilities, we calculate their distance to 0.5 (regardless of whether the probability is conditional on a 0 or a 1). For the border probabilities conditional on a 0, we calculate their distance to $\frac{1}{n}$, and for those conditional on a 1, we calculate their distance to $1 - \frac{1}{n}$. Afterward, for the two groups of central and border probabilities, we calculate, for each of the positions per run and value of n, the maximum, mean, and minimum of the deviations of each probability.

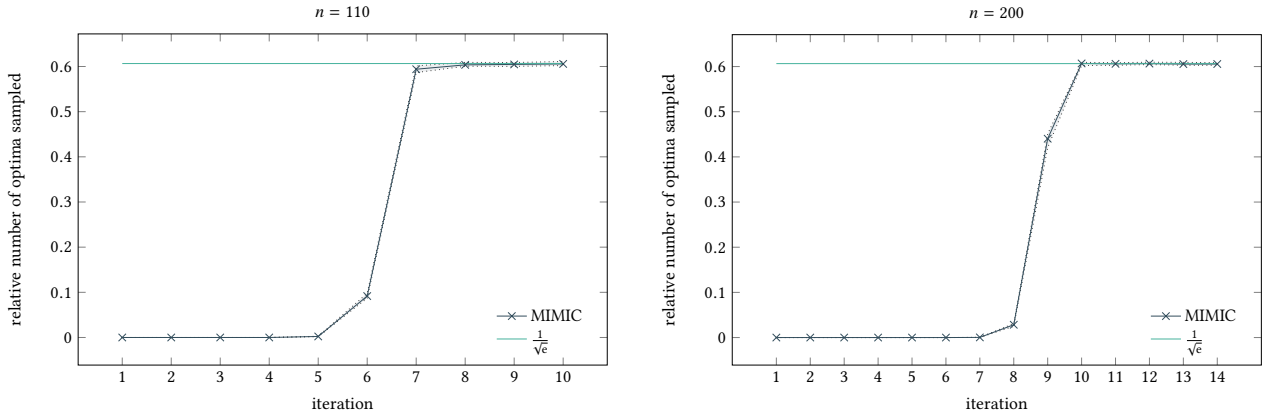
The results of these calculations for the central probabilities are depicted in Figure 2. The arguably most interesting result is the maximum deviation among all positions of a single run. This value seems to decrease with increasing n. However, a deviation of about 0.2 can still be considered rather large. We discuss in the following sections how this affects the quality of the model.

The deviations for the border probabilities are not depicted, as the maximum over all runs and all values of n was in the order of 10^{-6}. This suggests that the border probabilities are always very close to the borders in iteration $2T$.

Since we only looked at the model of MIMIC in iteration $2T$, Table 2 provides an excerpt of how the probabilities of the first block evolve over the iterations. We depict data from one of the runs with $n = 200$. From iteration 10 to 11 and from 12 to 13, we see that the probabilities of the positions 1 and 2 change their statuses of being central or border. This makes sense, as we already briefly discussed, as the order of the positions in a block does not matter for sampling a correct block. Given a correct block, it is then random which position MIMIC determines to be the first in its permutation (and, thus, central) and which it chooses next (being border). Thus, we conclude that MIMIC does not converge to a single model that is close to an ideal model but instead switches between different models from iteration to iteration.

3.3.3 Similarity to an ideal model. Since the results so far suggest that the model of MIMIC is close to an ideal model except for the deviation of the central probabilities (see Section 3.3.2), we now consider how well the model reflects the two properties of an ideal model that we describe in Section 2.4. We start with the probability to sample an optimum in each iteration.

Figure 4 shows how many of the solutions of the λ solutions sampled during each iteration are optima. We chose to depict this ratio for the cases of $n = 110$ and $n = 200$, which are cases where all of the mid 50 % of the runs used the same number of iterations (see also Figure 1a). This data can be interpreted as the probability of sampling an optimum in each iteration. Following our ideas discussed in Section 2.4, we also depict the value $1/\sqrt{e}$ in both plots,



(a) The relative number of optima in an iteration for $n = 110$. **(b) The relative number of optima in an iteration for $n = 200$.**

Figure 4: Depicted are how the relative number of optima sampled evolves over the number of iterations for MIMIC optimizing EBOM. The horizontal line at the top shows the value $1/\sqrt{e} \approx 60.65\,\%$, which is roughly the probability of sampling an optimum in a single iteration, given an ideal model of MIMIC for EBOM (see Section 2.4). For a discussion of these plots, please refer to Section 3.3.3.

which represents the probability to sample an optimum, given an ideal model.

Both Figures 4a and 4b show that the empirical ratio is surprisingly close to the ideal value. This suggests that the model behaves similarly to an ideal model in terms of consistently sampling optima, despite the central probabilities sometimes deviating somewhat largely from 0.5 (see Figure 2). The fact that some data points show a ratio that is slightly higher than the theoretical optimum is due to the variance in the randomness of the algorithm.

Figure 3 shows how many of the optima that MIMIC found per run were distinct as well as how many runs only found distinct optima. Except for the cases $n = 50$ and $n = 60$, MIMIC found exclusively distinct optima per run. This result is remarkable and suggests that MIMIC builds a very general model that is capable of sampling a huge variety of different solutions.

We now argue that it is not unlikely for the cases $n = 50$ and $n = 60$ to have runs that failed to only find distinct optima. In Section 2.4 we derived a lower bound on how likely it is to have *no* duplicate in m samples. In a similar fashion, one can derive an upper bound (using that, for $a \leq b$, $a!/(a-m)! \lesssim (a - \frac{m}{2})^m$ and that, for $x \in [0,1]$, $(1-x)^m \approx e^{-xm}$) of roughly $e^{-m^2/2^{n/2+1}}$. Thus, a lower bound of having a duplicate in m tries is roughly at least $1 - e^{-m^2/2^{n/2+1}}$. For $n = 60$, using that 4 out of 6 iterations are used for sampling optima and that about $1.7 \cdot 10^4$ solutions are created in a run (retrieved from the data used for Figure 1), we get that the probability for a run to have a duplicate optimum is about 6 %, which means that we would expect about 6 failures. For $n = 70$, the probability to have a duplicate optimum drops already below 1 %.[3]

Overall, the results from Figures 3 and 4 suggest that the model of MIMIC behaves similarly to an ideal model. We thus consider it to actually be similar to an ideal model.

[3]This estimation makes the assumption that the model is ideal in 4 out of 6 iterations. However, data similar to that depicted in Figure 4 suggests that it takes at least one iteration until the model samples optima consistently.

4 CONCLUSION

We showed for the test function EBOM that MIMIC efficiently generates a probabilistic model that behaves similarly to an ideal model. Since EBOM exhibits an exponential number of optima, this suggests that MIMIC is capable of implicitly storing a large range of different solutions in its model. Our experiments show that the model that MIMIC generates over time

- has a permutation and border probabilities (almost) as in an ideal model, that the model
- does not create duplicate optimal solutions with increasing input size, and that it
- samples optima in each iteration with a probability that is close to the theoretical optimum of $1/\sqrt{e}$.

Looking at sample data about the probabilistic model further suggests that the model is built such that it can generate an exponential number of optima. This is impressive, as MIMIC was not modified in any way and since this model is generated in a reasonable amount of time.

For future research, it is interesting to see if MIMIC also builds good models on more complicated functions with multiple optima, such as vertex cover on bipartite graphs. Further, since MIMIC has a very restricted type of bivariate model (namely, a path), considering other bivariate EDAs with a greater range of models, such as the bivariate marginal distribution algorithm ([22]; working on trees), would provide insights into whether the restriction of MIMIC's model to a path is a hindrance or not.

ACKNOWLEDGMENTS

This work was supported by COST action CA15140 and by a public grant as part of the Investissement d'avenir project reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

REFERENCES

[1] Ignasi Belda, Sergio Madurga, Teresa Tarragó, Xavier Llorà, and Ernest Giralt. 2007. Evolutionary computation and multimodal search: A good combination to tackle molecular diversity in the field of peptide design. *Molecular Diversity* 11 (2007), 7–21. https://doi.org/10.1007/s11030-006-9053-1

[2] Jeremy S. De Bonet, Charles Lee Isbell Jr., and Paul A. Viola. 1996. MIMIC: finding optima by estimating probability densities. In *Proc. of NIPS '96*. 424–430. http://papers.nips.cc/paper/1328-mimic-finding-optima-by-estimating-probability-densities

[3] Chung-Yao Chuang and Wen-Lian Hsu. 2010. Multivariate multi-model approach for globally multimodal problems. In *Proc. of GECCO '10*. 311–318. https://doi.org/10.1145/1830483.1830544

[4] Kenneth Alan De Jong. 1975. *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. Dissertation. USA. University of Michigan.

[5] Benjamin Doerr and Martin S. Krejca. 2020. The univariate marginal distribution algorithm copes well with deception and epistasis. In *Proc. of EvoCOP '20*. To appear.

[6] Benjamin Doerr and Weijie Zheng. 2019. Sharp bounds for genetic drift in EDAs. *CoRR* abs/1910.14389 (2019). https://arxiv.org/abs/1910.14389

[7] Tobias Friedrich, Timo Kötzing, and Martin S. Krejca. 2016. EDAs cannot be balanced and stable. In *Proc. of GECCO '16*. 1139–1146. https://doi.org/10.1145/2908812.2908895

[8] Georges R. Harik., Fernando G. Lobo, and David E. Goldberg. 1999. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computations* 3, 4 (1999), 287–297. https://doi.org/10.1109/4235.797971

[9] Georges R. Harik, Fernando G. Lobo, and Kumara Sastry. 2006. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer, 39–61. https://doi.org/10.1007/978-3-540-34954-9_3

[10] Mark Hauschild, Martin Pelikan, Claudio F. Lima, and Kumara Sastry. 2007. Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In *Proc. of GECCO '07*. 523–530. https://doi.org/10.1145/1276958.1277070

[11] Cem Hocaoğlu and Arthur C. Sanderson. 1997. Multimodal function optimization using minimal representation size clustering and its application to planning multipaths. *Evolutionary Computation* 5, 1 (1997), 81–104. https://doi.org/10.1162/evco.1997.5.1.81

[12] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. The MIT Press.

[13] Martin Krejca and Carsten Witt. 2020. Theory of estimation-of-distribution algorithms. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, Chapter 9, 405–442. https://doi.org/10.1007/978-3-030-29414-4, available online under https://arxiv.org/abs/1806.05392.

[14] Per Kristian Lehre and Phan Trung Hai Nguyen. 2019. On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In *Proc. of FOGA '19*. 154–168. https://doi.org/10.1145/3299904.3340316

[15] Samir W. Mahfoud. 1996. *Niching methods for genetic algorithms*. Ph.D. Dissertation. USA. University of Illinois at Urbana-Champaign.

[16] Heinz Mühlenbein and Gerhard Paaß. 1996. From recombination of genes to the estimation of distributions I. Binary parameters. In *Proc. of PPSN IV*. 178–187. https://doi.org/10.1007/3-540-61723-X_982

[17] Brad L. Miller and Michael J. Shaw. 1996. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proc. of CEC '96*. 786–791. https://doi.org/10.1109/ICEC.1996.542701

[18] José Peña, Jose Lozano, and Pedro Larranaga. 2005. Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. *Evolutionary Computation* 13 (2005), 43–66. https://doi.org/10.1162/1063656053583432

[19] Martin Pelikan and David E. Goldberg. 2001. Escaping hierarchical traps with competent genetic algorithms. In *Proc. of GECCO '01*. 511–518. https://dl.acm.org/doi/10.5555/2955239.2955318

[20] Martin Pelikan and David E. Goldberg. 2003. Hierarchical BOA solves Ising spin glasses and MAXSAT. In *Proc. of GECCO '03*. 1271–1282. https://doi.org/10.1007/3-540-45110-2_3

[21] Martin Pelikan, Mark Hauschild, and Fernando G. Lobo. 2015. Estimation of distribution algorithms. In *Springer Handbook of Computational Intelligence*. Springer, 899–928. https://doi.org/10.1007/978-3-662-43505-2_45

[22] Martin Pelikan and Heinz Mühlenbein. 1999. The bivariate marginal distribution algorithm. In *Advances in Soft Computing*. Springer, 521–535. https://doi.org/10.1007/978-1-4471-0819-1_39

[23] Gulshan Singh and Kalyanmoy Deb. 2006. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *Proc. of GECCO '06*. 1305–1312. https://doi.org/10.1145/1143997.1144200