Mixed-Integer Bayesian Optimization Utilizing A-Priori Knowledge on Parameter Dependences

Michael T. M. Emmerich a Rui Li a Anyi Zhang a Ildikó Flesch b
Peter Lucas c

a LIACS, Leiden University, Niels Bohrweg 1, 2333-CA Leiden, The Netherlands
b Tilburg Center for Creative Computing, Tilburg University, Warandelaan 2, 5037 AB, Tilburg, The Netherlands
c Institute for Computing and Information Sciences, Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract

Mixed-integer optimization problems arise in various application fields, such as chemical engineering and the optimization of medical image processing pipelines. Stochastic optimization algorithms, such as evolution strategies and estimation of distribution algorithms, can be used as solution methods for solving these problems approximately. Especially for real-world problems they often prove to be powerful methods due to their flexibility and robustness.

This paper introduces a new estimation of distribution algorithm that extends the Bayesian optimization algorithm (with fixed network structure) from binary optimization problems to mixed-integer optimization problems. We show that a-priori knowledge on dependences between decision variables can be exploited by this algorithm in order to improve convergence speed and reliability. In discussing the properties of heterogeneous Bayesian networks, representing multivariate distributions of mixed-variable type, we point out which kind of dependence information can be utilized. We assess the performance of the new approach using mixed-integer $N_k$-landscape models.

1 Introduction

Mixed-integer optimization deals with objective functions, the decision variables of which are of mixed type, for instance continuous, nominal discrete, and ordinal discrete. Mixed-integer optimization problems arise in various application fields, such as chemical engineering [6], optical filter design [15], or machine learning [11]. Stochastic optimization algorithms, such as Mixed-Integer Evolution Strategies (MIES) [6], proved to be powerful and flexible metaheuristics for solving real-world mixed-integer optimization problems.

A shortcoming of existing mixed-integer evolutionary algorithms, such as MIES, is that their variation procedures mutate each decision variable independently. Therefore, dependences between variables, even if they are known a-priori, cannot be taken into account. This contribution aims at designing and testing a mixed integer evolutionary algorithm that can utilize knowledge about such dependences.

One of the new types of evolutionary algorithms are the so-called Estimation of Distribution Algorithms (EDAs). EDAs do neither have a crossover nor a mutation operator. Instead, a new population is generated by sampling the probability distribution, which is estimated and updated based on the distribution of recently obtained ‘successful’ individuals. Different instantiations of EDA differ by the distribution types and update rules they use. For instance, the classical population-based incremental learning (PBIL) algorithm samples from an independent joint distribution of Bernoulli type [2], while the Univariate Marginal Distribution Algorithm (UMDA) [8, 17] features independent joint distributions of Gaussian type.

In this paper, we propose a Mixed-Integer Bayesian Optimization Algorithm (MIBOA), that is a variant of EDAs. MIBOA is an extension of the BOA approach to mixed-integer spaces using special types of Bayesian networks dealing with random variables of mixed-type. Moreover, a special type of mixed-integer $N_k$-landscape [1, 10] will be introduced that is well suited for testing the new approach.
The development of the new approach is motivated by problems in medical image analysis where the parameters of a medical image processing pipeline are to be optimized. Though the optimization of these systems is essentially a black-box optimization problem, dependence information can be extracted heuristically from the known structure of the processing pipeline. For details on the application domain the interested reader is referred to [5].

The paper is structured as follows: Section 2 introduces the problem definition. In Section 3 we discuss briefly mixed-integer evolution strategies (MIES) and estimation of distribution algorithms with independent sampling distributions. Section 4 introduces Bayesian optimization and generalizes it to the mixed-integer case. After introducing test problems based on Nk-landscapes in Section 5, we present results of mixed-integer BOA on these landscapes. Finally, the main results of the paper are summarized and directions of future research are discussed.

2 Problem Definition of Mixed-Integer Optimization

In this contribution we define the mixed-integer optimization as follows:

$$\text{minimize } f(r, z, d), r \in \mathbb{R}^l, z \in \mathbb{Z}^m, d \in D_1 \times \ldots \times D_n$$  \hspace{1cm} (1)

Here, $r$ denotes a vector of real numbers, $z$ from a finite set of integer values (or ordinal discrete values), whereas $d$ defines a $n$-tuple of nominal discrete variables with finite domains $D_i$, $i = 1, \ldots, n$. The function $f$ is considered to be a black-box function, or more precisely a function the mathematical structure of which is mainly unknown to the user. The only a-priori knowledge that we can exploit about $f$ are assumptions about parameter dependences (interaction of variables). A common feature of functions in which interactions occur is that they cannot be decomposed into a sum of functions depending only on single variables. For example, if $r_1$ interacts with $z_1$ and all other parameters are independent from each other, we can write the function as:

$$f(r, z, d) \equiv f_{1, l+1}(r_1, z_1) + f_2(r_2) + \ldots + f_l(r_l) + f_{l+2}(z_2) \ldots f_{l+m}(z_m) + f_{l+m+1}(d_1) + \ldots + f_{l+m+n}(d_n)$$

and $f_{1, l+1}(r_1, z_1)$ cannot be written as a sum of functions of $r_1$ and $z_1$. Non-separability makes it potentially difficult to optimize these functions by optimization routines that exploit such an assumption, such as coordinate search but also evolutionary algorithms that mutate variables independently from each other. In Section 5, with the ADG-based Nk-landscapes, an example for a function class in which various variable interactions can be introduced will be discussed.

3 Algorithms with independent sampling distributions

Next, let us introduce the evolution strategy (ES) and the estimation of distribution algorithm (EDA) as two basic evolutionary algorithms for parameter optimization\footnote{The ES is introduced, as it is a state-of-the-art technique in mixed integer optimization we will compare to later.}: The canonical $(\mu + \lambda)$ evolution strategy has the following iteration scheme:

**Step 1**: Create initial population $P := \{(a_1, \varsigma_1), \ldots, (a_\mu, \varsigma_\mu)\}$, where $\varsigma_i$ denotes a vector of dispersion parameters of the mutation distribution, e.g. standard deviations or mutation probabilities.

**Step 2**: Create offspring population $Q$ of size $\lambda$ by choosing randomly elements from $P$ and mutating first the distribution parameters $\varsigma_i$ to $\varsigma_i'$ and then the object variables $a_i$ using distribution parameters $\varsigma_i'$.

**Step 3**: Set $P$ to the $\mu$ best points (with respect to $f$) coupled with their mutated distribution parameters $\varsigma_i'$ out of $P \cup Q$.

**Step 4**: If termination criterion reached return best found solution, otherwise go to Step 2.

In contrast to this, estimation of distribution algorithms apply the following main loop:

**Step 1**: Initialize distribution parameters of distribution $D_0$.

**Step 2**: Create offspring population $Q$ of size $\lambda$ by sampling from the distribution $D_0$. 
Step 3: Set $P$ to the $\mu$ best points in $Q$ with respect to $f$.

Step 4: Update parameters $\theta$ of the distribution $D_\theta$ as a weighted average of the estimation of $\theta$ based on $P$ and the current parameter set $\theta$.

Step 5: If termination criterion reached then return best found solution, otherwise go to Step 2.

While in ES the basic variation operator is mutation, the variation operator in EDA is sampling from a multivariate distribution the parameters of which are dynamically updated based on positive examples.

Next, let us describe the mutation and sampling procedure for the mixed-integer case (without parameter dependences).

The mutation of mixed-integer evolution strategies can be described as a procedure:

Continuous mutation: Set $r_i = r_i + \text{Normal}(0, s_r)$, $i = 1, \ldots, l$.

Integer mutation: Set $z_i = z_i + \text{Geometric}(0, s_z) - \text{Geometric}(0, s_z)$, $i = 1, \ldots, l$.

Nominal discrete mutation: If $\text{Uniform}(0, 1) < p_d$ set $d_i$ to random value from $D_i - \{d_i\}$.

Here $\text{Normal}(0, s_r)$ computes a normally distributed random number with standard deviation parameter $s_r$, $\text{Geometric}(0, s_z)$ generates geometrically distributed random variables with mean $s_z$ [11], while $\text{Uniform}(0, 1)$ generates a uniformly distributed random number between 0 and 1. Before the mutation of the distribution parameter $s_r$ we employ the log-normal distribution as proposed by Schwefel [16] et al. $s_r \leftarrow s_r \exp(\tau_r \text{Normal}(0, 1))$ with $\tau_r = 1/\sqrt{l}$ being the learning rate. Accordingly, $s_z \leftarrow s_z \exp(\tau_z \text{Normal}(0, 1))$, with $\tau_z = 1/\sqrt{m}$ is used to adapt the step-size for integer mutations. The probability parameter $p_d$ is mutated based on a logistic mutation see (e.g. [15] et al.) that makes sure that the value of $p_d$ stays in $[0, 1]$. All three mutations of strategy parameters have the property, that increments are as likely as decrements of the value. The ES discussed here is termed mixed-integer evolution strategy and was discussed in several publications [6, 11].

For the sampling in the mixed-integer estimation of distribution algorithm similar distribution types are used. We employ the joint distribution $D_\theta$ composed of:

- a vector of $l$ independent multivariate normal distributions, with mean values $\mu_1, \ldots, \mu_l$ and standard deviations $\sigma_1, \ldots, \sigma_l$.
- a vector of $m$ random variables of type $\xi_i + Z_1(s_z) - Z_2(s_z)$, whereas $Z_1(s_z)$ and $Z_2(s_z)$ denote identically independent geometrically distributed random variables with mean value $s_z$.
- a vector of $n$ Bernoulli distributed binary random variables with probability parameters $p_1, \ldots, p_n$.

The described estimation of distribution algorithm is new for the mixed-integer search space. However, for binary nominal discrete parameters the algorithm is the classical population based incremental learning (PBIL) algorithm [2] and reduced to its continuous part it equals the so-called Univariate Marginal Distribution Algorithm (UMDA) [17, 8]. In the sequel, we will refer to the EDA algorithm for mixed-integer search space as MIPBIL.

The aforementioned two algorithms are used as reference algorithms to find out whether the introduction of dependence information improves the algorithms behavior or not. Next, we will look at an extension of MIPBIL that allows to integrate dependence information.

### 4 Mixed-Integer Bayesian Optimization Algorithm

In order to design a new mixed-integer estimation of distribution algorithm that can take into account dependences between variables of the objective functions we will replace the independent joint distribution $D_\theta$ used in the MIPBIL approach by an heterogeneous Bayesian network with fixed structure. This approach is also used in the bayesian optimization algorithm (BOA) by Pelikan et al. [14]. Their BOA method is applied for binary search spaces and also learns the structure of the network, while our approach is defined for mixed-integer search spaces and requires a-priori knowledge on the dependency structure of variables in the objective function. To emphasize the similarity to the BOA algorithm, we will term the new approach Mixed-Integer BOA (MIBOA).
Bayesian networks yield very powerful probabilistic graphical representations. The key to their popularity is their ease of representation of independence relations, and their support for reasoning with uncertainty.

A Bayesian network is a graphical representation of a probabilistic problem, formally defined as a pair \( B = (G, P) \), where \( P \) is the joint probability distribution on the set of random variables and \( G \) is an ADG representing the dependence and independence relations among this set of random variables, where each graphically represented marginal and conditional independence also has to be valid in the joint probability distribution [13]. Clearly, the definition of Bayesian networks implies as well that a dependence in the graph does not have to define a dependence in the joint probability distribution \( P \).

Let \( \{X_1, \ldots, X_d\} \) be a set of random variables. Then, based on the independence relations in the graph \( G \), the joint probability distribution \( P \) can be factorised as follows:

\[
P(X_1, \ldots, X_d) = \prod_{v=1}^{d} P(X_v \mid \pi(X_v)),
\]

where \( \pi(X_v) \) denotes the graphically represented set of parents of random variable \( X_v \). This implies that a joint probability distribution can be defined in terms of local distributions resulting in significant computational savings.

For reasoning in Bayesian networks there are several exact methods proposed that make use of local computations [4]. Here, local computations are based on the construction of join trees.

Hybrid Bayesian networks consist of both discrete and continuous random variables [3]. In these networks, local computations are possible, however, the correctness of the inference method depends on whether parents of a variable are discrete, continuous, a mixture of discrete and continuous, and on the choice of the local probability distribution.

The first method, introduced by Lauritzen [9], using exact inference is based on conditional Gaussian distributions. The restriction of this inference is that discrete random variables are not allowed to have continuous parents when hybrid Bayesian networks are concerned. To overcome this problem, Koller proposed a method which defines the distribution of these discrete nodes by a mixture of exponentials. However, for the inference Monte Carlo methods are used [7]. As another solution to this problem, we may discretise continuous variables, but discretisation introduces errors because we use approximation methods. However, in the experiment performed in this contribution we did not yet study the case of discrete nodes having continuous parents. For the Bayesian networks related experiments the BNT tool developed by Murphy was used [12]. The same basic algorithm than for PBIL was used, except that the distribution type and the update procedure was changed. A detailed description of the update algorithm would exceed the scope of the paper, and we refer to [12].

5 ADG-based \( Nk \)-landscapes

ADG-based \( Nk \)-landscapes (ADG-NKL), that we will introduce next, are attractive as models for optimization as their interaction structure corresponds to the dependence structure of Bayesian networks. Let \( x_1, \ldots, x_d \) denote a set of decision variables (the type of which can be continuous or discrete) and assume the interaction structure of the function is described by some ADGs, which is basically defined by a function \( \pi(\cdot) \), that assigns the set of parent nodes to each node, where the nodes represent parameters to be optimized. Then the ADG-based \( Nk \)-landscape can be written as a function of component functions \( f_i \):

\[
f(x_1, \ldots, x_d) = \sum_{i=1}^{d} f_i(x_i, \pi(x_i))
\]

Note, that this expression has the same structure as the expression \( \log P(X_1, \ldots, X_d) \) (see Equation (2)). Note also, that the \( x_1, \ldots, x_d \) denote variables of the objective function in contrast to \( X_1, \ldots, X_d \) denoting random variables.

The construction of the ADG-based \( Nk \)-landscapes corresponds to that of classical mixed-integer \( Nk \)-landscapes [10] with one exception. As for classical \( Nk \)-landscapes for each decision variable (or gene) \( x_i \) we choose \( k \) epistatic genes that interact with \( x_i \), in ADG-based \( Nk \)-landscapes we chose exactly the parent nodes as epistatic genes. Note, that the number of them can vary with the index of the decision variable in question. That is why the \( k \) in the expression '\( Nk \)-landscape' is not referring to the number of epistatic genes anymore - we kept it, however in the term, as it makes it easier to match the corresponding well known
\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1}
\caption{Example for an ADG-based Nk-landscape. The function values at the edge of the search space \([0,1]^d\) are set randomly between 0 and 1. Values inbetween are interpolated [10].}
\end{figure}

Nk-landscapes with the ADG-based Nk-landscapes. As with classical Nk-landscapes, the definition of the component functions in ADG-based Nk-landscapes is based on randomly generated function tables [10], as visualized in Figure 1. In the mixed-integer case multilinear functions are used to interpolate between the randomly chosen function values at the edges of a hypercube as described in [10].

## 6 Results

In order to check whether a-priori knowledge on the interaction structure integrated in the structure of the Bayesian network helps to speed up search we have conducted experiments on various ADG types that are visualized in Figure 2. These ADGs were used to construct Nk-landscapes indicating that the represented independence and dependence relations in an ADG are also included in the Nk-landscape constructed from this ADG. The same ADG is used as a structure for the Bayesian network as a-priori knowledge. For the probability tables, however, no a-priori knowledge is used. They are initialized based on the first population of selected individuals.

We applied three types of algorithms on ADG-based Nk-landscapes. 15 variables are considered, 5 for each type (l=m=n=5). As the population size turned out to be a crucial parameter, two different population sizes, 28 and 100, are tried. A number of 20 runs were statistically evaluated for each strategy.

Figures 3 to 5 show convergence dynamics for different sample landscapes defined by their ADG, each of which with a different structure. Averaged objective function values (difference to the global optimum) and standard deviations are plotted over the number of evaluations performed.

On the landscape ‘chain’ (Figure 3), the MIBOA performs best, when the population size is set to 100. For a population size of 28 the MIBOA performs almost equally to the MIES. In both cases the MIPBIL algorithm was clearly outperformed.

On the landscape ‘bitree’ (Figure 4), a binary tree, the MIBOA performs best, when the population size is set to 100. For a population size of 28 the MIBOA is faster but in the long run MIPBIL results in (almost) the same good value. MIES seems to have a problem in this landscape, which may be due to step-size reduction which can be harmful in multimodal landscapes. The large standard deviation supports this conjecture.

On the landscape ‘invtree’ (Figure 5), again the MIBOA has a big advantage in the beginning. Here this acceleration is more visible than for the previous landscape types. Again the MIES algorithm seems to have problems to converge to the global optimum, while the MIPBIL is more reliable, but suffers from a low convergence speed.

Comparing a population size of 100 with a population size of 28, it was observed that the MIBOA algorithm performs better with the larger population size. The standard deviation of results in that case is remarkably lower, indicating a good reliability of the good results. In Table 1 we summarize more results, including the ADG types ’tritree’, ’struct2’, and ’struct3’. The ranking after 2000, 5000, 10000, and 20000
Figure 2: Various types of ADGs used to define ADG-based Nk-landscapes and corresponding Bayesian networks. From left to right, ADGs are termed 'chain', 'struct2', 'struct3', 'bitree', 'tritree', and 'invtree'. Node types are defined as follows: discrete nodes(1-5), continuous nodes(6-10), integer nodes(11-15).

Figure 3: Convergence dynamics of MIES, MIPBIL, and MIBOA on a 'chain'-type ADG-NKL.

Figure 4: Convergence dynamics of MIES, MIPBIL, and MIBOA on a 'bitree'-type ADG-NKL.
Figure 5: Convergence dynamics of MIES, MIPBIL, and MIBOA on the 'invtree'-type ADG-NKL.

Table 1: Ranking position of average objective function values for $MIES_{\mu}$, $MIPBIL_{\mu}$, and $MIBOA_{\mu}$ with population size 28 and 100 on ADG-based NkL-landscapes after different numbers of evaluations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ranks $\sum$</td>
<td>Ranks $\sum$</td>
<td>Ranks $\sum$</td>
<td>Ranks $\sum$</td>
<td>Ranks $\sum$</td>
</tr>
<tr>
<td>$MIPBIL_{28}$</td>
<td>555565</td>
<td>31</td>
<td>555555</td>
<td>30</td>
<td>353545</td>
</tr>
<tr>
<td>$MIPBIL_{100}$</td>
<td>666656</td>
<td>35</td>
<td>666666</td>
<td>36</td>
<td>666666</td>
</tr>
<tr>
<td>$MIES_{28}$</td>
<td>222222</td>
<td>12</td>
<td>313333</td>
<td>16</td>
<td>433334</td>
</tr>
<tr>
<td>$MIES_{100}$</td>
<td>444444</td>
<td>24</td>
<td>444444</td>
<td>24</td>
<td>344441</td>
</tr>
<tr>
<td>$MIBOA_{28}$</td>
<td>111111</td>
<td>6</td>
<td>231111</td>
<td>9</td>
<td>221223</td>
</tr>
<tr>
<td>$MIBOA_{100}$</td>
<td>333333</td>
<td>18</td>
<td>122222</td>
<td>11</td>
<td>112112</td>
</tr>
</tbody>
</table>

iterations is reported. This table provides further evidence for the hypothesis that the introduction of the dependence information in the MIBOA is beneficial. In addition it can be observed that a small population size helps to speed up convergence of the algorithm in the short term, while a large population size improves its long term behaviour. For further details and results of this study we refer to [18].

7 Summary and Outlook

In this contribution we studied how knowledge on acyclic dependency structures can be integrated into stochastic optimization for mixed-variable search spaces. The Mixed-integer Bayesian Optimization Algorithm (MIBOA), an estimation of distribution algorithm working with heterogeneous Bayesian networks with a-priori set structure, was designed and studied. As a test environment mixed-integer NkL-landscapes have been modified to ADG-based mixed-integer NkL-landscapes. The dependence structure of their variables is defined as an ADG, and as a proof of concept it had to be studied, whether the MIBOA can exploit a-priori knowledge on this dependency structure or not. The test shows that the MIBOA algorithm can indeed take advantage of this a-priori information on dependences. In all cases of ADGs discussed ('chain', 'struct2', 'struct3', 'bitree', 'tritree', and 'invtree') we observed a performance gain as compared to mixed-integer evolution strategies and estimation of distribution algorithms, both working with an independent joint distribution, namely MIES and MIPBIL. The population size of MIBOA turned out to be an important parameter to control the trade-off between a fast convergence speed in the beginning and a reliable convergence to the global optimum towards the end of the search.

Future work will have to focus on studies on further synthetic and real-world problems, including cases where discrete parameters depend on continuous parameters, which turned out to be difficult to handle. In particular we are interested in applying the new algorithm in the context of optimization of image processing pipelines, the acyclic structure of which makes the MIBOA a particularly promising technique.

Acknowledgments This research is supported by the Netherlands Organization for Scientific Research (NWO).
References


