Mateda 3.0: EDAs and other model-based optimization algorithms with Matlab

Roberto Santana
Intelligent Systems Group
University of the Basque Country (UPV/EHU)
roberto.santana@ehu.eus

Abstract

keywords: genetic algorithms, estimation of distribution algorithms, model-based optimization, probabilistic graphical models, neural models, Matlab

1 To do

1.1 About running the program

- Determine if pmtk is enough to run the Bayesian network functions
- Check that all the examples of optimization problems run.
- Check possibility of using other Matlab toolboxes
- Check that all available visualization utilities work

1.2 Link with other libraries

- Library of CMAEs to convert results of comparing algorithms to latex tables and figures
- Library on multi-objective optimization with benchmark of functions
- Review functionalities available in Python packages

1.3 Addition of neural models

- Simple models based on RBMs
- Use of generators: GANs and VAEs
- Back-tracking and adversarial examples for generating solutions

1.4 Paper structure

- Simple introduction to model-based optimization and EDAs
- List of other papers with detailed information about models and algorithms tried
- Based on previous report, structure the new report: representations, operators, real-world problems

1.5 New contents to add

- Previous work Hossein on multi-objective modeling of variables and objectives
- Work Marcella on modeling Bayesian networks

- Work on permutation representation
- Analysis of genetic operators as models [work Josean]
- Models extracted from the Markov network book
- Inference methods based on belief propagation

1.6 New problems to add

- Ecology problems
- Battery design problems
- Neuro-evolutionary problems
- Traveling-thief problem
- Problems related to analysis of text

1.7 Link with other libraries

-
-

2 Introduction

A number of recent works in evolutionary algorithms have emphatically highlighted the need of exploiting the problem structure information when addressing optimization problems with evolutionary algorithms (EAs) [170, 172]. Expressions such as "problem structure matters" [170] or "blind (search) no more" [172] refer to the excruciating importance of identifying and exploiting problem information. Furthermore, the term *gray-boy optimization* has been coined to refer to a variety of algorithms, all sharing the characteristic of using, to different extents, the problem structure information. Research in this direction fosters the idea that exploiting information about the problem characteristics can produce important gains in efficiency. Gray-box optimization research then attempts new ways to solve the optimization problems for which some a-priori information is available, notably those problems with a "suitable" structure. The underlying assumption is that knowing this type of "structural" information can not only serve to improve traditional EA implementations, but also to create significantly novel and more efficient approaches.

Presenting ways of using knowledge about the problem characteristics in the design of genetic algorithms (GAs) [40, 57] was an early topic of attention in EAs [6, 28, 82]. DeJong [30] mentions two main approaches to adapt the classical GA definition to the characteristics of the problem: 1) To design an alternative representation of the same (solution) space for which the traditional (genetic) operators are appropriate. 2) To select different genetic operators that are more appropriate to the "natural representation". These two approaches have been present in several GA applications, typically in those that implement knowledge-based or heuristic genetic operators [17, 41, 127, 179].

For the analysis made in this paper we will assume that the "problem structure" exploited by gray-box optimizers corresponds to a singular representation of the a-priori known characteristics of the problem. Furthermore, we will assume that what makes gray-box optimization depart from other EA approaches, such as those mentioned above, is the conjunction of a particular type of problem knowledge (the structure of the interactions among variables) with very specific methods to exploit this knowledge. Therefore, while knowledge-based operators have been extensively investigated in evolutionary computation, we will examine the methods conceived for exploiting the problem structure, such as those advocated by gray-box optimization, as novel.

Problem structure, which is used by gray-box optimizers, has been previously exploited by other EAs. First and foremost, by model-building EAs, which create a representation of the relationships among the variables of the problem and use this representation to conduct a more efficient sampling of the search space. These algorithms have undergone rapid development in recent years [8, 45, 70, 151, 190, 191]. Significant research has been conducted in the field of estimation of distribution algorithms (EDAs) [70, 106] to better exploit information about the problem structure while solving the optimization problem.

With the exception of the simplest EDAs that assume independence between variables (e.g., PBIL [3] and UMDA [94]), the best known variant of EDAs are those that automatically learn probabilistic models of (black-box) optimization problems during the search. Nevertheless, there are EDAs whose factorizations are directly derived from the problem structure. They were originally called factorization-based distribution algorithms (FDAs) [105]. The analysis presented in this paper uses the similarities and differences between FDAs and gray-box optimization methods as a leitmotif to discuss several points involved in using problem information by EAs.

The goal of the paper is three-fold: First, to identify those concepts that are essentially identical in the two approaches, bridging the gap between analyses originated from different perspectives of the same problem. Secondly, to review research on the exploitation of the graphical representation of the problem structure in EAs. Finally, our aim is to discuss relevant questions related to the use of the problem structure in EAs, including the emergence of neural models, approaches that exploit the structure of multi-objective and constrained problems, and directions for convergence between gray-box optimization and EDAs.

The paper is organized as follows: The ideas brought up by works on gray-box optimization and gray-box optimizers are discussed in the next section. Some background on EDAs, and particularly on FDAs, is presented in Section 3. The relationship between gray-box optimization and EDAs is analyzed from different perspectives in Section 4. Section 5 illustrates the application of hyperplane-based and factorization-based formalisms to the analysis of a decomposable function. Section 6 discusses a number of relevant topics and challenging questions for EAs that exploit the problem structure. Section 7 concludes the paper.

3 Gray-box optimization problems and gray-box optimizers

To introduce the ideas of gray-box optimization, we briefly address the following questions:

1. A definition of problem structure.
2. An introduction to black-box, gray-box, and a finer color scale for problem structure characterization in optimization.
3. An explanation of what gray-box optimization is about.

3.1 Problem structure

We will assume in this paper that *problem structure* refers to the specific patterns of interactions among variables at the time of determining the objective function. In loose terms, a problem with no structure is a problem where all variables influence independently the values of the objective function. On the contrary, a problem with difficult or intricate structure is one in which several distinct large subsets of interacting variables, that partially overlap, participate in the computation of the function values.

The notion of problem structure we use here is related but essentially different to the one implicit in the analysis of fitness landscapes [38, 56, 64, 91], where the focus is not necessarily on the patterns of interactions among the variables but on other properties of the problem (e.g., number of local optima, basins of attraction, geometrical landscape features, etc.). Although we will mainly focus on single-optimization problems, the notion of problem structure can be naturally extended to multi-objective problems for which we can still evaluate how different patterns of interactions among the variables influence each of the objectives [1, 62, 77, 110, 180].

3.2 Gray-box optimization problems

Gray-box optimization problems are usually explained in contraposition to a black-box optimization problems. In the latter, we do not have any information about the structure of the function being optimized. In the first, case however, we know some information about the structure of the problem. For instance, in additively decomposed functions (ADFs) [105], the value of the function is the sum of the evaluation of a number of subfunctions defined on subsets of all the variables. If we consider that these subsets of (interacting) variables define the structure of the problem, and this structure is

Structure	Subfunction def.	Type of problems
White	White	All definition sets and their corresponding subfunctions are known.
White	Gray	All definition sets are known but some subfunctions are unknown.
White	Black	All definition sets are known but no information about subfunctions is available.
Gray	White	Definition sets are partially known, together with all their corresponding subfunctions.
Gray	Gray	Definition sets are partially known and only some of their corresponding subfunctions are available.
Gray	Black	Definition sets are partially known. No information about the subfunctions is available.
Black	Black	Nothing is known about the structure and consequently the subfunctions.

Table 1: The White-Gray-Black (WGB) classification of optimization problems according to the type and extent of problem information available. The information is classified using two criteria: 1) Definition sets of the function (Function structure). 2) Expression or procedure to define each subfunction (Subfunction definition). *White* refers to the case where the information is fully available. *Gray* to the situation in which the information is partially known, and *Black* when there is no available information.

known, then ADFs can be seen as a gray-box optimization problem. Several real-world problems could be included in the class of gray-box, e.g., MAX-kSAT, Ising model, NK-landscape, etc. [171].

While gray-box optimization is a relatively recent concept, the term *gray-box identification* [112] is given a similar meaning in modeling physical and networked systems and in control theory. It refers to situations in which a generic model structure is given and the parameters are those to be estimated from data.

Even if the difference between black-box and gray-box optimization problems seems sufficiently clear, there are situations in which information about the problem exists but it is only partial. For instance, we could know which are the groups of related variables where subfunctions are defined, but not the way in which they are related (i.e., the expression for the subfunctions defined in each group). It is also possible that structural relationships are only known for a limited number of groups, i.e., some definition sets of the function are unknown.

For combinatorial problems, we introduce in this paper a finer grain definition of the type of available problem information and the way it characterizes the optimization problem. We split the available information into: 1) Information about the structural relationships among variables (definition sets). 2) Information about the way in which the interactions are expressed within each group (definition of the subfunctions). Table 1 shows the White-Gray-Black (WGB) classification of optimization problems according to the type and extent of problem information available. Using this classification, black-box optimization problems would be classified as Black-Black problems, and gray-box optimization problems could be further divided into another 6 groups. To avoid confusion, and for ease of presentation, we will stick to the “gray-box” term in this paper.

3.3 Gray-box optimizers

Gray-box optimizers are optimization algorithms that exploit the information available about the structure of an optimization problem in order to make a more efficient search. It is assumed that, at least for some classes of gray-box problems, using this information will produce gains in efficiency. However, this is a very general definition since different classes of optimizers can use information about the problem structure in distinct ways and with different degrees of efficiency.

In the papers where the term gray-box optimization is discussed [20, 21, 163, 175], it is usually assumed that information about the structure is known *a priori*, i.e., it is not learned by the algorithm itself while conducting the search. Furthermore, while there are no apparent reasons to set constraints on the structural characteristics in the general class of gray-box optimization problems, for feasibility and efficiency reasons, gray-box optimizers assume that the structure of the problem is constrained. For instance, it is assumed that the size of any definition subset of an ADF to be optimized is upper-bounded by a parameter k (e.g., *k-bounded pseudo-Boolean* functions as presented in [171]).

A number of methods have been covered under the umbrella of gray-box optimizers. From highly efficient hill-climbers [21, 175], to enhanced partition crossover operators [20, 163], and combinations of black-box global optimizers and local-search gray-box optimizers [43].

4 Estimation of distribution algorithms

The main idea of Estimation of distribution algorithms (EDAs) [70, 79, 106] is to extract patterns shared by the best solutions, represent these patterns using a probabilistic graphical model (PGM) [67, 111], and generate new solutions from this model. In contrast to GAs, EDAs apply learning and sampling of distributions instead of classical crossover and mutation operators. Modeling the dependencies among the variables of the problem serves to efficiently orient the search to more promising areas of the search space by explicitly capturing and exploiting potential relationships among the problem variables. The pseudocode of an EDA is shown in Algorithm 1.

Algorithm 1: **Estimation of distribution algorithm**

```
Set $t \Leftarrow 0$. Create a population $D_0$ by generating $N$ random solutions.
do {
    Evaluate $D_t$ using the fitness function.
    From $D_t$, select a population $D_t^S$ of $K \leq N$ solutions according to a selection method.
    Compute a probabilistic model of $D_t^S$.
    Generate $D_{t+1}$ sampling from the model.
    $t \Leftarrow t + 1$
} until Termination criteria are met.
```

4.1 Factorized distribution algorithms

While it is usually assumed that EDAs learn the structure of the problem from data, this is not always the case. In fact, the first EDA based on the theory of PGMs was called Factorized Distribution Algorithm (FDA) [105] and it computed a factorization from a problem structure known apriori. In the field of EDAs, the term FDA was initially used to refer to EDAs that learned only the parameters of the probabilistic models and not the structure [98, 99, 100, 134, 145, 183]. However, the term was later also used to cover EDAs that learn the structure from data [81, 97, 104, 108, 109, 131, 146]. In this sense, the "FDA" and "EDA" terms were both used indistinctly to refer to the same class of algorithm. While the term "EDA" attempts to emphasize the role played in the algorithm by the *distribution estimation step*, the term "FDA" highlights that a *factorization of the distribution* is used in the estimation.

FDAs that use a fixed, a priori known structure of the problem to factorize the distribution were mainly applied for only a short time due to the rapid introduction of EDAs capable to learn higher order factorizations directly from the data, such as those based on Bayesian networks [36, 69, 115, 156] and Markov networks [2, 131, 150]. This may explain why FDAs based on a fixed structure are known to only a reduced number of early EDA adopters. However, and this is one of the main claims made in this paper, several of the questions originally addressed for the first FDAs (those based on a fixed structure), and the answers given to these questions, are related and are a deep concern to current research in gray-box optimization.

4.2 Alternative views of EDAs

In order to understand the different implications of using probabilistic modeling in EAs, and the significance of taking into consideration the problem structure in the construction of the models, it is convenient to approach EDAs from different perspectives. In this section we briefly discuss four of these perspectives.

4.2.1 An EA that learns and samples a probability model

The most common definition of an EDA is an EA which replaces crossover and mutation operators by the process of learning and sampling a model. This understanding of EDAs emphasizes their difference to GAs.

4.2.2 An automatic way to generate models of an optimization problem

When structural learning of the probabilistic model is applied, EDAs can be seen as methods that iteratively capture the "hidden" structure of the optimization problem. This means, in each generation, the learning method "mines" the selected population as traditional data-mining methods [49] do and unearths a model of the relations in the population.

In some applications, the model learned can be as advantageous as finding the optimal solution. Such a type of structure can serve to define similarity relationships among different instances of the same problem [51, 133], or to design transfer learning strategies [60, 116, 142]. In this sense EDAs can be seen as an automatic way to generate models of the problem. Furthermore, since these models are produced in a temporarily ordered manner, it is possible to associate the structural characteristics of the models to different stages of the search.

4.2.3 An effective method for problem decomposition

Problem decomposition is a general goal in Artificial Intelligence. The implicit parallelism hypothesis used to explain GAs assumes that multiple subproblems are simultaneously solved during the GA evolution, eventually leading to the optimal solution. A fundamental question for the design of effective GAs is to find a tight encoding of the building blocks so to avoid their disruption, the so-called linkage problem [48, 57]. FDAs with a fixed model solve this problem by appropriately encapsulating and respecting the interactions during the sampling process, effectively exchanging the building blocks of the parents in the new population.

EDAs that learn the structure actually identify the building blocks of the problem and organize them in the model in such a way that they will be usable for sampling. Both the process of identifying those building blocks (learning), and the process of combining them by sampling, are two essential ingredients of the automatic method for problem decomposition as implemented in EDAs. The problem decomposition perspective is applicable for methods, not necessarily EDAs, that can reuse the building blocks for implementing other alternative ways for improving the solutions [88, 143, 147]. This perspective of problem decomposition is also closely linked to gray-box optimization, showing a path for inclusion of gray-box optimization algorithms in EAs [43].

4.2.4 An algorithm to evolve probability distributions

An EDA can be seen as a process in which the different components of the algorithm assign a probability to every solution of the search space. To illustrate this fact, we introduce some basic notation.

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a vector of discrete variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $\{1, \ldots, n\}$, and X_S (respectively x_S) a subset of the variables of $\mathbf{X}$ (respectively $\mathbf{x}$) determined by the indices in S.

Let D_t and D_{t+1} denote the EDA populations at generations t and $t+1$, respectively. The selection method defines a probability of selection ($\rho_t^s(\mathbf{x})$) for each solution $\mathbf{x}$. Similarly, the factorization encoded by the probabilistic model defines a probability distribution that assigns a probability $\rho_t^a(\mathbf{x})$ of appearing in the next population to each solution. Figure 1 shows one possible representation of the way in which EDA components define the probability distributions of the solutions.

The GA recombination operators could be also seen as defining probability distributions on the solution space. What makes EDAs different is that the probabilistic models make the probabilities assigned to the solutions explicit. In GAs, approximating the probabilities determined by crossover and mutation can be cumbersome. In EDAs, given a probabilistic model, the computation of the probability is straightforward. Looking at EDAs from the perspective of algorithms that move in the space of probabilities is pertinent for the theoretical analysis of the algorithms [33, 34, 78, 101, 103] and can serve as a basis for implementing different types of inferences about the characteristics of the search space.

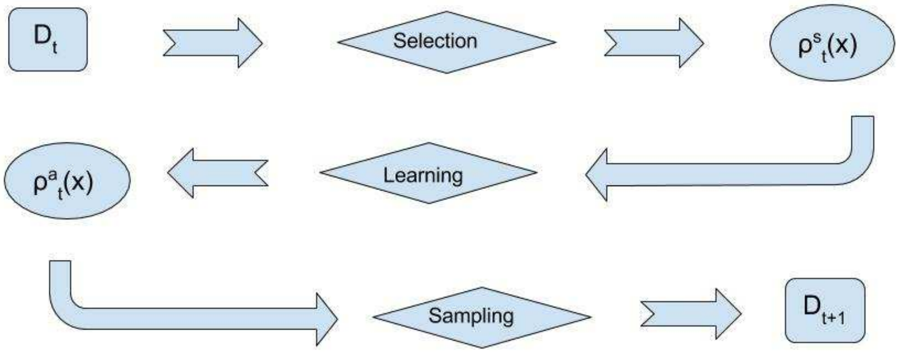

Figure 1: Joint probability distributions determined by the components of an EDA. D_t, D_{t+1}: populations at generation t and $t{+}1$; $\rho^s_t(\mathbf{x})$, $\rho^a_t(\mathbf{x})$: Joint probability distributions determined by selection and the probabilistic model approximation.

5 Gray-box optimization and EDAs

In this section we analyze different aspects of the relationship between gray-box optimization and EDAs. In particular, the following aspects will serve to illustrate these relationships:

1. Class of problems used to study the algorithms.
2. Representation of the problem structure.
3. Modeling the search for optimal solutions.
4. Using the problem structure for the search.
5. Implications of the structure for the behavior of the algorithms.

5.1 Class of problems

In additively decomposable functions (ADFs), possible interactions among the variables are reduced to a subset of these interactions. ADFs are used to simulate problems that can be decomposed into smaller subproblems.

Definition 1 *An ADF of order k is defined by*

$$f(x) = \sum_{s_i \in S} f_i(\Pi_{s_i}\mathbf{x}) \qquad S = \{s_1, \dots, s_l\} \quad s_i \subseteq X, |s_i| = k \tag{1}$$

where S is the set comprising the definition sets of the function, and $\Pi_{s_i}\mathbf{x}$ is the projection of $\mathbf{x} \in \mathbf{X}$ onto the subspace X_S.

Other approaches in EAs conceptualize the notion of ADF with different names. For instance, an embedded landscape with bounded epistasis k is defined in [53] as a function that can be decomposed as the sum of m subfunctions, each one depending at most on k input variables.

ADFs have been extensively used to investigate the influence of the problem structure and study the convergence properties of EDAs [15, 35, 100, 117, 183]. The behavior of these algorithms for other decomposable problems has been also studied from the perspective of ADFs [46, 148]. Linkage identification methods have been analyzed using ADFs [19, 24, 128, 165, 190].

When $x_i \in \{0, 1\}$, k-order ADFs are essentially identical to *k-bounded pseudo-Boolean*, functions which are used in several works addressing gray box optimization [20, 171, 174]. For instance, in [171], any problem that is expressed as a k-bounded pseudo-Boolean optimization problem, and for

gray-box	EDAs	reference EDAs	reference gray-box
Mk landscape problem	k-order ADF	[173]	[105]
structure-aware mutation	BB-wise mutation	[20, 173]	[76, 88, 147]
variable interaction graph	interaction graphs	[20, 173]	[96, 97, 135]
Tree decomposition Mk landscapes	ADF with bounded tree-width of k	[174]	[105]

Table 2: A cross-index of related concepts in gray-box optimization and EDAs.

which $M = O(n)$ is called an *Mk landscape*. In this paper, we stick to the terms of k-order ADFs and k-order decomposable problems since these terms are self-explanatory.

5.2 Representation of the problem structure

One of the key points in the analysis of the problem structure is to find meaningful representations of the relationships among the variables that allow the exploitation of the potential patterns hidden in the structure. In this section we analyze the graphical representations of the structure usually applied in gray-box optimization and EDAs, and the way these representations have been exploited.

5.2.1 Interaction graphs

In gray-box optimization, a variable interaction graph (VIG) [20, 173] is defined by associating one vertex to each variable of the problem. Every pair of variables that appear together in some subfunction are connected by an edge in the VIG. Figure 2a) shows an interaction graph for a problem with $n = 10$ variables and $M = 10$ subfunctions of order $k = 3$.

The VIG represented in Figure 2a) clearly displays, as triangles, the three-way interactions among subsets of variables. In addition to the local patterns, the graph seems to indicate a global symmetric pattern with respect to horizontal and vertical axes. Variables symmetries [23, 107] are only one example of the types of regularities of a problem structure that could be exploited by "structure-informed" optimization methods [120, 141].

Usually, EDAs use probabilistic graphical models (PGMs) to represent a factorization of the probability distribution. The PGM contains a graphical representation of the dependency relationships among the variables. Common PGMs are dependency trees, Bayesian networks, and Markov networks. Although PGMs are the norm when addressing black-box optimization problems for which the structure has to be learned, interaction graphs were early applied in EDAs [96, 97, 135] to investigate the relationship between the structure of the problem and the factorized approximation used to solve them. Table 4.2.1 shows a cross-index of related concepts in gray-box optimization and EDAs.

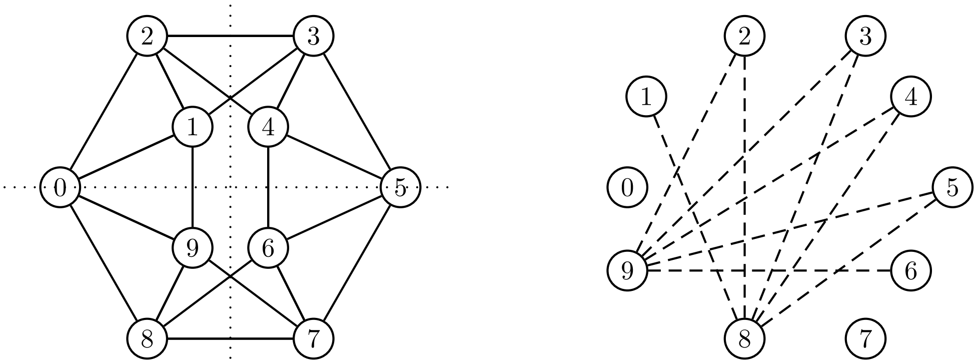

Figure 2: Left) Interaction graph for a problem with $n = 10$ variables and $M = 10$ subfunctions of order $k = 3$. Horizontal and vertical axes are represented with dotted lines to emphasize the symmetries that exist in the problem structure. Right) Edges added to the original interaction graph to obtain a chordal graph.

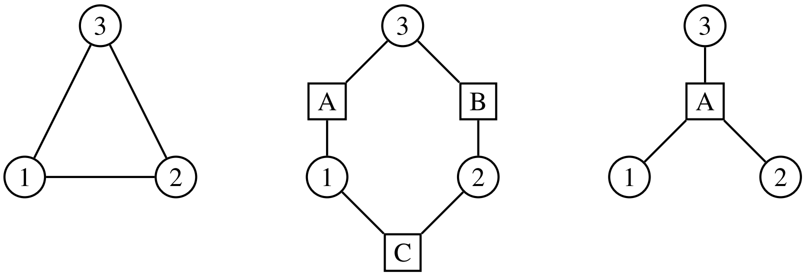

Figure 3: Left) Interaction graph of variables corresponding to a problem with variables X_1, X_2, and X_3. Center) Factor graph representing three pair-wise interactions among the variables. Right) Factor graph showing one three-variate interactions.

5.2.2 Factor graphs

One limitation of interaction graphs is that they are not expressive enough to show the order of the interactions among the variables. This is illustrated in Figure 3 Left) where we can not discern from the graph whether there are three pair-wise interactions, without a higher three-order contribution, or whether the three variables jointly interact.

A *factor graph* [68] is a bipartite graph that can serve to represent the factorized structure of a distribution. It has two types of nodes: variable nodes (represented as a circle), and factor nodes (represented as a square). In the graphs, factor nodes are represented by capital letters starting from A, and variable nodes by numbers starting from 1. Variable nodes are indexed with letters starting with i, and factor nodes with letters starting with a. The existence of an edge connecting variable node i to factor node a means that x_i is an argument of function f_a in the referred factorization.

Figure 3 Center) and Figure 3 Right) show the way in which factor graphs provide more complete information about the interactions among the variables of a problem. They represent two different patterns of interactions among the variables that would have an identical representation using an interaction graph.

Despite the more expressive nature of a factor graph to represent the problem structure, its adoption to model variable interactions in EAs has been limited [55, 75, 87, 95], probably due to the additional number of factor nodes it requires to model a factorization.

5.3 Modeling the search

The schema theory, the most extended formalism used to explain GAs, starts from the assumption that GAs change the sampling rates of hyperplanes in an n-dimensional hypercube corresponding to a binary encoding of the solution space [40, 57]. A schema is simply a hyperplane in the search space. It is represented by a chain of symbols taken from the variables domain plus a “don’t care” symbol (usually $\#$). Two features are used to describe a schema. Its order (number of positions in the schema that do not have the $\#$ symbol) and its defining length (distance between the outermost defined positions). Building blocks are defined as short, low-order schemata.

The schema theory tries to explain the behavior of algorithms that are defined by the application of crossover and mutation operators. It is not straightforward enough to apply it to explain FDAs, where the graphical representation of the interactions among the variables provides a more flexible encoding of the epistatic nature of the problem. For example, the role of the “defining length” of a schema is, to the very least, secondary for the analysis of FDAs, because these algorithms can guarantee that proper building blocks will not be disrupted during the sampling step, whatever the position of the variables in the solution representation. As long as the relevant dependencies of the problem are sufficiently covered by the probabilistic model, the behavior of FDAs will not be harmed by the position of the variables in the representation. This is an important difference with GAs, for which the defining length of the schemata can play a primary role to explain the behavior of the crossover operators.

The defining length is one of the conceptual components of the schema formalism that become irrelevant for the analysis of FDAs. There are also conceptual gaps, or missing components, in the schema theory that prevent their application to explain FDAs.

The limitations of the schema theory to explain EDAs have been analyzed in detail in [136] and an in-depth analysis of this question is beyond the scope of this paper. The significant point here is whether the key concepts used in schema theory (e.g., hyperplanes, schemata, building blocks, etc.) are appropriate to explain, and eventually further develop the work on gray-box optimization. On one hand, gray-box optimization builds on previous work on GAs, proposing ways to enhance traditional recombination operators (e.g., by introducing the partition crossover [163]). On the other hand, in order to take advantage of the variable interactions, gray-box optimization relies heavily on the use of a graphical representation, and increasingly on concepts and algorithms defined for them [174]. The dilemma lies in the fact that some of these graphical-model related concepts and methods have difficulty fitting into the theoretical schema analysis framework traditionally used to analyze GAs.

By selecting the theory of probabilistic graphical models as a framework to develop EDAs, it was possible to find a more precise way to explain the behavior of algorithms that exploit problem interactions than recurring to the schema theory. The theory of PGMs also makes it easier to develop these algorithms, since factors and factorizations are taken into account in the design of the algorithms. The position defended in this paper is that work on gray-box optimization illustrates a phenomenon in which competing distinct analytical characterizations of how EAs work collide in providing a coherent framework to investigate this type of algorithms.

While a framework that integrates the concepts from the schema theory and those commonly used to explain and design EDAs is not available, some understanding of the relationship between terms deployed in the two areas is needed in order to identify the links between the algorithms. A pair of terms that serve to illustrate this matter is (hyperplanes, factors).

5.3.1 Hyperplanes and factors

As mentioned before, the analysis of *hyperplanes* has been traditionally used to explain the behavior of GAs and the influence of recombination operators in the search for optimal solutions. Hyperplanes are usually assigned a mean fitness value computed by using all the solutions that belong to the hyperplane. To illustrate the concept, let us use the definitions introduced in [171] to present the concept of *order-j deceptive* function.

"Let h denote a $(n-j)$-dimensional hyperplane where j variables have preassigned bit values, and $\alpha(h)$ be a mask with 1 bits marking the locations where the j variables appear in the problem location and 0 elsewhere. Let $MAX(\mathbf{x}, \alpha(h))$ return the hyperplane with the best mean over all 2^j order j hyperplanes that can be defined using the $\alpha(h)$ mask. A function is *order-j deceptive* [171] if the j bit values returned by $MAX(\mathbf{x}, \alpha(h))$ for all hyperplanes of order j are not the same as the bit values found in a string which is a global optimum."

In the explanations of EDAs, *factors* play a role similar in importance to that played by hyperplanes in the most common theories used to explain GAs. But there is not a complete match between the terms due to the difference between the algorithms. Basically, a factor is a subset of variables from the problem that are modeled together. Usually, the variables are grouped in a factor to indicate that there is some sort of interaction among them in the problem. When a probabilistic model is learned, marginal probabilities are learned for each factor. A marginal probability assigns a probability value to each possible joint configuration of the variables in the factor. Ideally, in the marginal probability of the factor, the configurations of the variables with the highest probability will correspond to those with the highest fitness contribution to the global solution.

In terms of hyperplanes, we can see each of the 2^j configurations of a factor $(X_1, \dots, X_j)$ as a hyperplane, in which the marginal value of the configuration serves as an estimate of the corresponding fitness of the hyperplane. Since the table of marginal distributions contains the marginal values for all 2^j configurations, we can easily determine the factor configuration with the best fitness, similarly to the way in which the above-mentioned function $MAX(\mathbf{x}, \alpha(h))$ returns the hyperplane with the best mean over all 2^j order j hyperplanes. Whether the marginal probability is an accurate estimate of the fitness depends on the population size. Assuming an infinite population size, the estimate can be exact.

What makes factors exceptionally practical from the algorithm design point of view is that they can be conveniently combined to form factorizations. As a rule, factorizations are understood as factorizations of a distribution, but factorizations can also be interpreted in more general terms, as possible ways to represent the problem decomposition (see discussion in Section 3.2.3). In EDAs, a factorization can serve to represent the way in which the problem should be divided into sets of interacting variables, and equally important, *how these sets should relate to each other* in order to efficiently generate new solutions. Notice that the question of how to effectively encode (reorder) the sets of interacting variables in the hyperplane representation to increase the efficiency of gray-box optimizers is related to the issue of how to "factorize" the representation according to the problem interactions. One of the tricky issues in the analysis of hyperplanes is the scenario where different hyperplanes overlap. This is elegantly solved in FDAs by means of the graphical representation of the factors, and the methods to sample solutions working on this representation [105].

Factorizations can be grouped into classes according to the way factors are related, for instance in [105, 132, 136] the classes of valid and invalid factorizations are described. In general, decomposable problems have simple factorizations from which it is feasible to compute the best configuration in a short time. However, more difficult problems can also be approached from the point of view of factorizations.

While the way in which gray-box optimization and FDAs use the information about the structure of the problem is different, the implications that the characteristics of the graphical representation of the problem structure have for the complexity of the optimization problem are very similar for the two approaches. Decomposable problems are as trivial for FDAs [105], as order-k separable Mk landscapes are for a gray-box optimizer [174]. A more challenging question in the study of factorizations is the analysis of those problems that are not trivial, their identification, and the specific approaches conceived to deal with them by exploiting the problem structure. In Section 4.5 we discuss this question.

5.3.2 Other representations

In addition to hyperplanes and factors, other formalisms have been proposed to model the relationships among the variables in the design of EAs. One of these formalisms is the family of subsets (FOS) [160]. A FOS is a set of subsets of a certain main set S, i.e., a subset of the power-set of S. When used to analyze or implement EAs [11, 159, 160, 161, 166], the set S comprises all the problem variables indices. Thus, a FOS actually represents different subsets of the problem variables, and the choice of these subsets may be related to the problem structure or to the relationships imposed by the algorithm (e.g., a particular way to factorize the problem).

Although a subset of indices belonging to a FOS can be interpreted as a factor, and the idea of FOS is similar to the way in which region based approximations are defined in statistical physics to approximate the free energy [65, 178], the FOS representation is very flexible since a certain FOS can be seen independently of the mechanism used to generate it. Algorithms can be characterized by the properties of the FOS they generate. Kikuchi approximations [59], also used to represent sets of subsets of variables is EDAs, are defined by a specific algorithm that generates them. While FOS have been used to analyze EDAs and hybrid algorithms with characteristics of GAs and EDAs (e.g., linkage-tree GA [159]), as a formalism it is closer to factors and factorizations than to hyperplanes.

5.4 Using the structure for the search

In addition to providing an understanding of the patterns of relationships among the variables, graphical representations are handy for designing new operators and improving the components of EAs. Among the ways the structure can be used to improve the performance of EAs are:

- To bias the learning of the probabilistic models.
- To design more efficient local search and mutation operators.
- To design new, hybrid, search approaches.
- To implement transfer learning strategies.

5.4.1 Using the structure to bias the learning of the probabilistic model

Perhaps the most widely employed procedure to exploit the graphical structure of a problem in EDAs is to bias or constrain the way a probabilistic model is learned [4, 52, 132, 149]. This use of the structure is slightly different from the way in which FDAs use the graphical structure. In the latter case, the structure is employed to build a factorization of the problem that will be fixed along the evolution [134]. In the former case, however, a probabilistic model is learned in each generation. The graphical structure of the problem helps to narrow the space of probabilistic models that are searched [4, 104, 132], or to bias the learning process in such a way that those models that capture features of the known problem structure will be more likely to be learned [52, 118, 149].

5.4.2 Design of more efficient search operators

Designing more efficient search operators is the core idea of local-search gray-box optimization algorithms, such as the hill climber algorithms proposed in [20, 21], which use information about the structure of the function. In the hill climber approach it is assumed that the problem is k-bounded Pseudo-Boolean, and that the number of potential improving moves to be applied by the local optimizer is drastically reduced by taking into account the interactions among the variables. The influence of the interaction among the variables, as captured by the graph structure, is implicitly contemplated in the implementation of the method. As stated in [171], if it is assumed that if there are no more single-bit improving bit-flips, then the only pairs of variables that could be mutated together in order to achieve an improvement in the fitness of the function are those linked in the interaction graph.

The use of the problem structure is also at the very core of EDAs that use the information about the interactions among the variables to sample them together, instead of independently [71]. Furthermore, a number of proposals of structure-aware mutation operators, and other methods that exploit knowledge about the local structure of the interactions have been proposed in EDAs. Among these proposals are:

- Structure-aware mutation operators via probabilistic model building of neighborhoods [147].
- Using global statistical information to implement informed mutation operators such as guided mutation [185, 187].
- Substructural neighborhoods for local search in the Bayesian optimization algorithm [76, 80].
- Using belief propagation methods to exchange information about the best local configurations for each set of interacting variables [75, 87, 88].

Efficiency can also be improved by using the structural information as a way to initialize the solutions or populations previous to the application of the genetic operators. In [47], an algorithm is introduced that uses configurations of variables that correspond to hyperplanes with high quality average fitness to initialize stochastic local search for MAXSAT.

5.4.3 Hybrid methods that exploit problem structure

Besides gray-box optimization algorithms and EDAs, there are other methods that model the structure of the problem and use the models to generate new solutions. Among these methods are Optimal Mixing EAs (OMEAs) [160]. We consider these methods as “hybrid” because they learn graphical representations using machine learning algorithms similar to those traditionally used by EDAs but, instead of generating new solutions using probabilistic sampling methods, they apply operators reminiscent of those used by a traditional GA.

A distinguished feature of the different variants of OMEAs is the intermediate evaluation of solutions during the generation step. Solutions are created by making partial changes to an initial template (e.g., a solution from the population). Every time that a solution is partially modified, it will be evaluated to ascertain whether the modification made over the previous solution was beneficial. At the end of the process, the acceptance of the offspring in the new population will depend on whether any fitness improvement has been produced. The sets of variables that are modified together at each step of the solution generation are selected according to a "linkage" model. The values assigned to this set of values come from another "donor" solution that can be selected following different criteria. In the original Linkage Tree GA (LTGA) [159], mixing is restricted to two parents, in the Genepool Optimal Mixing EA (GOMEA) [160], the values selected for each factor of variables come from a random individual.

Another characteristic of the OMEAs is the way in which the structural information about the problem is represented. One of the most used models is the linkage tree, which is a structure commonly used to represent hierarchical clusters. The bottom level of the structure contains n single sets, and in every other level the two most similar sets in the previous level are clustered together. This multilevel representation adds a different perspective to the way in which structural information is represented. It allows to capture some hierarchical structural relationships among the problem components. However, this convenient property of linkage tree models is obtained at the expense of not being able to represent problems with overlapping structure. The Linkage Neighbor GOMEA [11] attempts to address this limitation by modeling, for each variable, the nearest neighbors in terms of linkage (i.e., similar to the concept of Markov blanket in Markov networks).

The large corpus of work on OMEAs [11, 12, 44, 129, 160, 169] provides a set of valuable lessons for the use of the problem structure in EAs, among them we identify the following:

- Parsimonious construction of solutions, with a monotonic improvement of the fitness and strong criterion for acceptance in the new generation, arise as a feasible alternative to probabilistic sampling, and as an effective way of keeping the population size small. It has been claimed that OM is a superior combination of local search and EAs [10].
- The uses given to the selected solutions at the time of model selection can be various. Contrary to the usual practice in EDAs, in LT-GOMEAs, selected solutions are used to learn the model structure [161], but they are not used in any way for biasing the assignments of the variables values.
- The efficiency of the OMEAs depends on the choice of the FOS or linkage sets, and on the relationship between these linkage sets and the problem structure [169]. LTGA does not seems to perform well for 2D spin glasses [118] and problems with complex overlapping linkage structure [44].
- In terms of the implications that knowledge about the problem structure may have for the search, perhaps the main message from the application of GOMEAs is as stated in [160]: "This supports the notion that it is not just finding a good configuration of a sufficiently complex structure, it is also the way in which this structural information is exploited upon creating new solutions that is of vital importance."

5.4.4 Transfer learning strategies based on structural similarity

A natural extension of the use of the graphical structure of a given problem is to exploit this information for problems that are different but share some structural characteristics. This is the case, for example, of different instances of the same problem (e.g., Max-SAT, NK-landscape, etc.). In these scenarios it would be desirable that EAs exploit information about the structure of one (source) problem at the time of optimizing other (target) problems. In [116] a transfer learning approach that computes a distance-based metric to bias the learning of the probabilistic model while solving a target problem instance is proposed. The metric is computed from the Bayesian network structures learned by hBOA while solving other source instances of the same problem. In [142], different variants of structural transfer are discussed. For the application examined it is shown that the use of partial information extracted from a set of source problems reduces the computational cost associated with probabilistic model learning and improves the quality of the final results for the target problem.

5.5 Problem structure, problem difficulty, and algorithm behavior

One initial consideration that we make is the difference between the general question of what makes a problem difficult for EAs, as investigated in [37, 83, 91, 162], and the more specific question of how the structure of a problem influences its difficulty for an EA. The first question is very broad since several factors can make a problem difficult for EAs.

Although more specific, the question of how the problem structure influences the difficulty that the problem poses to an EA is itself very complex and beyond the scope of this paper. Here we only sketch some points related to the problem structure and their impact on the results of FDAs and gray-box optimization algorithms. Our main claim here is that there are structural features that have a very similar impact on the two approaches. Indeed, most of these features can be derived from the analysis of the problem graphical structure.

5.5.1 Trivial and hard functions for gray-optimization and FDAs

k-order separable decomposable functions are easy both for gray-box optimization methods [173] and FDAs [105]. The complexity for both algorithms is exponential in k. A number of recent results in gray-box optimization are related to this fact [171, 174]. In [171], a pre-processing algorithm has been introduced that solves all separable Mk landscapes problems in $O(n)$ time. It has been also shown that “Localized Mk landscapes” can be constructed that are not adjacent, but which also can be solved in polynomial time.

Two findings shape the discussion on the relationship between the structural or graphical characteristics of a problem and its difficulty for FDAs. The first result, [105], points to the identification of the tree-width of a junction tree (the size of the largest node in the junction tree minus 1) constructed from the ADF structure as an indicator of problem complexity for FDA. This result was a breakthrough since it allowed to establish a connection between GA analysis, the theory of probabilistic graphical models, and well-known dynamic programming algorithms.

A second, very influential, result was presented by Gao and Culberson in [39]. They showed that for random ADFs the tree-width of the corresponding graphical structure is of order n. While k-order separable ADFs have tree-width $(k-1)$ and can be solved in $O(n)$, random ADFs of order k are exponentially complex for FDAs. Interestingly, EDAs that learn Bayesian networks can also see their performance quickly deteriorate for problems with an increasing number of k-order subfunctions [35]. Recently, the connection between the work Gao and Culberson [39] and the complexity of ADFs for gray-box optimization problems has been discussed in [174]. As expected, random functions are also exponentially complex for gray-box optimization algorithms.

While results for random functions point to the limits of optimization methods that use the problem structure, real-world optimization problems are not random in general. Therefore, the essential consideration at the time of addressing one real-world problem is what is the tree-width of its associated structure graphical representation. Nonetheless, even if some features that characterize the problem difficulty for FDAs and gray-box optimization methods can be extracted from the analysis of its graphical structure, the extraction of these features is not always straightforward. For example, the problem of finding the triangulation of a graph with minimum tree-width problems is NP-hard [177]. A positive note is that there are fast algorithms which are capable of computing approximations [54, 158].

Another question that is usually overlooked in the analysis of the problem structure is the type and strength of the interactions among the variables. In [135], capturing benign and malign interactions [61] in the model is shown to have a different effect on the behavior of EDAs. Similarly, weak and strong interactions are not equally significant at the time of modeling a problem. Therefore, while an analysis based on the tree-width of the graphical representation and on other structural characteristics provides a global picture of the complexity of the problem, a detailed picture will depend on the type and strength of the interactions represented by the structure.

6 Competing analytical characterizations at work

In this section we present an example that illustrates how an analytical approach based on the schema theory can be misleading at the time of using and understanding how the problem structure influences the behavior of the algorithms.

6.1 A link between hyperplane statistics and marginal distributions

6.1.1 Boltzmann distribution

The Boltzmann probability distribution $p_B(\mathbf{x})$ (also called Gibbs distribution) is defined as:

$$p_B(\mathbf{x}) = \frac{e^{\frac{g(\mathbf{x})}{T}}}{\sum_{\mathbf{x}'} e^{\frac{g(\mathbf{x}')}{T}}}, \tag{2}$$

where $g(\mathbf{x})$ is a given objective function and T is the system temperature that can be used as a parameter to smooth the probabilities. The inverse temperature $\beta = \frac{1}{T}$ is frequently used instead of the temperature.

A convenient property of $p_B(\mathbf{x})$ is that it assigns a higher probability to solutions with better fitness. The solutions with the highest probability correspond to the optima. The relationship between the fitness function and the variables dependencies that arise in the selected solutions can be modeled using the Boltzmann probability distribution [58, 81, 103, 105].

6.2 The case of problems with circular structure

To illustrate how an analysis based on hyperplanes can be misleading, we take an adjacent MK landscape presented in [171, 174]. For this function, we replicate the computation of a number of hyperplanes, as done in [171, 174], and also compute other factorizations of the problem derived from the analysis of its graphical representation. We then compare the findings from these two approaches.

The additive function has $n = 10$ variables and is computed as the sum of 10 subfunctions of order $k = 3$. Table 3 presents the definition as taken from [171]. Each subfunction has four local optima with evaluation 1 and all other points have evaluation 0. For each subfunction, one of the four local optima is reached at 111. The global optimum of the function is reached at a vector with all variables set to 1.

Index	subfunction	local optima				codomain vector
1	$f(x_1, x_2, x_3)$	000	001	100	111	$< 1, 1, 0, 0, 1, 0, 0, 1 >$
2	$f(x_2, x_3, x_4)$	011	001	100	111	$< 0, 1, 0, 1, 1, 0, 0, 1 >$
3	$f(x_3, x_4, x_5)$	001	010	101	111	$< 0, 1, 1, 0, 0, 1, 0, 1 >$
4	$f(x_4, x_5, x_6)$	110	100	011	111	$< 0, 0, 0, 1, 1, 0, 1, 1 >$
5	$f(x_5, x_6, x_7)$	001	100	110	111	$< 0, 1, 0, 0, 1, 0, 1, 1 >$
6	$f(x_6, x_7, x_8)$	000	100	010	111	$< 1, 0, 1, 0, 1, 0, 0, 1 >$
7	$f(x_7, x_8, x_9)$	001	010	101	111	$< 0, 1, 1, 0, 0, 1, 0, 1 >$
8	$f(x_8, x_9, x_0)$	000	001	100	111	$< 1, 1, 0, 0, 1, 0, 0, 1 >$
9	$f(x_9, x_0, x_1)$	000	010	011	111	$< 1, 0, 1, 1, 0, 0, 0, 1 >$
10	$f(x_0, x_1, x_2)$	000	101	110	111	$< 1, 0, 0, 0, 0, 1, 1, 1 >$

Table 3: Example of adjacent MK landscape presented in [171]. Each subfunction has four local optima with evaluation 1; all other points have evaluation 0. These values are represented by the codomain vector. The global optimum is the string of all 1 bits, and each subfunction reaches the optimum at 111.

We then generate the complete space of $2^{10} = 1024$ solutions and evaluate them. Using the solutions and their corresponding evaluations, it is possible to compute the average fitness of any partial configuration of any subset of variables (factor). We start by computing the marginal frequencies for factors corresponding to all definition sets of the function, i.e., those triplets of variables shown in column 2 of Table 4.

Since all factors in a given column have the same size, instead of calculating the average fitness, we simply add the fitness values of all solutions where a specific configuration of the factor is present. For example, for column 2, where $k = 3$ and the number of solutions of the search space that share a given configuration (x_i, x_{i+1}, x_{i+2}) is $2^{n-k} = 2^7 = 128$, we do not divide the sum of the frequencies for (x_i, x_{i+1}, x_{i+2}) by 128. The rank of the frequencies for each factor will coincide with the rank of the statistics computed for the corresponding hyperplanes of order k. For $k = 3$,

Index	$j = 3$	$j = 4$	$j = 5$
1	(x_1, x_2, x_3)	(x_1, x_2, x_3, x_4)	$(x_1, x_2, x_3, x_4, x_5)$
2	(x_2, x_3, x_4)	(x_2, x_3, x_4, x_5)	$(x_2, x_3, x_4, x_5, x_6)$
3	(x_3, x_4, x_5)	(x_3, x_4, x_5, x_6)	$(x_3, x_4, x_5, x_6, x_7)$
4	(x_4, x_5, x_6)	(x_4, x_5, x_6, x_7)	$(x_4, x_5, x_6, x_7, x_8)$
5	(x_5, x_6, x_7)	(x_5, x_6, x_7, x_8)	$(x_5, x_6, x_7, x_8, x_9)$
6	(x_6, x_7, x_8)	(x_6, x_7, x_8, x_9)	$(x_6, x_7, x_8, x_9, x_0)$
7	(x_7, x_8, x_9)	(x_7, x_8, x_9, x_0)	$(x_7, x_8, x_9, x_0, x_1)$
8	(x_8, x_9, x_0)	(x_8, x_9, x_0, x_1)	$(x_8, x_9, x_0, x_1, x_2)$
9	(x_9, x_0, x_1)	(x_9, x_0, x_1, x_2)	$(x_9, x_0, x_1, x_2, x_3)$
10	(x_0, x_1, x_2)	(x_0, x_1, x_2, x_3)	$(x_0, x_1, x_2, x_3, x_4)$

Table 4: Different sets of factors used to compute the marginal frequencies of the function defined in Table 3.

the results are shown in Table 5, where, for each factor, the configurations with the highest marginal frequency are underlined.

	Frequencies for the subfunctions									
	1	2	3	4	5	6	7	8	9	10
000	$\underline{768}$	$\underline{704}$	608	480	480	544	$\underline{704}$	$\underline{736}$	640	736
001	512	$\underline{704}$	$\underline{800}$	736	416	672	576	672	$\underline{768}$	544
010	512	576	544	672	608	480	$\underline{704}$	672	576	672
011	512	576	736	672	$\underline{800}$	480	576	480	704	736
100	640	$\underline{704}$	608	416	736	736	$\underline{704}$	672	640	544
101	640	576	544	672	544	608	576	608	640	480
110	$\underline{768}$	576	544	608	736	$\underline{800}$	576	608	448	608
111	$\underline{768}$	$\underline{704}$	736	$\underline{864}$	$\underline{800}$	$\underline{800}$	$\underline{704}$	672	704	$\underline{800}$

Table 5: Marginal frequencies for all the configurations of factors of order 3. These factors correspond to the definition sets of the function presented in Table 3. Marginal frequencies are computed as the sum of the fitness for all solutions that contain the corresponding configuration of the factor. In the table, the configurations where the marginal configuration reached the highest value are underlined.

It can be seen in Table 5 that among the factor configurations with highest value, there is always one that agrees with the global optima of the function. This happens for all factors except in three of them: 3, 8, and 9. In some way these hyperplanes are deceptive. In the vein of the analysis presented in [171], we expand the hyperplanes to include more adjacent variables as shown in columns 3 and 4 of Table 4. The results of the computation of the marginal frequencies are displayed in Table 6 and Table 7, respectively.

	Frequencies for the subfunctions									
	1	2	3	4	5	6	7	8	9	10
0000	384	304	240	256	208	304	$\underline{368}$	336	368	$\underline{432}$
0001	384	$\underline{400}$	368	224	272	240	336	$\underline{400}$	272	304
0010	256	304	400	320	208	368	336	304	368	272
0011	256	$\underline{400}$	400	416	208	304	240	368	$\underline{400}$	272
0100	288	304	208	384	336	272	$\underline{368}$	336	304	336
0101	224	272	336	288	272	208	336	336	272	336
0110	224	240	304	320	$\underline{400}$	208	272	176	304	368
0111	288	336	$\underline{432}$	352	$\underline{400}$	272	304	304	$\underline{400}$	368
1000	320	304	240	224	336	400	$\underline{368}$	304	368	336
1001	320	$\underline{400}$	368	192	$\underline{400}$	336	336	368	272	208
1010	320	240	272	288	272	336	336	272	304	240
1011	320	336	272	384	272	272	240	336	336	240
1100	$\underline{416}$	304	208	352	$\underline{400}$	$\underline{432}$	304	304	240	304
1101	352	272	336	256	336	368	272	304	208	304
1110	352	304	304	416	$\underline{400}$	368	336	272	304	400
1111	$\underline{416}$	$\underline{400}$	$\underline{432}$	$\underline{448}$	$\underline{400}$	$\underline{432}$	$\underline{368}$	$\underline{400}$	$\underline{400}$	400

Table 6: Marginal frequencies for all the configurations of factors of order 4 corresponding to the definition sets of the function presented in Table 3.

For factors of order 4 (Table 6), among the factor configurations with the highest value, there is always one that agrees with the global optima of the function for all factors except one: factor

10. The same happens for factors of order 5 (Table 7), but in this case the "deceptive" factor is 9. We point out that the configurations with the highest marginal values for $j = 5$ do not agree with those published in [171, 174] where only for 4 of the 10 hyperplanes the configuration 11111 was among those with the highest value of the hyperplane statistics[1]. This lack of consistency between the configuration of the global optima and the configuration of the hyperplanes led the authors to consider the function as deceptive at order 5. The exact quote in [174] is: "Despite the fact that this is a simple Adjacent NK Landscape (with localized variable interactions and localized nonlinearity) the order 5 hyperplane statistics do not provide any recognizable mechanism for identifying the global optimum. This simple function is still deceptive at order 5."

	Frequencies for the subfunctions									
	1	2	3	4	5	6	7	8	9	10
00000	168	120	128	112	120	160	168	192	$\underline{216}$	$\underline{216}$
00001	216	184	112	144	88	144	200	144	152	$\underline{216}$
00010	168	200	160	112	152	144	152	192	136	152
00011	216	200	208	112	120	96	184	208	136	152
00100	136	120	$\underline{224}$	176	120	192	168	160	184	152
00101	120	184	176	144	88	176	168	144	184	120
00110	104	168	192	208	88	144	88	160	200	120
00111	152	$\underline{232}$	208	208	120	160	152	208	200	152
01000	120	120	112	176	184	144	168	192	184	168
01001	168	184	96	208	152	128	200	144	120	168
01010	88	136	144	144	152	128	152	160	136	168
01011	136	136	192	144	120	80	184	176	136	168
01100	120	88	176	176	$\underline{216}$	112	136	96	152	200
01101	104	152	128	144	184	96	136	80	152	168
01110	120	136	208	176	184	128	120	128	200	168
01111	168	200	$\underline{224}$	176	$\underline{216}$	144	184	176	200	200
10000	136	120	128	96	184	208	168	176	$\underline{216}$	168
10001	184	184	112	128	152	192	200	128	152	168
10010	136	200	160	96	$\underline{216}$	192	152	176	136	104
10011	184	200	208	96	184	144	184	192	136	104
10100	168	88	160	160	152	176	168	144	152	136
10101	152	152	112	128	120	160	168	128	152	104
10110	136	136	128	192	120	128	88	144	168	104
10111	184	200	144	192	152	144	152	192	168	136
11000	184	120	112	160	$\underline{216}$	$\underline{224}$	136	176	152	152
11001	$\underline{232}$	184	96	192	184	208	168	128	88	152
11010	152	136	144	128	184	208	120	144	104	152
11011	200	136	192	128	152	160	152	160	104	152
11100	184	120	176	$\underline{224}$	216	192	168	144	152	$\underline{216}$
11101	168	184	128	192	184	176	168	128	152	184
11110	184	168	208	$\underline{224}$	184	208	152	176	200	184
11111	$\underline{232}$	$\underline{232}$	$\underline{224}$	$\underline{224}$	$\underline{216}$	$\underline{224}$	$\underline{216}$	$\underline{224}$	200	$\underline{216}$

Table 7: Marginal frequencies for all the configurations of factors of order 5 corresponding to the definition sets of the function presented in Table 3.

As previously mentioned, our computation of the marginal frequencies of the factors does not agree with the results published in [171, 174]. However, in both cases, in the results presented in [171, 174] as well as in Table 7, not all factors have a maximal configuration that agrees with the global optima. Therefore, from the hyperplane analysis we could label the function "deceptive".

This example shows that expanding the hyperplanes by considering contiguous variables to detect the degree of deception in functions with overlapping definition sets can be misleading. The mishap is at the time of identifying which are the relevant hyperplanes or factors. The fact that not every factor among the 10 of order 5 shown in Table 4 are consistent with the global optima does not mean that there could be other factors, of the same order, the combination of which could lead an EA to the optimum. Actually, as we will show in what follows, six factors of order five are sufficient to get an exact factorization for this problem. Therefore, for these factors, the problem is not deceptive.

The problem of identifying the "relevant" hyperplanes can be re-framed as the problem of finding a minimal valid factorization that captures every single interaction of the original function, and this can be achieved by the triangulization of the interaction graph and the computation of a junction tree from the triangulized graph. Computing this type of factorizations from the problem structure and

[1] As supplementary information, we have included the code we have used to compute the marginal frequencies for all factors, as well as the output of the algorithms. This code is available from `https://github.com/rsantana-isg/graybox_fda_paper`

using them to solve deceptive functions were two of the most important results introduced in [105] and extensively exploited in EDAs since then.

Figure 2 Right) shows a set of edges that, when added to the original interaction graph Figure 2 Left), produce a chordal or triangulated graph. Table 8 shows the six factors obtained from the triangulated graph as well as the marginal frequencies for all the factor configurations. It can be seen that, for all factors the configuration 11111, consistent with the global optima, is among those with the highest fitness value. Equation (3) shows the factorization that can be obtained from these six factors.

	Factor frequencies					
	$(0,1,2,8,9)$	$(1,2,3,8,9)$	$(2,3,4,8,9)$	$(3,4,5,8,9)$	$(4,5,6,8,9)$	$(5,6,7,8,9)$
00000	$\underline{216}$	160	120	120	144	168
00001	168	160	136	136	160	200
00010	168	128	88	88	96	152
00011	152	160	136	136	144	184
00100	$\underline{216}$	$\underline{208}$	184	104	160	168
00101	168	$\underline{208}$	200	120	176	168
00110	168	176	152	72	144	88
00111	152	$\underline{208}$	200	120	192	152
01000	152	144	168	152	128	168
01001	168	144	184	168	144	200
01010	104	112	136	120	80	152
01011	152	144	184	168	128	184
01100	152	192	168	200	112	136
01101	168	192	184	$\underline{216}$	128	136
01110	104	160	136	168	96	120
01111	152	192	184	$\underline{216}$	144	184
10000	152	144	104	184	192	168
10001	200	176	120	200	208	200
10010	136	112	72	152	144	152
10011	$\underline{216}$	176	120	200	192	184
10100	120	128	168	136	144	168
10101	168	160	184	152	160	168
10110	104	96	136	104	128	88
10111	184	160	184	152	176	152
11000	120	128	152	184	208	136
11001	168	160	168	200	$\underline{224}$	168
11010	104	96	120	152	160	120
11011	184	160	168	200	208	152
11100	152	176	216	200	192	168
11101	200	$\underline{208}$	$\underline{232}$	$\underline{216}$	208	168
11110	136	144	184	168	176	152
11111	$\underline{216}$	$\underline{208}$	$\underline{232}$	$\underline{216}$	$\underline{224}$	$\underline{216}$

Table 8: Marginal frequencies for factorization represented by Equation (3).

$$
\begin{aligned}
p(\mathbf{x}) = {} & p(x_0x_1x_2x_8x_9) \cdot p(x_3|x_1x_2x_8x_9) \cdot p(x_4|x_2x_3x_8x_9) \\
& \cdot p(x_5|x_3x_4x_8x_9) \cdot p(x_6|x_4x_5x_8x_9) \cdot p(x_7|x_5x_6x_8x_9)
\end{aligned}
\tag{3}
$$

In [174], it is also proved that functions with circular structure of order K, where K is the number of neighbors (cyclic adjacent NK landscapes), can be converted to acyclic adjacent NK landscape of order $2K$. The function shown in Table 3 is a cyclic adjacent NK landscape and for a function with the same structure it is also shown in [174] how to construct a possible acyclic adjacent NK landscape with $K = 4$, i.e., factors of order five.

7 Improving and extending the scope of methods that exploit the problem structure

To this point we have provided evidence that FDAs and gray-box optimization algorithm benefit from the graphical structure of the problems and that they are similarly constrained in their performance by the properties of these structures. Therefore, in this section we discuss a number of issues that are challenging for these two classes of algorithms, and for other EAs that exploit the problem structure. We also discuss issues that represent an opportunity to enhance and extend these algorithms. The following topics are covered:

- Problems with large definition sets and a high cardinality of the variables.
- Estimating or learning the structure of black-box problems.
- Redefining problem structure for constrained and multi-objective problems.
- The question of evolvability.
- Combining gray-box optimization and EDAs.

7.1 Large sets of interacting variables and high cardinality of the variables

One common challenging problem for EDAs and gray-box optimization are problems for which there are large interacting subsets of variables. Usually, FDAs and gray-box optimizers explicitly set constraints on the structural characteristics of the problem. For example, the gray-box optimization method introduced in [20] assumes that the number of subfunctions in which any variable appears must be bounded by some constant c. Similarly, some EDAs set a limit to the size of the biggest clique in the learned factorization [137, 140], or to the maximum number of neighbors a variable may have in the interaction graph [152]. The downside of this type of restrictions is that either the algorithms can not be applied to problems that do not respect the structural assumptions, or the behavior of the algorithms in these cases cannot be predicted. It is worth noticing that in some real-word problems some variables can play a critical role by appearing in a large number of subfunctions [175].

Another problematic issue is the cardinality of the variables. Most gray-box optimization methods and FDAs have been exclusively tested using binary problems. This restriction is doubly limiting, on one hand because this is the lowest cardinality for problems with discrete representation, on the other hand because binary problems also have the cardinality for all variables fixed. Therefore, problems with non-uniform cardinality of the variables are not usually addressed and it is not possible to assess how a non-uniform distribution of the variables cardinalities can impact the behavior of these algorithms. The most difficult scenario is comprised by problems that exhibit the two characteristics, they have large sets of interacting variables and high cardinality of the variables. For these problems it has been shown that even FDAs that use exact factorizations suffer an important impact in their performance compared with the scenario in which binary problems are solved [140].

7.2 Estimating or learning the structure of black-box problems

When no information about the problem structure exists, methods that learn a model from data are usually employed in EDAs. Probabilistic graphical models actually learn a representation of the dependence relationships among the variables and probabilistic patterns (e.g., marginal distributions) in the data. However, it is usually assumed in EDAs that the dependence relationships reflect the problem structure. The link between problem structure and probabilistic dependencies is not clearly defined because it depends on aspects such as the choice of the probabilistic model, the problem representation, the strength of selection, etc. To complicate the matter more, models that accurately capture the interactions between the problem variables are not always more effective than models that discard or ignore these interactions. Therefore, an algorithm can excel at recovering the problem structure and producing mediocre results in terms of the best solutions found.

The study of the link between problem structure and model dependencies is beyond the scope of this paper. The interested reader can consult [14, 15, 16, 35, 90, 93] for different perspectives on this topic. Similarly, an account of the different types of probabilistic models used in EDAs and how they represent the relationships between variables can be obtained from several surveys on EDAs [50, 70, 79, 151]. We focus our analysis on the recent renewed interested in the application of neural models in EDAs.

7.2.1 Neural and deep learning models in estimation of distribution algorithms

The use of neural networks (NNs) as models in EDAs is not new [85, 153, 157, 182]. However, an increasing number of recent works [5, 26, 123, 124, 130] propose the application of neural models which, in some cases, are also deep learning architectures. The particular characteristics of the neural models and the possibility of applying "deep neural models" in EDAs motivates the brief review included in this section. The main questions we address are: 1) What is essentially different in the information about the problem structure conveyed by neural models? 2) What advantages and disadvantages exhibit the neural model representations, and NN learning and sampling algorithms, within the context of model-based EAs?

A fundamental difference of NNs over graphical models is that information about the problem structure is usually represented in latent variables or distributed structures that make interpretability of the model a difficult task. From the perspective of the main topic investigated in this paper, the exploitation of the problem structural information for optimization, the question arises as to what extent a latent representation of the optimization problem can be efficiently exploited.

In classification problems, latent features produced by NN learning methods can play a very important role [27]. In addition to the direct use of the latent features produced by NNs for generating new solutions in the contexts EAs, there are potential benefits in extracting knowledge about the search space from these latent representations. An increasing number of methods are being proposed for interpreting neural models [92]. In particular, methods for the identification of feature interactions of arbitrary order from the analysis of the neural network weights have been recently proposed [164]. A number of papers that have applied neural models in EDAs have inspected and analyzed the models learned as a source of knowledge about the problem characteristics. Table 9 summarizes some relevant characteristics of a number of works that introduce neural models.

Although the main evident difference among the approaches shown in Table 9 is the type of neural network used by the algorithms, significant differences can exist in the particular way the neural models are applied. In traditional EDAs based on probabilistic graphical models, different methods can be used for learning the model. Perhaps the best known example is the variety of learning algorithms employed by EDAs based on Bayesian networks [36, 102, 115]). Something similar applies for sampling methods in EDAs, where even for a specific Markov network based EDA, different sampling procedures produce significantly distinct behaviors of the algorithm [143].

In EDAs that use neural models, these differences can be more noticeable because there can be multiple ways to exploit the information contained in a NN. On top of that, the NN can be applied in multiple ways. For instance, autoencoders [7, 167], a particular class of NNs, have been used to sample new solutions in a way that resembles sampling in EDAs [121], but also to adaptively learn a "non-linear mutation operator" that attempts to widen the basins of attraction around the best individuals. In terms of the EDA performance, the potential of the neural model to capture intricate relationships among the variables is as relevant as the specific way it is utilized within the evolutionary algorithm.

Algorithm	year	NN model	Ref.	Analysis Str.	Generative	Deep
BEA	2000	HM	[182]	no	yes	no
MONEDA	2008	GNG	[85]	no	no	no
RBM-EDA	2010	RBM	[157]	no	no	no
REDA	2013	RBM	[154]	no	yes	no
DAGA	2014	DA	[25]	yes	no	no
DBM-EDA	2015	DBM	[123]	yes	yes	yes
AED-EDA	2015	DA	[121]	yes	no	no
GAN-EDA	2015	GAN	[122]	no	yes	no
GA-dA	2016	DA	[26]	yes	no	no
GA-NADE	2016	NADE	[26]	yes	yes	no
RBM-EDA	2017	RBM	[124]	yes	yes	no
Deep-Opt-GA	2017	DNN	[5]	yes	yes	yes

Table 9: Description of some of the main neural network models proposed for EDAs. Algorithms are organized following the chronological order of the publications. NN model refers to the neural model. HM: Helmzholtz machine; GNG: Growing Neural Gas; RBM: Restricted Boltzmann machine; DA: Denoising autoencoders; DBM: Deep Boltzmann machine; GAN: Generative adversarial network; NADE: Neural autoregressive distribution estimation; DNN: Deep NN. The rest of columns respectively indicate: whether the "structure" of the learned networks is inspected and discussed, whether the NN model is generative, whether the neural model used for the experiments is deep (contains 2 or more hidden layers).

One crucial dilemma in EDAs that learn complex graphical models is how to trade-off the pros and cons of the different classes of models. Bayesian networks are expensive to learn but can be efficiently sampled. Markov networks and factor graphs can be learned efficiently but usually require expensive iterative inference schemes (e.g., Gibbs sampling) to sample new solutions [95]. In order for a neural model to be competitive with state-of-the-art graphical models used in EDAs, they should be able to capture intricate relationships and, in addition, their learning and sampling algorithms should outperform in efficiency those for graphical models.

The neural models that have been tried for EDAs, exhibit a variety of behaviors: autoencoders used in a traditional EDA scheme in [121] are extremely fast when compared with methods that learn Bayesian networks but they fail to achieve the same efficiency than BOA in terms of function evaluations. When used as a mutation distribution in [26] GA-dA (see Table 9) outperforms BOA in some problems (notably on the knapsack problem) but it is outperformed on the hierarchical HIFF function. As previously mentioned, for the performance of the EDA the class of model used might be as relevant as the particular way it is used. Some of the neural models that have been tried, such as the generative adversarial network (GAN), do not produce competitive results, neither in the number of fitness evaluations nor in the computational time [122].

The convenience of using deep neural models is another important question to discern given the impressive results of DNNs in other domains. One of the conclusions obtained from the evaluation of the deep Boltzmann machine (DBM) neural network in EDAs is that the effort for learning the multi-layered DBM model does not seem to pay off for the optimization process [123]. Also in [5], where DNNs with 5 and 10 layers are used as neural models, it is acknowledged that the learning process can be time consuming. While the Deep-Opt-GA is evaluated across a set of diverse artificial and real-world problems, it is not possible to determine the gain of the algorithm over EDAs since it is compared to a fast local optimizer and a GA.

In the following, we summarize a number of advantages and disadvantages of the use of neural models in EDAs.

Advantages of approaches that use NN models:

- Flexible models. The type of dependencies of the neural model do not have to be specified a priori [5].
- Usually can be efficiently (in terms of time) learned [25, 121].
- The gains in efficiency by means of parallelism (e.g., by using GPU architectures) can be dramatic [26].
- They can be naturally applied to implement transfer learning approaches in EAs [25].

Disadvantages of approaches that use NN models:

- Depending on the model, sampling can be cumbersome and costly.
- NN models can be very sensitive to the initial parameters used for training.
- The NN model representation does not always correspond to the representation of the problem (e.g., RBMs use binary representation, autoencoders use continuous representation, etc.).
- The number of parameters required by the NN model can be very large (e.g., up to $2.5 \times n^2$ for autoencoders in some problems [25]).
- Regarding the number of parameters, overfitting can arise while learning the models [121].

7.3 Redefining problem structure for constrained and multi-objective problems

Two scenarios that present significant challenges for using the problem structure are multi-objective and constrained optimization problems.

Addressing multi-objective problems (MOPs) is one of the most difficult areas for the implementation of methods that exploit the problem structure. The main difficulty is due to the variety and complexity of the modeling scenarios. One primary question is: What “structure” is the most relevant for MOPs? The structure describing the relationships among non-dominated points, or the (grouped or consensual) structure of all the objective functions involved ? Even if we assume that both types of structures are important, the question of finding a usable common representation for all the objectives is itself a challenge. The variety of selection, replacement, and archiving operators used by MOEAs determines important effects in the relationship between the original problem structure and the interactions which are significant for the search. Finding a common ground for the investigation of the structure in this context is a very difficult task.

In EDAs, the question of how to take advantage of the problem structure for MOPs has been less explored than for single-objective optimization problems. Although several multi-objective EDAs (MO-EDAs) exist [9, 29, 72, 84, 119, 188, 189] they have been proposed mainly for problems with a continuous representation. The characteristics of the search space for these problems allow other types of approaches which are different to those used for discrete problems.

In terms of the representation of the problem structure for MOPs, one idea proposed in [139] was *the extension of the representational capabilities of the probabilistic model to include information about the objectives*. In [62], objectives were included as additional nodes of the EDA probabilistic graphical model, and edges/arcs were learned from data to capture the relationships among variables and objectives. This initial work was successful when tested on continuous MOPs. Another work in the same direction, but addressing the combinatorial multi-objective knapsack problem, was presented in [86]. In the proposed approach, the graphical model represents, in addition to the decision variables and objectives, the parameters of a local optimization method. Other recent work [180, 181] has investigated the structures of probabilistic models learned in different subproblems of the MOEA/D algorithm [184]. Although this work has shown that the models in MOEA/D capture the structure of the problem, the convenience of using these structures to create factorizations for the MOPs addressed in the paper was not addressed.

In [22], a multi-objective Hamming-ball hill climber is proposed for a MOP, where each objective is a pseudo-Boolean functions with k-bounded epistasis. This gray-box optimization algorithm uses as a representation of the problem structure a co-occurrence graph in which two variables are joined by an edge if they co-occur in any of the subfunctions for any of pseudo-Boolean functions. The introduced algorithm guarantees that each move is bounded in constant time if each variable appears in a constant number of subfunctions. The algorithm is tested on instances of the MNK-Landscape with two and three objectives and up to $100,000$ variables. Although it is impossible to determine how close the results are to the optimal Pareto front, it shows that they improve by increasing the radius of the hill climber, i.e., considering higher order dependencies among the variables.

The difficulty with the search in a constrained space is that the original interactions of the problems can be distorted due to the constraints. Similarly, constraints can produce new interactions among the variables. Early work investigated how to modify FDAs to deal with problems with unitation constraints [144, 145]. This research showed that while constraints can have an impact on the strength of dependencies among those variables related by the problem structure, it was still possible to exploit the original structure of the problem to make a more efficient search. Recent approaches propose the use of probability models defined exclusively on the space of feasible solutions [18].

As expected, constraints also have an effect on the behavior of gray-box optimizers. In [21], it is shown that the constant time per move of gray-box hill climbers is not guaranteed to be achieved when constraints are added to the problem. However, it is also shown in [21] that the proposed structure-informed hill-climber has as worst-case run-time $O(n)$ for constrained multi-objective k-bounded pseudo-Boolean problems.

7.4 The question of evolvability

Evolvability has been defined in numerous ways, and the implications of the term both in the biological and evolutionary computation domains are controversial. It can be defined as an organism's capacity to generate heritable phenotypic variation [66], the increased potential of an individual or population to further evolution [89], or the ability of random variations to sometimes produce improvement [168]. Recently, Wilder and Stanley [176] have advocated for making a difference between the concepts *evolvable individuals* and *evolvable populations*. Evolvable individuals are more likely than others to introduce phenotypic variation in their offspring, and in evolvable populations a greater amount of phenotypic variation is accessible to the population as a whole, regardless of how evolvable any individual may be in isolation [176].

From the perspective of the analysis developed in this paper, one key question is whether the use of the problem structure information in EAs hinders or promotes the evolvability of solutions and populations. The focus of gray-box optimizers, EDAs, and other algorithms that exploit the problem structure has been efficient function optimization, with the number of function evaluations as the main target. However, it is not clear whether designing model-based operators and algorithms that constrain the way genotypic variation is manifested through evolution can also decrease the capacity of these evolutionary systems to adapt to changes.

Much of function benchmarks on which the algorithms discussed in this paper have been tested exclusively comprise non-dynamic objective functions with a simple genotype-phenotype mapping. Consequently, it is difficult to evaluate how robust these algorithms would be to deal with dynamic functions, particularly with those functions where the structure of interactions changes over time. It can be argued that model-based EAs that learn the model from the data have a natural way to adapt to potential changes in the fitness function. The ability of gray-box optimizers to treat these problems is not clear. Whatever the capacity of these algorithms to deal with dynamic functions is, what remains as an open question is whether the learning and sampling methods used by model-based EAs constrain, promote, or are neutral in generating evolvable individuals or populations.

There is an intense debate around the question of whether and how evolvability evolves [13, 31, 66, 126]. Related questions are: how genes create phenotypes in such a way that options for future evolution are enhanced [13]?; is evolvability a consequence of adaptation and/or selection [31, 66, 155]?; is evolvability influenced by the existence of neutral networks in genotype space [32]?

In the context of model-based EAs one relevant topic is to what extent can the evolution of evolvability be modeled and eventually biased by exploiting the model. If evolvability can be evolved, is it possible to learn and exploit the structure of systems that evolve evolvability? Which models could be used to learn the patterns behind the individuals propensity to evolve? Another related question is whether problem structure plays any role in the algorithms that try to directly evolve for evolvability or in those EAs which indirectly encourage evolvability [63, 73, 74, 89].

7.5 Combination of gray-box optimization and EDAs

A straightforward way of combining gray-box optimization and EDAs is to use gray-box optimization as a local search technique within EDAs. This is the approach followed in [43], where the HBHC is combined with the Parameter-less population pyramid (P3) [42], a model-building EA. In fact, when successful for real-world and large optimization problems, the vast majority of EDAs require the application of local optimizers [104, 113, 138, 186]. One of the conclusions from the research presented in [43] is that gray-box optimization by itself may fail to obtain optimal results and the hypothesis that this may be due to the existence of plateaus is advanced. A simple hybrid algorithm combining uniform crossover with HBHC is able to outperform the HBHC.

Several works in EAs show the great success of a variety of local optimization methods [114, 125, 138] when combined with EDAs. Similarly, the emergence of new paradigms such as Optimal Mixing indicates that the possibilities to combine local search and problem structure have not been exhausted yet. Therefore, it will require an in-depth investigation to determine to what extent gray-box local search optimizers can outperform the results of the other hybrid approaches.

8 Conclusions

While pioneering work in EAs showed the importance of exploiting problem information, work in FDAs, gray-box optimization, optimal mixing, and other approaches reveals that, when information about the problem structure is available in the form of a graphical representation, a variety of methods can be designed to efficiently exploit this structure. The different ways in which these methods exploit the structure reveal the great, largely unexplored, potential of approaches that use graphical models.

In this paper we have reviewed algorithms that exploit the problem structure in evolutionary computation with a focus on gray-box optimizers and FDAs. We have used a common perspective to analyze these algorithms, highlighting the similarities and differences among them. We have contrasted the explanations of the EA behavior based on the schema analysis with those that emphasize the role of factors and factorizations, and presented examples of these characterizations at work. Optimal mixing algorithms, as a hybrid approach that combines characteristics from EDAs and GAs have been also covered.

The White-Gray-Black (WGB) classification of optimization problems, according to the type and extent of problem information available, has been introduced as a more refined way to identify the use of problem structure. A review of recently introduced neural approaches used as models has been conducted. Finally, the paper has addressed a number of open questions and hot-topics where opportunities for algorithms that exploit the problem structure arise.

Acknowledgment

References

[1] H. E. Aguirre and K. Tanaka. Working principles, behavior, and performance of MOEAs on MNK-landscapes. *European Journal of Operational Research*, 181(3):1670–1690, 2007.

[2] M. Alden and R. Miikkulainen. MARLEDA: effective distribution estimation through Markov random fields. *Theoretical Computer Science*, 633:4–18, 2016.

[3] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[4] S. Baluja. Incorporating a priori knowledge in probabilistic-model based optimization. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 205–222. Springer, 2006.

[5] S. Baluja. Deep learning for explicitly modeling optimization landscapes. *CoRR*, abs/1703.07394, 2017.

[6] D. Beasley, D. R. Bull, and R. R. Martin. Reducing epistasis in combinatorial problems by expansive coding. In *ICGA*, pages 400–407, 1993.

[7] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[8] P. A. Bosman and J. Grahl. Matching inductive search bias and problem structure in continuous estimation of distribution algorithms. *European Journal of Operational Research*, 185:1246–1264, 2008.

[9] P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.

[10] P. A. Bosman and D. Thierens. The roles of local search, model building and optimal mixing in evolutionary algorithms from a BBO perspective. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 663–670. ACM, 2011.

[11] P. A. Bosman and D. Thierens. Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2012*, pages 585–592. ACM, 2012.

[12] P. A. Bosman and D. Thierens. More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 359–366. ACM, 2013.

[13] J. Brookfield. Evolution: the evolvability enigma. *Current Biology*, 11(3):R106–R108, 2001.

[14] A. E. I. Brownlee, J. A. McCall, and L. A. Christie. Structural coherence of problem and algorithm: An analysis for EDAs on all 2-bit and 3-bit problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2066–2073. IEEE, 2015.

[15] A. E. I. Brownlee, J. A. McCall, and M. Pelikan. Influence of selection on structure learning in Markov network EDAs: An empirical study. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2012*, pages 249–256. ACM, 2012.

[16] A. E. I. Brownlee, J. A. McCall, S. K. Shakya, and Q. Zhang. Structure learning and optimisation in a Markov-network based estimation of distribution algorithm. In *Proceedings of the 2009 Congress on Evolutionary Computation CEC-2009*, pages 447–454, Norway, 2009. IEEE Press.

[17] M. V. Butz, M. Pelikan, X. Llorá, and D. E. Goldberg. Automated global structure extraction for effective local building block processing in XCS. *Evolutionary Computation*, 14(3):345–380, 2006.

[18] J. Ceberio, A. Mendiburu, and J. A. Lozano. A square lattice probability model for optimising the graph partitioning problem. In *Proceedings of the 2017 Congress on Evolutionary Computation CEC-2017*, San Sebastian, Spain, 2017. IEEE Press. In press.

[19] Y.-p. Chen, C.-Y. Chuang, and Y.-W. Huang. Inductive linkage identification on building blocks of different sizes and types. *International Journal of Systems Science*, 43(12):2202–2213, 2012.

[20] F. Chicano, D. Whitley, and A. M. Sutton. Efficient identification of improving moves in a ball for pseudo-Boolean problems. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 437–444. ACM, 2014.

[21] F. Chicano, D. Whitley, and R. Tinós. Efficient hill climber for constrained pseudo-Boolean optimization problems. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 309–316. ACM, 2016.

[22] F. Chicano, D. Whitley, and R. Tinós. Efficient hill climber for multi-objective pseudo-Boolean optimization. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 88–103. Springer, 2016.

[23] S. Choi, Y. Kwon, and B. Moon. Properties of symmetric fitness functions. *Evolutionary Computation, IEEE Transactions on*, 11(6):743–757, 2007.

[24] C.-Y. Chuang and Y.-p. Chen. Linkage identification by perturbation and decision tree induction. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 357–363. IEEE, 2007.

[25] A. W. Churchill, S. Sigtia, and C. Fernando. A denoising autoencoder that guides stochastic search. *CoRR*, abs/1404.1614, 2014.

[26] A. W. Churchill, S. Sigtia, and C. Fernando. Learning to generate genotypes with neural networks. *CoRR*, abs/1604.04153, 2016.

[27] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237. Barcelona, Spain, 2011.

[28] S. H. Clearwater and T. Hogg. Exploiting problem structure in genetic algorithms. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, pages 1310–1315. AAAI Press, 1994.

[29] M. Costa and E. Minisci. MOPED a multi-objective Parzen-based estimation of distribution algorithm for continuous problems. In *Evolutionary Multi-Criterion Optimization: Second International Conference, EMO-2003*, volume 2632 of *Lecture Notes in Computer Science*, page 71. Springer Berlin-Heidelberg, 2003.

[30] K. De Jong. Learning with genetic algorithms: An overview. *Machine learning*, 3(2-3):121–138, 1988.

[31] D. J. Earl and M. W. Deem. Evolvability is a selectable trait. *Proceedings of the National Academy of Sciences of the United States of America*, 101(32):11531–11536, 2004.

[32] M. Ebner, P. Langguth, J. Albert, M. Shackleton, and R. Shipman. On neutral networks and evolvability. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 1–8. IEEE, 2001.

[33] C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano. Toward understanding EDAs based on Bayesian networks through a quantitative analysis. *IEEE Transactions on Evolutionary Computation*, 16(2):173–189, 2012.

[34] C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano. On the taxonomy of optimization problems under estimation of distribution algorithms. *Evolutionary Computation*, 21(3):471–495, 2013.

[35] C. Echegoyen, Q. Zhang, A. Mendiburu, R. Santana, and J. A. Lozano. On the limits of effectiveness in estimation of distribution algorithms. In *Proceedings of the 2011 Congress on Evolutionary Computation CEC-2007*, pages 1573–1580. IEEE Press, 2011.

[36] R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 332–339. Editorial Academia. Havana, Cuba, 1999.

[37] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *Machine Learning*, 13(2/3):285–319, 1993.

[38] M. Gallagher and B. Yuan. A general-purpose tunable landscape generator. *Evolutionary Computation, IEEE Transactions on*, 10(5):590–603, 2006.

[39] Y. Gao and J. C. Culberson. Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143, 2005.

[40] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[41] D. E. Goldberg, K. Deb, and D. Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, 1993. Also IlliGAL Report No. 92009.

[42] B. W. Goldman and W. F. Punch. Parameter-less population pyramid. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 785–792. ACM, 2014.

[43] B. W. Goldman and W. F. Punch. Gray-box optimization using the parameter-less population pyramid. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 855–862. ACM, 2015.

[44] B. W. Goldman and D. R. Tauritz. Linkage tree genetic algorithms: variants and analysis. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 625–632. ACM, 2012.

[45] J. Grahl, S. Minner, and P. A. Bosman. Learning structure illuminates black boxes–an introduction to estimation of distribution algorithms. In *Advances in Metaheuristics for Hard Optimization*, pages 365–395. Springer, 2008.

[46] J. Grahl, S. Minner, and F. Rothlauf. Decomposition of dynamic single-product and multi-product lotsizing problems and scalability of EDAs. In *Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management*, pages 231–251. Springer, 2008.

[47] D. Hains, D. Whitley, A. Howe, and W. Chen. Hyperplane initialized local search for MAXSAT. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 805–812. ACM, 2013.

[48] G. R. Harik and D. E. Goldberg. Learning linkage. *Foundations of Genetic Algorithms*, 4:247–262, 1996.

[49] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer, 2001.

[50] M. Hauschild and M. Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and evolutionary computation*, 1(3):111–128, 2011.

[51] M. Hauschild, M. Pelikan, K. Sastry, and D. E. Goldberg. Using previous models to bias structural learning in the hierarchical BOA. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2008*, pages 415–422, New York, NY, USA, 2008. ACM.

[52] M. W. Hauschild, M. Pelikan, K. Sastry, and D. E. Goldberg. Using previous models to bias structural learning in the hierarchical BOA. *Evolutionary Computation*, 20(1):135–160, 2012.

[53] R. B. Heckendorn. Embedded landscapes. *Evolutionary Computation*, 10(4):345–369, 2002.

[54] P. Heggernes. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297–317, 2006.

[55] B. H. Helmi, A. T. Rahmani, and M. Pelikan. A factor graph based genetic algorithm. *International Journal of Applied Mathematics and Computer Science*, 24(3):621–633, 2014.

[56] L. Hernando, F. Daolio, N. Veerapen, and G. Ochoa. Local optima networks for the permutation flowshop scheduling problem: makespan vs. total flow time. In *Proceedings of the 2017 Congress on Evolutionary Computation CEC-2017*, San Sebastian, Spain, 2017. IEEE Press. In press.

[57] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

[58] R. Höns. *Estimation of Distribution Algorithms and Minimum Relative Entropy*. PhD thesis, University of Bonn, Bonn, Germany, 2006.

[59] R. Höns. Using maximum entropy and generalized belief propagation in estimation of distribution algorithms. In S. Shakya and R. Santana, editors, *Markov Networks in Evolutionary Computation*, pages 175–190. Springer, 2012.

[60] M. Kaedi and N. Ghasem-Aghaee. Biasing Bayesian optimization algorithm using case based reasoning. *Knowledge-Based Systems*, 24(8):1245–1253, 2011.

[61] L. Kallel, B. Naudts, and R. Reeves. Properties of fitness functions and search landscapes. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 177–208. Springer, 2000.

[62] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Multiobjective estimation of distribution algorithm based on joint modeling of objectives and variables. *IEEE Transactions on Evolutionary Computation*, 18(4):519–542, 2014.

[63] N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences (PNAS)*, 102(39):13773–13778, 2005.

[64] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann. Detecting funnel structures by means of exploratory landscape analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 265–272. ACM, 2015.

[65] R. Kikuchi. A theory of cooperative phenomena. *Physical Review*, 81(6):988–1003, 1951.

[66] M. Kirschner and J. Gerhart. Evolvability. *Proceedings of the National Academy of Sciences*, 95(15):8420–8427, 1998.

[67] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[68] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[69] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAA-IK-4/99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.

[70] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on probabilistic graphical models in evolutionary computation. *Journal of Heuristics*, 18(5):795–819, 2012.

[71] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[72] M. Laumanns and J. Ocenasek. Bayesian optimization algorithm for multi-objective optimization. In J. J. Merelo, P. Adamidis, H. G. Beyer, J. L. Fernández-Villacañas, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 298–307, Granada, Spain, 2002. Springer.

[73] J. Lehman and R. Miikkulainen. Enhancing divergent search through extinction events. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 951–958. ACM, 2015.

[74] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[75] C. F. Lima, M. Pelikan, F. G. Lobo, and D. E. Goldberg. *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, chapter Loopy Substructural Local Search for the Bayesian Optimization Algorithm, pages 61–75. Springer, Berlin Heidelberg, 2009.

[76] C. F. Lima, M. Pelikan, K. Sastry, M. V. Butz, D. E. Goldberg, and F. G. Lobo. Substructural neighborhoods for local search in the Bayesian optimization algorithm. In T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. M. Guervós, L. D. Whitley, and X. Yao, editors, *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN IX)*, volume 4193 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 2006.

[77] M. López-Ibánez, A. Liefooghe, and S. Verel. Local optimal sets and bounded archiving on multi-objective NK-landscapes with correlated objectives. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 621–630. Springer, 2014.

[78] L. Lozada-Chang and R. Santana. Univariate marginal distribution algorithm dynamics for a class of parametric functions with unitation constraints. *Information Sciences*, 181(11):2340–2355, 2011.

[79] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer, 2006.

[80] H. N. Luong, H. T. Nguyen, and C. W. Ahn. Entropy-based substructural local search for the Bayesian optimization algorithm. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 335–342. ACM, 2010.

[81] T. Mahnig. *Populations basierte Optimierung durch das Lernen von Interaktionen mit Bayes'schen Netzen*. PhD thesis, University of Bonn, Sankt Augustin, Germany, 2001. GMD Research Series No. 3/2001.

[82] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka. Knowledge-based nonuniform crossover. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 22–27. IEEE, 1994.

[83] B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In *Proceedings of the fourth international conference on genetic algorithms*, pages 143–150. San Mateo, CA: Morgan Kauffman, 1991.

[84] L. Marti, J. Garcia, A. Berlanga, C. A. Coello, and J. M. Molina. On current model-building methods for multi-objective estimation of distribution algorithms: Shortcommings and directions for improvement. Technical Report GIAA2010E001, Department of Informatics of the Universidad Carlos III de Madrid, Madrid, Spain, 2010.

[85] L. Marti, J. García, A. Berlanga, and J. M. Molina. Introducing MONEDA: scalable multi-objective optimization with a neural estimation of distribution algorithm. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation GECCO-2008*, pages 689–696, New York, NY, USA, 2008. ACM.

[86] M. S. Martins, M. R. Delgado, R. Santana, R. Lüders, R. A. Gonçalves, and C. P. d. Almeida. HMOBEDA: Hybrid multi-objective Bayesian estimation of distribution algorithm. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 357–364. ACM, 2016.

[87] A. Mendiburu, R. Santana, and J. A. Lozano. Introducing belief propagation in estimation of distribution algorithms: A parallel framework. Technical Report EHU-KAT-IK-11/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country, October 2007.

[88] A. Mendiburu, R. Santana, and J. A. Lozano. Fast fitness improvements in estimation of distribution algorithms using belief propagation. In R. Santana and S. Shakya, editors, *Markov Networks in Evolutionary Computation*, pages 141–155. Springer, 2012.

[89] H. Mengistu, J. Lehman, and J. Clune. Evolvability search: directly selecting for evolvability in order to study and produce it. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 141–148. ACM, 2016.

[90] K. M. Mishra and M. Gallagher. Variable screening for reduced dependency modelling in Gaussian-based continuous estimation of distribution algorithms. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[91] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254, 1992.

[92] G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *CoRR*, abs/1706.07979, 2017.

[93] R. Morgan and M. Gallagher. Using landscape topology to compare continuous metaheuristics: A framework and case study on EDAs and ridge structure. *Evolutionary computation*, 20(2):277–299, 2012.

[94] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.

[95] H. Mühlenbein. Convergence theorems of estimation of distribution algorithms. In S. Shakya and R. Santana, editors, *Markov Networks in Evolutionary Computation*, pages 91–108. Springer, 2012.

[96] H. Mühlenbein and R. Höns. Approximate factorizations of distributions and the minimum relative entropy principle. In *Proceedings of the 7th annual workshop on Genetic and evolutionary computation*, pages 199–211. ACM, 2005.

[97] H. Mühlenbein and R. Höns. The factorized distributions and the minimum relative entropy principle. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 11–38. Springer, 2006.

[98] H. Mühlenbein and T. Mahnig. Convergence theory and applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology*, 7(1):19–32, 1998.

[99] H. Mühlenbein and T. Mahnig. The Factorized Distribution Algorithm for additively decomposed functions. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 301–313, Havana, Cuba, March 1999.

[100] H. Mühlenbein and T. Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376, 1999.

[101] H. Mühlenbein and T. Mahnig. Evolutionary computation and beyond. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 123–188. CSLI Publications, Stanford, California, 2001.

[102] H. Mühlenbein and T. Mahnig. Evolutionary synthesis of Bayesian networks for optimization. In M. Patel, V. Honavar, and K. Balakrishnan, editors, *Advances in Evolutionary Synthesis of Intelligent Agents*, pages 429–455. MIT Press, Cambridge, Mass., 2001.

[103] H. Mühlenbein and T. Mahnig. Evolutionary algorithms and the Boltzmann distribution. In K. A. DeJong, R. Poli, and J. Rowe, editors, *Foundation of Genetic Algorithms 7*, pages 133–150. Morgan Kaufmann, 2002.

[104] H. Mühlenbein and T. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal on Approximate Reasoning*, 31(3):157–192, 2002.

[105] H. Mühlenbein, T. Mahnig, and A. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247, 1999.

[106] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lectures Notes in Computer Science*, pages 178–187, Berlin, 1996. Springer.

[107] C. Munteanu and A. Rosa. Symmetry at the genotypic level and the simple inversion operator. *Progress in Artificial Intelligence*, pages 209–222, 2007.

[108] A. Ochoa, H. Mühlenbein, and M. R. Soto. A Factorized Distribution Algorithm using single connected Bayesian networks. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI*

6th International Conference, Paris, France, September 16-20 2000. Springer. Lecture Notes in Computer Science 1917.

[109] A. Ochoa, M. R. Soto, R. Santana, J. Madera, and N. Jorge. The factorized distribution algorithm and the junction tree: A learning perspective. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 368–377, Havana, Cuba, March 1999. Editorial Academia. Havana, Cuba.

[110] T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer. Voronoi-based estimation of distribution algorithm for multi-objective optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation CEC-2004*, pages 1594–1601, Portland, Oregon, 2004. IEEE Press.

[111] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2000.

[112] R. K. Pearson and M. Pottmann. Gray-box identification of block-oriented nonlinear models. *Journal of Process Control*, 10(4):301–315, 2000.

[113] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*, volume 170 of *Studies in Fuzziness and Soft Computing*. Springer, 2005.

[114] M. Pelikan and D. E. Goldberg. Hierarchical BOA solves Ising spin glasses and MAXSAT. In *Genetic and Evolutionary Computation GECCO-2003*, pages 1271–1282. Springer, 2003.

[115] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, volume I, pages 525–532, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[116] M. Pelikan and M. Hauschild. Transfer learning, soft distance-based bias, and the hierarchical BOA. MEDAL Report No. 2012004, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), March 2012.

[117] M. Pelikan and M. W. Hauschild. Learn from the past: Improving model-directed optimization by transfer learning based on distance-based bias. MEDAL Report No. 2012007, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), 2012.

[118] M. Pelikan, M. W. Hauschild, and D. Thierens. Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1005–1012. ACM, 2011.

[119] M. Pelikan, K. Sastry, and D. E. Goldberg. Multiobjective estimation of distribution algorithms. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 223–248. Springer, 2006.

[120] S. Picek, R. I. McKay, R. Santana, and T. D. Gedeon. Fighting the symmetries: The structure of cryptographic boolean function spaces. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 457–464, Madrid, Spain, 2015.

[121] M. Probst. Denoising autoencoders for fast combinatorial black box optimization. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1459–1460. ACM, 2015.

[122] M. Probst. Generative adversarial networks in estimation of distribution algorithms for combinatorial optimization. *CoRR*, abs/1509.09235, 2015.

[123] M. Probst and F. Rothlauf. Deep Boltzmann machines in estimation of distribution algorithms for combinatorial optimization. *CoRR*, abs/1509.06535, 2015.

[124] M. Probst, F. Rothlauf, and J. Grahl. Scalability of using restricted Boltzmann machines for combinatorial optimization. *European Journal of Operational Research*, 256(2):368–383, 2017.

[125] E. Radetic, M. Pelikan, and D. E. Goldberg. Effects of a deterministic hill climber on hBOA. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 437–444. ACM, 2009.

[126] M. Radman, I. Matic, and F. Taddei. Evolution of evolvability. *Annals of the New York Academy of Sciences*, 870(1):146–155, 1999.

[127] M. C. Riff-Rojas. Evolutionary search guided by the constraint network to solve csp. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 337–342. IEEE, 1997.

[128] P. Rohlfshagen and J. A. Bullinaria. Identification and exploitation of linkage by means of alternative splicing. In *Linkage in Evolutionary Computation*, pages 189–223. Springer, 2008.

[129] K. L. Sadowski, P. A. Bosman, and D. Thierens. Learning and exploiting mixed variable dependencies with a model-based EA. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 4382–4389. IEEE, 2016.

[130] S. Saikia, L. Vig, A. Srinivasan, G. Shroff, P. Agarwal, and R. Rawat. Neuro-symbolic EDA-based optimisation using ILP-enhanced DBNs. *CoRR*, abs/1612.06528, 2017.

[131] R. Santana. A Markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 337–348, Dubrovnik, Croatia, 2003. Springer.

[132] R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.

[133] R. Santana, R. Armañanzas, C. Bielza, and P. Larrañaga. Network measures for information extraction in evolutionary algorithms. *International Journal of Computational Intelligence Systems*, 6(6):1163–1188, 2013.

[134] R. Santana, E. P. de León, and A. Ochoa. The incident edge model. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 352–359, Havana, Cuba, March 1999.

[135] R. Santana, P. Larrañaga, and J. A. Lozano. Interactions and dependencies in estimation of distribution algorithms. In *Proceedings of the 2005 Congress on Evolutionary Computation CEC-2005*, pages 1418–1425, Edinburgh, U.K., 2005. IEEE Press.

[136] R. Santana, P. Larrañaga, and J. A. Lozano. Challenges and open problems in discrete EDAs. Technical Report EHU-KZAA-IK-1/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country, October 2007.

[137] R. Santana, P. Larrañaga, and J. A. Lozano. Adaptive estimation of distribution algorithms. In C. Cotta, M. Sevaux, and K. Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 177–197. Springer, 2008.

[138] R. Santana, P. Larrañaga, and J. A. Lozano. Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem. *Journal of Heuristics*, 14:519–547, 2008.

[139] R. Santana, P. Larrañaga, and J. A. Lozano. Research topics on discrete estimation of distribution algorithms. *Memetic Computing*, 1(1):35–54, 2009.

[140] R. Santana, P. Larrañaga, and J. A. Lozano. Learning factorizations in estimation of distribution algorithms using affinity propagation. *Evolutionary Computation*, 18(4):515–546, 2010.

[141] R. Santana, R. I. McKay, and J. A. Lozano. Symmetry in evolutionary and estimation of distribution algorithms. In *Proceedings of the 2013 Congress on Evolutionary Computation CEC-2013*, Cancun, Mexico, 2013. IEEE Press. In Press.

[142] R. Santana, A. Mendiburu, and J. A. Lozano. Structural transfer using EDAs: An application to multi-marker tagging SNP selection. In *Proceedings of the 2012 Congress on Evolutionary Computation CEC-2012*, pages 3484–3491, Brisbane, Australia, 2012. IEEE Press. Best Paper Award of 2012 Congress on Evolutionary Computation.

[143] R. Santana, A. Mendiburu, and J. A. Lozano. Message passing methods for estimation of distribution algorithms based on Markov networks. In *Proceedings of the 4th Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO-2013)*, Lectures Notes in Computer Science, pages 419–430, Chennai, India, 2013. Springer. In press.

[144] R. Santana and A. Ochoa. Dealing with constraints with estimation of distribution algorithms: The univariate case. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 378–384, Havana, Cuba, March 1999.

[145] R. Santana, A. Ochoa, and M. R. Soto. Factorized Distribution Algorithms for functions with unitation constraints. In *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, pages 158–165, Havana, Cuba, March 2001.

[146] R. Santana, A. Ochoa, and M. R. Soto. The mixture of trees factorized distribution algorithm. In L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pages 543–550, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[147] K. Sastry and D. E. Goldberg. Designing competent mutation operators via probabilistic model building of neighborhoods. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 114–125. Springer, 2004.

[148] K. Sastry, M. Pelikan, and D. E. Goldberg. Decomposable problems, niching, and scalability of multiobjective estimation of distribution algorithms. *CoRR*, cs/0502057, 2005.

[149] J. Schwarz and J. Ocenasek. A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning. In *Proceedings of the 4th joint conference on knowledge-based software engineering*, pages 51–58. IOS Press, 2000.

[150] S. Shakya and J. McCall. Optimization by estimation of distribution with DEUM framework based on Markov random fields. *International Journal of Automation and Computing*, 4(3):262–272, 2007.

[151] S. Shakya and R. Santana. A review of estimation of distribution algorithms and Markov networks. In S. Shakya and R. Santana, editors, *Markov Networks in Evolutionary Computation*, pages 21–37. Springer, 2012.

[152] S. Shakya, R. Santana, and J. A. Lozano. A Markovianity based optimisation algorithm. *Genetic Programming and Evolvable Machines*, 13(2):159–195, 2012.

[153] V. A. Shim and K. C. Tan. Probabilistic graphical approaches for learning, modeling, and sampling in evolutionary multi-objective optimization. In *IEEE World Congress on Computational Intelligence*, pages 122–144. Springer, 2012.

[154] V. A. Shim, K. C. Tan, C. Y. Cheong, and J. Y. Chia. Enhancing the scalability of multi-objective optimization via restricted Boltzmann machine-based estimation of distribution algorithm. *Information Sciences*, 248:191–213, 2013.

[155] P. D. Sniegowski and H. A. Murphy. Evolvability. *Current Biology*, 16(19):R831–R834, 2006.

[156] M. R. Soto, A. Ochoa, S. Acid, and L. M. Campos. Bayesian evolutionary algorithms based on simplified models. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 360–367, Havana, Cuba, March 1999.

[157] H. Tang, V. Shim, K. Tan, and J. Chia. Restricted Boltzmann machine based algorithm for multi-objective optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[158] R. E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *Siam Journal of Computing*, 13(3):566–579, August 1984.

[159] D. Thierens. The linkage tree genetic algorithm. *Parallel Problem Solving from Nature–PPSN XI*, pages 264–273, 2011.

[160] D. Thierens and P. A. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2011*, pages 617–624, 2011.

[161] D. Thierens and P. A. Bosman. Hierarchical problem solving with the linkage tree genetic algorithm. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 877–884. ACM, 2013.

[162] D. Thierens, D. E. Goldberg, and A. G. Pereira. Domino convergence, drift, and the temporal-salience structure of problems. In *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, pages 535–540, Anchorage, AK, 1998.

[163] R. Tinós, D. Whitley, and F. Chicano. Partition crossover for pseudo-Boolean optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 137–149. ACM, 2015.

[164] M. Tsang, D. Cheng, and Y. Liu. Detecting statistical interactions from neural network weights. *CoRR*, abs/1705.04977, 2017.

[165] M. Tsuji, M. Munetomo, and K. Akama. A network design problem by a GA with linkage identification and recombination for overlapping building blocks. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 349–356. IEEE, 2007.

[166] Y.-F. Tung and T.-L. Yu. Theoretical perspective of convergence complexity of evolutionary algorithms adopting optimal mixing. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 535–542. ACM, 2015.

[167] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[168] G. P. Wagner and L. Altenberg. Perspective: complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

[169] S.-M. Wang, Y.-F. Tung, and T.-L. Yu. Investigation on efficiency of optimal mixing on various linkage sets. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2475–2482. IEEE, 2014.

[170] D. Whitley. Blind no more: Constant time non-random improving moves and exponentially powerful recombination. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 391–407. ACM, 2015.

[171] D. Whitley. Mk landscapes, NK landscapes, MAX-kSAT: A proof that the only challenging problems are deceptive. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 927–934. ACM, 2015.

[172] D. Whitley. Blind no more: Deterministic partition crossover and deterministic improving moves. In *Proceedings of the Companion Publication of the 2016 on Genetic and Evolutionary Computation Conference*, pages 515–532. ACM, 2016.

[173] D. Whitley and W. Chen. Constant time steepest descent local search with lookahead for NK-landscapes and MAX-kSAT. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 1357–1364. ACM, 2012.

[174] D. Whitley, F. Chicano, and B. W. Goldman. Gray box optimization for Mk landscapes (NK landscapes and MAX-kSAT). *Evolutionary computation*, 24(3):491–519, 2016.

[175] D. Whitley, A. E. Howe, and D. Hains. Greedy or not? best improving versus first improving stochastic local search for MAXSAT. In *AAAI*, 2013.

[176] B. Wilder and K. Stanley. Reconciling explanations for the evolution of evolvability. *Adaptive Behavior*, 23(3):171–179, 2015.

[177] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.

[178] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2002-35, Mitsubishi Electric Research Laboratories, August 2002.

[179] T.-L. Yu, D. E. Goldberg, K. Sastry, C. F. Lima, and M. Pelikan. Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary computation*, 17(4):595–626, 2009.

[180] M. Zangari-de Souza, R. Santana, A. Mendiburu, and A. Pozo. On the design of hard mUBQP instances. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 421–428. ACM, 2016.

[181] M. Zangari-de Souza, R. Santana, A. T. R. Pozo, and A. Mendiburu. MOEA/D-GM: using probabilistic graphical models in MOEA/D for solving combinatorial optimization problems. *CoRR*, abs/1511.05625, 2015.

[182] B. T. Zhang, , and S. Y. Shin. Bayesian evolutionary optimization using Helmholz machines. In *Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, volume 1917, pages 827–836. Springer, 2000.

[183] Q. Zhang. On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 8(1):80–93, 2004.

[184] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[185] Q. Zhang and J. Sun. Iterated local search with guided mutation. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 924–929. IEEE, 2006.

[186] Q. Zhang, J. Sun, E. Tsang, and J. Ford. Combination of guided local search and estimation of distribution algorithm for quadratic assignment problems. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2003*, pages 42–48, 2003.

[187] Q. Zhang, J. Sun, and E. P. K. Tsang. Evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 9(2):192–200, 2005.

[188] Q. Zhang, A. Zhou, and Y. Jin. RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008.

[189] A. Zhou, Q. Zhang, and Y. Jin. Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. *Evolutionary Computation, IEEE Transactions on*, 13(5):1167–1189, 2009.

[190] S. Zhou, R. B. Heckendorn, and Z. Sun. Detecting the epistatic structure of generalized embedded landscape. *Genetic Programming and Evolvable Machines*, 9(2):125–155, 2008.

[191] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1-4):373–395, 2004.