Missouri Estimation of Distribution Algorithms Laboratory

Introduction to Estimation of Distribution Algorithms

Martin Pelikan, Mark W. Hauschild, and Fernando G. Lobo

MEDAL Report No. 2012003

February 2012

Abstract

Estimation of distribution algorithms (EDAs) guide the search for the optimum by building and sampling explicit probabilistic models of promising candidate solutions. However, EDAs are not only optimization techniques; besides the optimum or its approximation, EDAs provide practitioners with a series of probabilistic models that reveal a lot of information about the problem being solved. This information can in turn be used to design problem-specific neighborhood operators for local search, to bias future runs of EDAs on a similar problem, or to create an efficient computational model of the problem. This chapter provides an introduction to EDAs as well as a number of pointers for obtaining more information about this class of algorithms.

Keywords

Estimation of distribution algorithms, evolutionary computation, graphical models, stochastic optimization.

Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis
One University Blvd., St. Louis, MO 63121
E-mail: pelikan@cs.umsl.edu
WWW: http://medal.cs.umsl.edu/

Introduction to Estimation of Distribution Algorithms

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis, St. Louis, MO 63121
E-mail: pelikan@cs.umsl.edu
WWW: http://www.cs.umsl.edu/~pelikan

Mark W. Hauschild
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis, St. Louis, MO 63121
E-mail: markhauschild@gmail.com
WWW: http://medal.cs.umsl.edu/personal.php?id=16

Fernando G. Lobo
Departamento de Engenharia Electrónica e Informática
e Centro de Investigação em Ambiente e Sustentabilidade (CENSE)
Faculdade de Ciências e Tecnologia
Universidade do Algarve
Campus de Gambelas, 8005-139 Faro, Portugal
E-mail: fernando.lobo@gmail.com
WWW: http://w3.ualg.pt/~flobo/

February 27, 2012

Abstract

Estimation of distribution algorithms (EDAs) guide the search for the optimum by building and sampling explicit probabilistic models of promising candidate solutions. However, EDAs are not only optimization techniques; besides the optimum or its approximation, EDAs provide practitioners with a series of probabilistic models that reveal a lot of information about the problem being solved. This information can in turn be used to design problem-specific neighborhood operators for local search, to bias future runs of EDAs on a similar problem, or to create an efficient computational model of the problem. This chapter provides an introduction to EDAs as well as a number of pointers for obtaining more information about this class of algorithms.

1 Introduction

Estimation of distribution algorithms (EDAs) [7, 53, 66, 84, 97, 108, 124, 130], also called probabilistic model-building genetic algorithms and iterated density estimation evolutionary algorithms, view optimization as a series of incremental updates of a probabilistic model, starting with the model encoding the uniform distribution over admissible solutions and ending with the model that

generates only the global optima. In the past decade and a half, EDAs have been applied to many challenging optimization problems [3, 5, 11, 31, 36, 51, 92, 79, 145, 157, 158, 175, 185]. In many of these studies, EDAs were shown to solve problems that were intractable with other techniques or no other technique could achieve comparable results. However, the motive for the use of EDAs in practice is not only that these algorithms can solve difficult optimization problems, but that in addition to the optimum or its approximation EDAs provide practitioners with a compact computational model of the problem represented by a series of probabilistic models [117, 64, 67]. These probabilistic models reveal a lot of information about the problem domain, which can in turn be used to bias optimization of similar problems, create problem-specific neighborhood operators, and many other tasks. While many metaheuristics exist that essentially sample implicit probability distributions by using a combination of stochastic search operators, the insight into the problem represented by the series of explicit probabilistic models of promising candidate solutions gives EDAs a clear edge over most other metaheuristics.

This chapter provides an introduction to EDAs. Additionally, the chapter presents numerous pointers for obtaining additional information about this class of algorithms.

The chapter is organized as follows. Section 2 outlines the basic procedure of an EDA. Section 3 presents a taxonomy of EDAs based on the type of decomposition encoded by the model and the type of local distributions used in the model. Section 4 reviews some of the most popular EDAs. Section 5 discusses major research directions and past results in theoretical modeling of EDAs. Section 6 focuses on efficiency enhancement techniques for EDAs. Section 7 gives pointers for obtaining additional information about EDAs. Section 8 summarizes and concludes the chapter.

2 Basic EDA Procedure

2.1 Problem Definition

An optimization problem may be defined by specifying (1) a set of potential solutions to the problem and (2) a procedure for evaluating the quality of these solutions. The set of potential solutions is often defined using a general representation of admissible solutions and a set of constraints. The procedure for evaluating the quality of candidate solutions can either be defined as a function that is to be minimized or maximized (often referred to as an objective function or fitness function) or as a partial ordering operator. The task is to find a solution from the set of potential solutions that maximizes quality as defined by the evaluation procedure.

As an example, let us consider the quadratic assignment problem (QAP), which is one of the fundamental NP-hard combinatorial problems [86]. In QAP the input consists of distances between n locations and flows between n facilities. The task is to find a one-to-one assignment of facilities to locations so that the overall cost is minimized. The cost for a pair of locations is defined as the product of the distance between these locations and the flow between the facilities assigned to these locations; the overall cost is the sum of the individual costs for all pairs of locations. Therefore, in QAP, potential solutions are defined as permutations that define assignments of facilities to locations and the solution quality is evaluated using the cost function discussed above. The task is to minimize the cost. As another example, consider the maximum satisfiability problem for propositional logic formulas defined in conjunctive normal form with 3 literals per clause (MAX3SAT). In MAX3SAT, each potential solution defines one interpretation of propositions (making each proposition either true or false), and the quality of a solution is measured by the number of clauses that are satisfied by the specific interpretation. The task is to find an interpretation that maximizes the number of satisfied clauses.

Without additional assumptions about the problem, one way to find the optimum is to repeat three main steps:

1. *Generate* candidate solutions.
2. *Evaluate* the generated solutions.
3. *Update* the procedure for generating new candidate solutions according to the results of the evaluation.

Ideally, the quality of generated solutions would improve over time and after a reasonable number of iterations, the execution of these three steps would generate the global optimum or its accurate approximation. Different algorithms implement the above three steps in different ways, but the key idea remains the same—iteratively update the procedure for generating candidate solutions so that generated candidate solutions continually improve in quality.

2.2 EDA Procedure

In estimation of distribution algorithms (EDAs) the central idea is to maintain an *explicit probabilistic model* to represent the distribution over candidate solutions, and to adjust the model based on the results of the evaluation of these solutions so that it will generate better candidate solutions in future. Note that using an explicit probabilistic model makes EDAs quite different from many other metaheuristics, such as genetic algorithms [48, 74] or simulated annealing [29, 78], in which the probability distribution used to generate new candidate solutions is often defined *implicitly* by a search operator or a combination of several search operators. Researchers often distinguish two main types of EDAs:

Population-based EDAs. Population-based EDAs maintain a population (multiset) of candidate solutions, starting with a population generated at random according to the uniform distribution over all admissible solutions. Each iteration starts by creating a population of promising candidate solutions using the selection operator, which gives preference to solutions of higher quality. Any popular selection method for evolutionary algorithms can be used, such as truncation or tournament selection [26, 44]. For example, truncation selection can be used, which selects the top $\tau\%$ members of the population. A probabilistic model is then built for the selected solutions. New solutions are created by sampling the distribution encoded by the built model. The new solutions are then incorporated into the original population using a replacement operator. In full replacement, for example, the entire original population of candidate solutions is replaced by the new ones. A pseudocode of a population-based EDA is shown in Figure 1.

Incremental EDAs. In incremental EDAs, the population of candidate solutions is fully replaced by a probabilistic model. The model is initialized so that it encodes the uniform distribution over all admissible solutions. The model is then updated incrementally by repeating the process of (1) sampling several candidate solutions from the current model and (2) improving the model based on the evaluation of these candidate solutions and their comparison. A pseudocode of an incremental EDA is shown in Figure 2.

Incremental EDAs often generate only a few candidate solutions at a time, whereas population-based EDAs often work with a large population of candidate solutions, building each model from scratch. Nonetheless, it is easy to see that the two approaches are essentially the same because even the population-based EDAs can be reformulated in an incremental-based manner.

1. $t \leftarrow 0$
2. generate population $P(0)$ of random solutions
3. while termination criteria not satisfied, repeat
4. evaluate all candidate solutions in $P(t)$
5. select promising solutions $S(t)$ from $P(t)$
6. build a probabilistic model $M(t)$ for $S(t)$
7. generate new solutions $O(t)$ by sampling $M(t)$
8. create $P(t+1)$ by combining $O(t)$ and $P(t)$
9. $t \leftarrow t+1$

Figure 1: Population-based estimation of distribution algorithm.

1. $t \leftarrow 0$
2. initialize model $M(0)$ to represent the uniform distribution over admissible solutions
3. while termination criteria not satisfied, repeat
4. generate population $P(t)$ of candidate solutions by sampling $M(t)$
5. evaluate all candidate solutions in $P(t)$
6. create new model $M(t+1)$ by adjusting $M(t)$ according to evaluated $P(t)$
7. $t \leftarrow t+1$

Figure 2: Incremental estimation of distribution algorithm.

The main components of an EDA thus include (1) a selection operator for selecting promising solutions, (2) an assumed class of probabilistic models to use for modeling and sampling, (3) a procedure for learning a probabilistic model for the selected solutions, (4) a procedure for sampling the built probabilistic model, and (5) a replacement operator for combining the populations of old and new candidate solutions. The procedure for learning a probabilistic model usually requires two subcomponents: a metric for evaluating the probabilistic models from the assumed class, and a search procedure for choosing a particular model based on the metric used. EDAs differ mainly in the class of probabilistic models and the procedures used for evaluating candidate models and searching for a good model.

The general outline of an EDA is quite similar to that of a traditional evolutionary algorithm (EA) [38]; both guide the search toward promising solutions by iteratively performing selection and variation, the two key ingredients of any EA. In particular, components (1) and (5) are precisely the same as those used in other EAs. Components (2), (3), and (4), however, are unique to EDAs, and constitute their way of producing variation, as opposed to using recombination and mutation operators as is often done with other EAs.

As we shall see, this alternative perspective opens a way for designing search procedures from principled grounds by bringing to the evolutionary computation domain a vast body of knowledge from the machine learning literature, and in particular from probabilistic graphical models. The key idea of EDAs is to look at a population of previously visited good solutions as data, learn a model (or theory) of that data, and use the resulting model to infer where other good solutions might be. This approach is powerful, allowing a search algorithm to learn and adapt itself with respect to the optimization problem being solved, while it is being solved.

2.3 Simulation of an EDA by Hand

To better understand the EDA procedure, this section presents a simple EDA simulation by hand. The purpose of presenting the simulation is to clarify the components of the basic EDA procedure and to build intuition about the dynamics of an EDA run.

The simulation assumes that candidate solutions are represented by binary strings of fixed length $n > 0$. The objective function to maximize is onemax, which is defined as the sum of the bits in the input binary string $(X_1, X_2, \ldots, X_n)$:

$$f_{onemax}(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i, \tag{1}$$

The quality of a candidate solution improves with the number of 1s in the input string, and the optimum is the string of all 1s.

To model and sample candidate solutions, the simulation uses a *probability vector* [7, 76, 109]. A probability vector p for n-bit binary strings has n components, $p = (p_1, p_2, \ldots, p_n)$. The component p_i represents the probability of observing a 1 in position i of a solution string. To learn the probability vector, p_i is set to the proportion of 1s in position i observed in the selected set of solutions. To sample a new candidate solution $(X_1, X_2, \ldots, X_n)$, the components of the probability vector are polled and each X_i is set to 1 with probability p_i, and to 0 with probability $1 - p_i$.

The expected outcome of the learning and sampling of the probability vector is that the population of selected solutions and the population of new candidate solutions have the same proportion of 1s in each position. However, since the sampling considers each new candidate solution independently of others, the actual proportions may vary a little from their expected values. The probability-vector EDA described above is typically referred to as the *univariate marginal distribution algorithm (UMDA)* [108]; other EDAs [7, 62, 76] based on the probability vector model will be discussed in Section 4.1.

To keep the simulation simple, we consider a 5-bit onemax, a population of size $N = 6$, and truncation selection with threshold $\tau = 50\%$. Recall that the truncation selection with $\tau = 50\%$ selects the top half of the current population.

Figure 3 shows the first two iterations of the EDA simulation. The initial population of candidate solutions is generated at random. Truncation selection then selects the best 50% of candidate solutions based on their evaluation using onemax to form the set of promising solutions. Next, the probability vector is created based on the selected solutions and the distribution encoded by the probability vector is sampled to generate new candidate solutions. The resulting population replaces the original population and the procedure repeats.

In both iterations of the simulation, the average objective-function value in the new population is greater than the average value in the population before selection. The increase in the average quality of the population is good news for us because we want to maximize the objective function, but why does this happen? Since for onemax the solutions with more 1s are better than those with

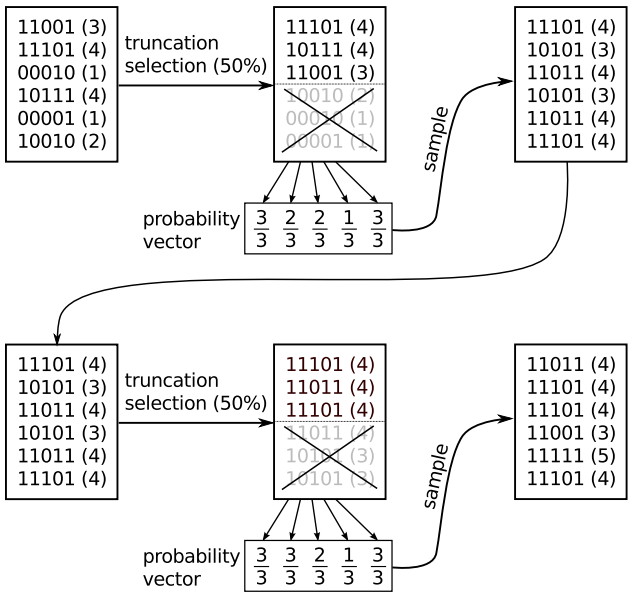


Figure 3: Simple simulation of an EDA based on the probability-vector model for onemax. The fitness values of candidate solutions are shown inside parentheses.

fewer 1s, selection should increase the number of 1s in the population. The learning and sampling of the probability vector is not expected to create or destroy any bits and that is why the new population of candidate solutions should contain more 1s than the original population (both in the proportion and in the actual number). Since onemax value increases with the number of 1s, we can expect the overall quality of the population to increase over time. Ideally, every iteration should increase the objective-function values in the population unless no improvement is possible.

Nonetheless, the increase of the average objective-function value tells only half the story. A similar increase in the quality of the population in the first iteration would be achieved by just repeating selection alone without the use of the probabilistic model. However, by applying selection alone, no new solutions are ever created and the resulting algorithm produces no variation at all. Since the initial population is generated at random, the EDA with selection alone would be just a poor algorithm for obtaining the best solution from the initial population. The learning and sampling of the probabilistic model provides a mechanism for both (1) improving quality of new candidate solutions (under certain assumptions), and (2) facilitating exploration of the set of admissible solutions.

What we have seen in this simulation was an example of the simplest kind of EDAs. The assumed class of probabilistic model, the probability vector, has a fixed structure. Under these circumstances, the procedure for learning it becomes trivial because there are really no alternative models to choose from. This class of EDAs is quite limited in what it can do. As we shall see in a moment, there are other classes of EDAs that allow richer probabilistic models capable of capturing interactions among the variables of a given problem. More importantly, these interactions can be learned automatically on a problem by problem basis. This results of course in a more complex model building procedure, but the extra effort has been shown to be well worth it, especially when solving more difficult optimization problems.

3 Taxonomy of EDA Models

This section provides a high-level overview of the distinguishing characteristics of probabilistic models. The characteristics are discussed with respect to (1) the types of interactions covered by the model and (2) the types of local distributions. This section only focuses on the key characteristics of the probabilistic models; a more detailed overview of EDAs for various representations of candidate solutions will be covered by the following sections.

3.1 Classification Based on Problem Decomposition

To make the estimation and sampling tractable with reasonable sample sizes, most EDAs use probabilistic models that *decompose* the problem using unconditional or conditional independence. The way in which a model decomposes the problem provides one important characteristic that distinguishes different classes of probabilistic models. Classification of probabilistic models based on the way they decompose a problem is relevant regardless of the types of the underlying distributions or the representation of problem variables.

Most EDAs assume that candidate solutions are represented by fixed-length vectors of variables and they use graphical models to represent the underlying problem structure. Graphical models allow practitioners to represent both direct dependencies between problem variables as well as independence assumptions. One way to classify graphical models is to consider a hierarchy of model types based on the complexity of a model (please see Figure 4 for illustrative examples) [66, 84, 124]:

1. **No dependencies.** In models that assume full independence, every variable is assumed to be independent of any other variable. That is, the probability distribution $P(X_1, X_2, \ldots, X_n)$ of the vector $(X_1, X_2, \ldots, X_n)$ of n variables is assumed to consist of a product of the distributions of individual variables:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i). \tag{2}$$

   The simulation presented in Section 2.3 was based on a model that assumed full independence of binary problem variables. EDAs based on univariate models that assume full independence of problem variables include the equilibrium genetic algorithm (EGA) [76], the population-based incremental learning (PBIL) [7], the univariate marginal distribution algorithm (UMDA) [108], the compact genetic algorithm [62], the stochastic hill climbing with learning by vectors of normal distributions [151], and the continuous PBIL [173].

2. **Pairwise dependencies.** In this class of models, dependencies between variables form a tree or forest graph. In a tree graph, each variable except for the root of the tree is conditioned on its parent in a tree that contains all variables. A forest graph, on the other hand, is a collection of disconnected trees. Again, the forest contains all problem variables. Denoting by R the set of roots of the trees in a forest, and by $X = (X_1, X_2, \ldots, X_n)$ the entire vector of variables, the distribution from this class can be expressed as:

$$P(X_1, X_2, \ldots, X_n) = \prod_{X_i \in R} P(X_i) \prod_{X_i \in X \setminus R} P\left(X_i | parent(X_i)\right) \tag{3}$$

   A special type of a tree model is sometimes distinguished, in which the variables form a sequence (or a chain), and each variable except for the first one depends directly on its predecessor.

Denoting by $\pi(i)$ the index of the ith variable in the sequence, the distribution is given by

$$P(X_1, X_2, \ldots, X_n) = P(X_{\pi(1)}) \prod_{i=2}^{n} P(X_{\pi(i)}|X_{\pi(i-1)}). \quad (4)$$

EDAs based on models with pairwise dependencies include the mutual information maximizing input clustering (MIMIC) [34], EDA based on dependency trees [8], and the bivariate marginal distribution algorithm (BMDA) [128].

3. **Multivariate dependencies.** Multivariate models represent dependencies using either directed acyclic graphs or undirected graphs. Two representative models are popular in EDAs: (1) Bayesian networks and (2) Markov networks. A Bayesian network is represented by a directed acyclic graph where each node corresponds to a variable and each edge defines a direct conditional dependence. The probability distribution encoded by a Bayesian network can be written as

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|parents(X_i)). \quad (5)$$

A Bayesian network represents problem decomposition by conditional independence assumptions; each variable is assumed to be independent of any of its antecedents in the ancestral ordering of the variables, given the values of the variable's parents. Note that all models discussed thus far were special cases of Bayesian networks. In fact, a Bayesian network can represent an arbitrary multivariate distribution; however, for such a model to be practical, it is often desirable to consider Bayesian networks of limited complexity.

In Markov networks (Markov random field models), two variables are assumed to be independent of each other given a subset of variables defining the condition if every path between these variables is separated by one or more variables in the condition.

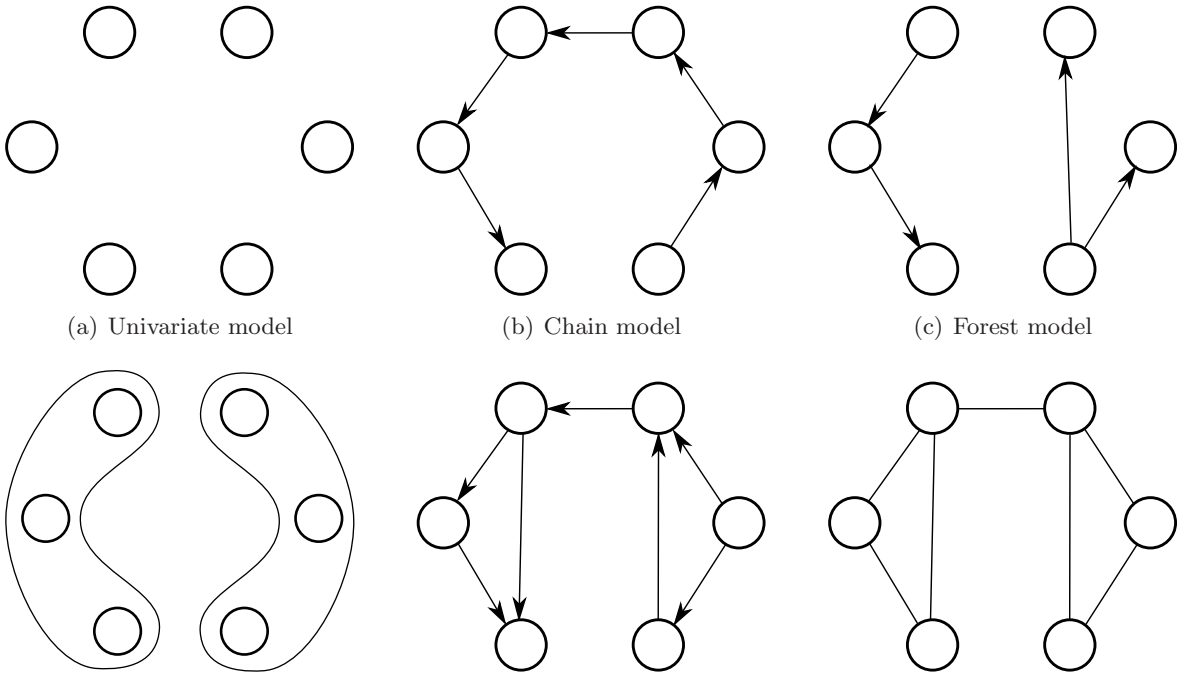
A special subclass of multivariate models is sometimes considered in which the variables are divided into disjoint clusters, which are independent of each other. These models are called marginal product models. Polytrees also represent a subclass of multivariate models in which a directed acyclic graph is used as the basic dependency structure but the graph is restricted so that at most one undirected path exists between any two vertices.

EDAs based on models with multivariate dependencies include the factorized distribution algorithm (FDA) [105], the learning FDA (LFDA) [105], the estimation of Bayesian network algorithm (EBNA) [39], the Bayesian optimization algorithm (BOA) [122, 123] and its hierarchical version (hBOA) [120], the extended compact genetic algorithm (ecGA) [60], the polytree EDA [184], the continuous iterated density estimation algorithm [21], the estimation of multivariate normal algorithm (EMNA) [83], and the real-coded BOA (rBOA) [2].

4. **Full dependence.** Models may be used that do not make *any* independence assumptions. However, such models must typically impose a number of other restrictions on the distribution to ensure that the models remain tractable for a moderate-to-large number of variables.

There are two additional types of probabilistic models that have been used in EDAs and that provide a somewhat different mechanism for decomposing the problem:

1. **Grammar models.** Some EDAs use stochastic or deterministic grammars to represent the probability distribution over candidate solutions. The advantage of grammars is that they



(a) Univariate model (b) Chain model (c) Forest model

(d) Marginal product model (e) Bayesian network (f) Markov network

Figure 4: Illustrative examples of graphical models. Problem variables are displayed as circles, dependencies are shown as edges between variables or clusters of variables.

allow modeling of variable-length structures. Because of this, grammar distributions are mostly used as the basis for implementing genetic programming using EDAs [99], which represents candidate solutions using labeled trees of variable sizes. Grammar models are used for example in the probabilistic-grammar based EDA for genetic programming [13], the program distribution estimation with grammar model (PRODIGY) [180] or the EDA based on probabilistic grammars with latent annotations [63].

2. **Feature-based models.** Feature-based models encode the distribution of the neighborhood of a candidate solution using position-independent substructures, which can be found in a variety of positions in fixed-length or variable-length solutions. This approach is used in the feature-based Bayesian optimization algorithm [94]. Other features may be discovered, encoded, and used for guiding the exploration of the space of candidate solutions. Model-directed neighborhood structures are also used in other EDA variants, as will be discussed in Section 6.2.

3.2 Classification Based on Local Distributions in Graphical Models

Regardless of how a graphical model decomposes the problem, each model must also assume one or more classes of distributions to encode local conditional and marginal distributions. Some of the most common classes of local distributions are discussed below:

1. **Probability tables.** For discrete representations, conditional and marginal probabilities can be encoded using *probability tables*, which define a probability for each relevant combination of values in each conditional or marginal probability term. This was the case for example in the simulation in Section 2.3, in which the probability distribution for each string position i was represented by the probability p_i of a 1; the probability of a 0 in the same position was simply

$1 - p_i$. As another example, in Bayesian networks, for each variable a probability table can be used to define conditional probabilities of any value of the variable given any combination of values of the variable's parents. While probability tables cannot directly represent continuous probability distributions, they can be used even for real-valued representations in combination with a discretization method that maps real-valued variables into discrete categories; each of the discrete categories can then be represented using a single probability entry. Probability tables are used for example in UMDA [108], BOA [123] and ecGA [60]. An example conditional probability table is shown in Figure 5.

2. **Decision trees or graphs, default tables.** To avoid excessively large probability tables when many probabilities are either similar or negligible, more advanced local structures such as decision trees, decision graphs or default tables may be used. In decision trees, for example, probabilities are stored in leaves of a decision tree in which each internal node represents a test on a variable and the children of the node correspond to the different outcomes of the test. Decision trees and decision graphs can also be used in combination with real-valued variables, in which the leaves store a continuous distribution in some way. More advanced structures such as decision trees and decision graphs are used for example in the decision-graph BOA (dBOA) [125], the hierarchical BOA (hBOA) [120], and the mixed BOA [110]. An example decision tree for representing conditional probabilities is shown in Figure 5.

3. **Multivariate, continuous distributions.** The normal distribution is by far the most popular distribution used in EDAs to represent univariate or multivariate distributions of real-valued variables. A multivariate normal distribution can encode a linear correlation between the variables using the covariance matrix, but it is often inefficient in representing many other types of interactions [15, 110]. Normal distributions were used in many EDAs for real-valued vectors [151, 173, 21, 83], although in many real-valued EDAs more advanced distributions were used as well. Examples of multivariate normal distributions are shown in Figure 6(a)-(c).

4. **Mixtures of distributions.** A mixture distribution consists of multiple components. Each component is represented by a specific local probabilistic model, such as a normal distribution, and each component is assigned a probability. Mixture distributions were used in EDAs especially to enable EDAs for real-valued representations to deal with real-valued distributions with multiple basins of attraction, in which a single-peak distribution does not suffice. Mixture distributions were used for example in the real-valued iterated density estimation algorithms [21] or the real-coded BOA [2]. The use of mixture distributions is more popular in EDAs for real-valued representations, although mixture distributions were also used to represent distributions over discrete representations in which the population consists of multiple dissimilar clusters [119] and in multiobjective EDAs [189, 132]. An example of a mixture of normal kernel distributions is shown in Figure 6(d).

5. **Histograms.** In a number of EDAs for real-valued representations, to encode local distributions, real-valued variables or sets of such variables are divided into rectangular regions using a histogram-like model, and a probability or a single probabilistic model is used to represent the distribution in each region. Histogram models can be seen as a special subclass of the decision-tree models for real-valued variables. In real-valued EDAs, histograms were used for example in the histogram-based continuous EDA [194]. Histogram models can also be used for other representations; for example, when optimizing permutations, histograms can be used to represent different relative ordering constraints and their importance with respect to solution quality [192, 193].

X2	X3	X4	p(X1 | X2, X3, X4)
0	0	0	0.75
0	0	0	0.25
0	0	0	0.25
0	0	0	0.25
1	1	1	0.20
1	1	1	0.20
1	1	1	0.20
1	1	1	0.20

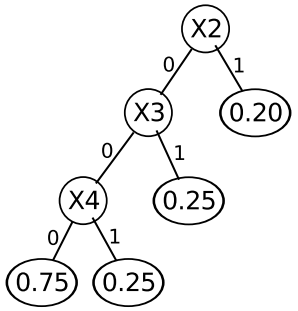


Figure 5: A conditional probability table for $p(X_1|X_2, X_3, X_4)$ and a corresponding decision tree that reduces the number of parameters (probabilities) from 8 to 4.

4 Overview of EDAs

This section gives an overview of EDAs based on the representation of candidate solutions, although some of the EDAs can be used across several representations. Due to the large volume of work in EDAs in the past two decades, we do not aim to list every single variant of an EDA discussed in the past; instead, we focus on some of the most important representatives.

4.1 EDAs for Fixed-Length Strings over Finite Alphabets

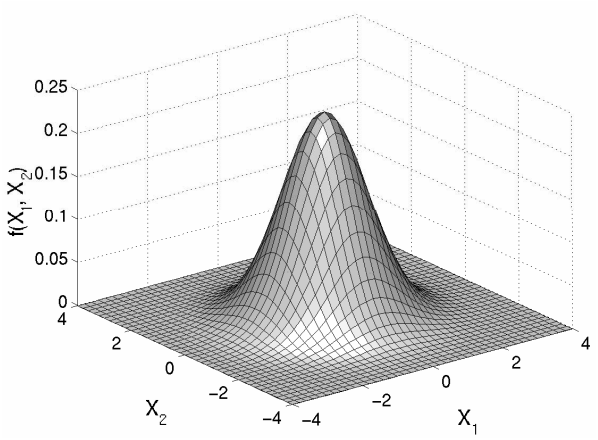
EDAs for candidate solutions represented by fixed-length strings over a finite alphabet can use a variety of model types, from simple univariate models to complex Bayesian networks with local structures. This section reviews some of the work in this area. Candidate solutions are assumed to be represented by binary strings of fixed length n, although most methods presented here can be extended to optimization of strings over an arbitrary finite alphabet. The section classifies EDAs based on the order of interactions in the underlying dependency model along the lines discussed in Section 3.1 [66, 84, 124].

4.1.1 No interactions

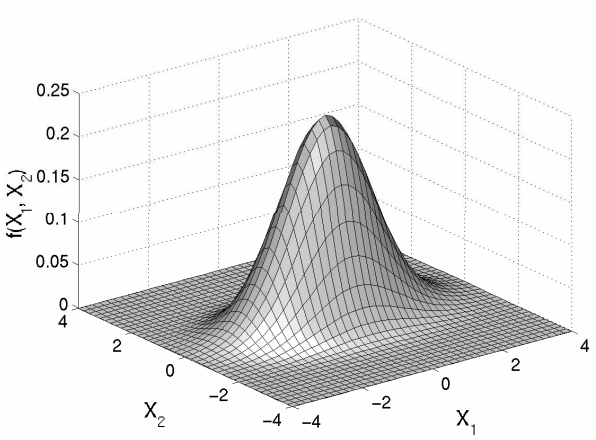
The equilibrium genetic algorithm (EGA) [76] and the population-based incremental learning (PBIL)[7] replace the population of candidate solutions represented as fixed-length binary strings by a probability vector $(p_1, p_2, \ldots, p_n)$, where n is the number of bits in a string and p_i denotes the probability of a 1 in the ith position of solution strings. Each p_i is initially set to 0.5, which corresponds to a uniform distribution over the set of all solutions. In each iteration, PBIL generates s candidate solutions according to the current probability vector where $s \geq 2$ denotes the selection pressure. Each value is generated independently of its context (remaining bits) and thus no interactions are considered (see Figure 4(a)). The best solution from the generated set of s solutions is then used to update the probability-vector entries using

$$p_i = p_i + \lambda(x_i - p_i),$$

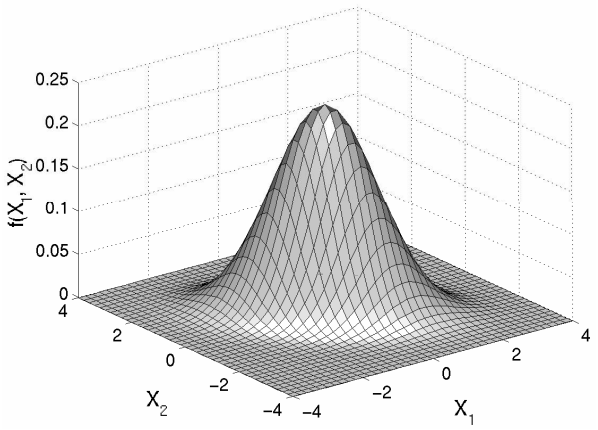
where $\lambda \in (0, 1)$ is the learning rate (say, 0.02), and x_i is the ith bit of the best solution. Using the above update rule, the probability p_i of a 1 in the ith position increases if the best solution contains a 1 in that position and decreases otherwise. In other words, probability-vector entries move *toward* the best solution and, consequently, the probability of generating this solution increases. The process of generating new solutions and updating the probability vector is repeated until some



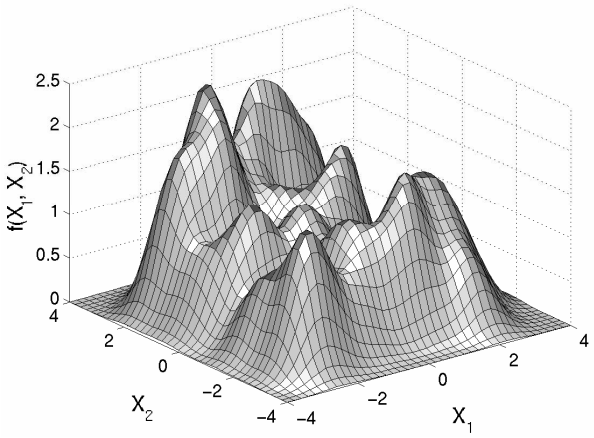
(a) Multivariate normal distribution with equal standard deviations and no covariance.



(b) Multivariate normal distribution with arbitrary standard deviations for each variable (diagonal covariance matrix).



(c) Multivariate normal distribution with an arbitrary (non-diagonal) covariance matrix.



(d) Joint normal kernels distribution.

Figure 6: Local models for continuous distributions over real-valued variables.

termination criteria are met; for instance, the run can be terminated if all probability-vector entries are sufficiently close to either 0 or 1.

Prior work refers to PBIL also as the hill climbing with learning (HCwL) [82] and the incremental univariate marginal distribution algorithm (IUMDA) [102].

PBIL is an incremental EDA, because it proceeds by executing incremental updates of the model using a small sample of candidate solutions. However, there is a strong correlation between the learning rate in PBIL and the population size in population-based EDAs or other evolutionary algorithms; essentially, decreasing the learning rate λ corresponds to increasing the population size.

The compact genetic algorithm (cGA) [62, 59] reduces the gap between PBIL and traditional steady-state genetic algorithms. Like PBIL, cGA replaces the population by a probability vector and all entries in the probability vector are initialized to 0.5. Each iteration updates the probability vector by mimicking the effect of a single competition between two sampled solutions, where the best replaces the worst, on a hypothetical population of size N. Denoting the bit in the ith position of the best and worst of the two sampled solutions by x_i and y_i, respectively, the probability-vector

entries are updated as follows:

$$p_i = \begin{cases} p_i + \frac{1}{N} & \text{if } x_i = 1 \text{ and } y_i = 0 \\ p_i - \frac{1}{N} & \text{if } x_i = 0 \text{ and } y_i = 1 \\ p_i & \text{otherwise} \end{cases}.$$

Although cGA uses a probability vector instead of a population, updates of the probability vector correspond to replacing one candidate solution by another one using a population of size N and shuffling the resulting population using a univariate model that assumes full independence of problem variables.

The univariate marginal distribution algorithm (UMDA) [109] maintains a population of solutions. Each iteration of UMDA starts by selecting a population of promising solutions using an arbitrary selection method of evolutionary algorithms. A probability vector is then computed using the selected population of promising solutions and new solutions are generated by sampling the probability vector. The new solutions replace the old ones and the process is repeated until termination criteria are met. Although UMDA uses a probabilistic model as an intermediate step between the original and new populations unlike PBIL and cGA, the performance, dynamics and limitations of PBIL, cGA, and UMDA are similar.

PBIL, cGA, and UMDA can solve problems decomposable into subproblems of order one in a linear or quadratic number of fitness evaluations. However, if decomposition into single-bit subproblems misleads the decision making away from the optimum, these algorithms scale up poorly with problem size [132, 187, 188].

4.1.2 Pairwise interactions

EDAs based on pairwise probabilistic models, such as a chain, a tree or a forest, represent the first step toward EDAs being capable of learning variable interactions and therefore solving decomposable problems of bounded order (difficulty) in a scalable manner.

The mutual-information-maximizing input clustering (MIMIC) algorithm [34] uses a chain distribution (see Figure 4(b)) specified by (1) an ordering of string positions (variables), (2) a probability of a 1 in the first position of the chain, and (3) conditional probabilities of every other position given the value in the previous position in the chain. A chain probabilistic model encodes the probability distribution where all positions except the first are conditionally dependent on the previous position in the chain. After selecting promising solutions and computing marginal and conditional probabilities, MIMIC uses a greedy algorithm to maximize mutual information between the adjacent positions in the chain. In this fashion the Kullback-Liebler divergence [81] between the chain and actual distributions is minimized. Nonetheless, the greedy algorithm does not guarantee global optimality of the constructed model (with respect to Kullback-Liebler divergence). The greedy algorithm starts in the position with the minimum unconditional entropy. The chain is expanded by adding a new position that minimizes the conditional entropy of the new variable given the last variable in the chain. Once the full chain is constructed for the selected population of promising solutions, new solutions are generated by sampling the distribution encoded by the chain.

There are two important drawbacks of using chain distributions. The first drawback is that chain distributions limit the expressiveness of probabilistic models by restricting dependencies between string positions that can be encoded. Despite that, chain distributions can encode dependencies between pairs of positions that can be located anywhere along the solution strings; these dependencies are not preserved by the univariate model-based EDAs. The second drawback is that there

is no known algorithm for learning the best chain distribution in polynomial time. Despite these disadvantages, the use of pairwise interactions was one of the most important steps in the development of EDAs capable of solving decomposable problems of bounded difficulty scalably. MIMIC was the first discrete EDA to not only learn and use a fixed set of statistics, but it was also capable of identifying the statistics that should be considered to solve the problem efficiently.

Baluja and Davies [8] use dependency trees (see Figure 4(b)) to model promising solutions. Like in PBIL, the population is replaced by a probability vector but in this case the probability vector contains all *pairwise* probabilities. The probabilities are initialized to 0.25. Each iteration adjusts the probability vector according to new promising solutions acquired on the fly. A dependency tree encodes the probability distribution where every variable except for the root is conditioned on the variable's parent in the tree. A variant of Prim's algorithm for finding the minimum spanning tree [138] can be used to construct an optimal tree distribution. Here the task is to find a tree that maximizes mutual information between parents (nodes with successors) and their children (successors). This can be done by first randomly choosing a variable to form the root of the tree, and "hanging" new variables to the existing tree so that the mutual information between the parent of the new variable and the variable itself is maximized. In this way, the Kullback-Liebler divergence between the tree and actual distributions is minimized as shown in ref. [32]. Once a full tree is constructed, new solutions are generated according to the distribution encoded by the constructed dependency tree and the conditional probabilities computed from the probability vector.

The bivariate marginal distribution algorithm (BMDA) [128] uses a forest distribution (a set of mutually independent dependency trees, see Figure 4(b)). This class of models is even more general than the class of dependency trees, because any forest that contains two or more disjoint trees cannot be generally represented by a tree. As a measure used to determine whether to connect two variables, BMDA uses a Pearson's chi-square test [98]. This measure is also used to discriminate the remaining dependencies in order to construct the final model. To learn a model, BMDA uses a variant of Prim's algorithm [138].

Pairwise models capture some interactions in a problem with reasonable computational overhead. EDAs with pairwise probabilistic models can identify, propagate, and juxtapose partial solutions of order two, and therefore they work well on problems decomposable into subproblems of order at most two [34, 8, 102, 128, 20]. Nonetheless, capturing only some pairwise interactions has still been shown to be insufficient for solving all decomposable problems of bounded difficulty scalably [128, 20].

4.1.3 Multivariate interactions

Using general multivariate models allows powerful EDAs capable of solving problems of bounded difficulty quickly, accurately, and reliably [84, 97, 105, 117, 130]. On the other hand, learning distributions with multivariate interactions necessitates more complex model-learning algorithms that require significant computational time and still do not guarantee global optimality of the resulting model. Nonetheless, many difficult problems are intractable using simple models and the use of complex models and algorithms is necessary.

The factorized distribution algorithm (FDA) [107] uses a fixed factorized distribution throughout the whole run. The model is allowed to contain multivariate marginal and conditional probabilities, but FDA learns only the probabilities, not the structure (dependencies and independencies). To solve a problem using FDA, we must first decompose the problem and then factorize the decomposition. While it is useful to incorporate prior information about the regularities in the search space, FDA necessitates that the practitioner is able to decompose the problem using a probabilistic

model ahead of time. FDA does not learn *what statistics* are important to process within the EDA framework, it must be given that information in advance. A variant of FDA where probabilistic models are restricted to polytrees was also proposed [184].

The extended compact genetic algorithm (ecGA) [60] uses a marginal product model (MPM) that partitions the variables into disjoint subsets (see Figure 4(d)). Each partition (subset) is treated as a single variable and different partitions are considered to be mutually independent. To decide between alternative MPMs, ecGA uses a variant of the minimum description length (MDL) metric [146, 147, 148], which favors models that allow higher compression of data (in this case, the selected set of promising solutions). More specifically, the Bayesian information criterion (BIC) [169] is used. To find a good model, ecGA uses a greedy algorithm that starts with each variable forming one partition (like in UMDA). Each iteration of the greedy algorithm merges two partitions that maximize the improvement of the model with respect to BIC. If no more improvement is possible, the current model is used. ecGA provides robust and scalable solution for problems that can be decomposed into independent subproblems of bounded order (separable problems) [163, 162, 165]. However, many real-world problems contain overlapping dependencies, which cannot be accurately modeled by dividing the variables into disjoint partitions; this can result in poor performance of ecGA.

The dependency-structure matrix genetic algorithm (DSMGA) [201, 202, 200] uses a similar class of models as ecGA that splits the variables into independent clusters or linkage groups. However, DSMGA builds models via dependency structure matrix clustering techniques.

The Bayesian optimization algorithm (BOA) [122] builds a Bayesian network for the population of promising solutions (see Figure 4(e)) and samples the built network to generate new candidate solutions. BOA uses the Bayesian-Dirichlet metric subject to a maximum model-complexity constraint [33, 70, 71] to discriminate competing models, but other metrics (such as BIC) have been analyzed in BOA as well. In all variants of BOA, the model is constructed by a greedy algorithm that iteratively adds a new dependency in the model that maximizes the model quality. Other elementary graph operators—such as edge removals and reversals—can be incorporated, but edge additions are most important. The construction is terminated when no more improvement is possible. The greedy algorithm used to learn a model in BOA is similar to the one used in ECGA. However, Bayesian networks can encode more complex dependencies and independencies than models used in ECGA. Therefore, BOA is also applicable to problems with overlapping dependencies. BOA uses an equivalent class of models as FDA; however, BOA learns both the structure and the probabilities of the model. Although BOA does not require problem-specific knowledge in advance, prior information about the problem can be incorporated using Bayesian statistics, and the relative influence of prior information and the population of promising solutions can be tuned by the user [65, 171].

A discussion of the use of Bayesian networks as an extension to tree models can also be found in Baluja's work [9]. An EDA that uses Bayesian networks to model promising solutions was independently developed by Etxeberria and Larrañaga [39], who called it the estimation of Bayesian network algorithm (EBNA). Mühlenbein and Mahnig [105] improved the original FDA by using Bayesian networks together with the greedy algorithm for learning the networks described above; the modification of FDA was named the learning factorized distribution algorithm (LFDA). An incremental version of BOA was proposed by Pelikan et al. [133].

The hierarchical BOA (hBOA) [120] extends BOA by employing local structures to represent local distributions instead of using standard conditional probability tables. This enables hBOA to represent distributions with high-order interactions. Furthermore, hBOA incorporates a niching technique called restricted tournament selection to ensure effective diversity preservation. The two

extensions enable hBOA to solve problems decomposable into subproblems of bounded order over a number of levels of difficulty of a hierarchy [196, 120].

Markov networks are yet another class of models that can be used to identify and use multivariate interactions in EDAs. Markov networks are undirected graphical models (see Figure 4(f)). Compared to Bayesian networks, Markov networks may sometimes cover the same distribution using fewer edges in the dependency model, but the sampling of these models becomes more complicated than the sampling of Bayesian networks. Markov networks are used for example in the Markov network EDA (MN-EDA) [156] and the density estimation using Markov random fields algorithm (DEUM) [176, 177].

Helmholtz machines used in the Bayesian evolutionary algorithm proposed by Zhang and Shin [204] can also encode multivariate interactions. Helmholtz machines encode interactions by introducing new, hidden variables, which are connected to every variable.

EDAs that use models capable of covering multivariate interactions can solve a wide range of problems in a scalable manner; promising results were reported on a broad range of problems, including several classes of spin-glass systems [117, 121, 127, 178], graph partitioning [106, 170, 171], telecommunication network optimization [150], silicon cluster optimization [162], scheduling [88], forest management [36], ground water remediation system design [4, 69], and others.

4.2 EDAs for Real-Valued Vectors

There are two basic approaches to extending EDAs for discrete fixed-length strings to other domains such as real-valued vectors:

1. Map the other representation to the domain of fixed-length discrete strings, solve the discrete problem, and map the solution back to the problem's original representation.
2. Extend or modify the class of probabilistic models to other domains.

A number of studies have been published about the mapping of real-valued representations into a discrete one in evolutionary computation [28, 30, 43, 48, 135]; this section focuses on EDAs from the second category. The approaches are classified along the lines presented in Section 3 [117, 124].

4.2.1 Single-peak normal distributions

The stochastic hill climbing with learning by vectors of normal distributions (SHCLVND) [151] is a straightforward extension of PBIL to vectors of real-valued variables using a normal distribution to model each variable. SHCLVND replaces the population of real-valued solutions by a vector of means $\mu = (\mu_1, \ldots, \mu_n)$, where μ_i denotes a mean of the distribution for the ith variable. The same standard deviation σ is used for all variables. See Figure 6(a) for an example model. At each generation, a random set of solutions is first generated according to μ and σ. The best solution out of this subset is then used to update the entries in μ by shifting each μ_i toward the value of the ith variable in the best solution using an update rule similar to the one used in PBIL. Additionally, each generation reduces the standard deviation to make the future exploration of the search space narrower. A similar algorithm was independently developed by Sebag and Ducoulombier [173], who also discussed several approaches to evolving a standard deviation for each variable.

4.2.2 Mixtures of normal distributions

The probability density function of a normal distribution is centered around its mean and decreases exponentially with square distance from the mean. If there are multiple "clouds" of values, a normal

distribution must either focus on only one of these clouds, or it can embrace multiple clouds at the expense of including the area between them. In both cases, the resulting distribution cannot model the data accurately. One way of extending standard single-peak normal-distribution models to enable coverage of multiple groups of similar points is to use a mixture of normal distributions. Each component of the mixture of normal distributions is a normal distribution by itself. A coefficient is specified for each component of the mixture to denote the probability that a random point belongs to this component. The probability density function of a mixture is thus computed by multiplying the density function of each mixture component by the probability that a random point belongs to the component, and adding these weighted densities together.

Gallagher et al. [40, 41] extended EDAs based on single-peak normal distributions by using an adaptive mixture of normal distributions to model each variable. The parameters of the mixture (including the number of components) evolve based on the discovered promising solutions. Using mixture distributions is a significant improvement compared to single-peak normal distributions, because mixtures allow simultaneous exploration of multiple basins of attraction for each variable.

Within the IDEA framework, Bosman and Thierens [21] proposed IDEAs using the joint normal kernels distribution, where a single normal distribution is placed around each selected solution (see Figure 6(d)). A joint normal kernels distribution can be therefore seen as an extreme use of mixture distributions with one mixture component per point in the training sample. The variance of each normal distribution can be either fixed to a relatively small value, but it should be preferable to adapt variances according to the current state of search. Using kernel distributions corresponds to using a fixed zero-mean normally distributed mutation for each promising solution as is often done in evolution strategies [143]. That is why it is possible to directly take up strategies for adapting the variance of each kernel from evolution strategies [143, 144, 172, 57].

4.2.3 Joint normal distributions and their mixtures

What changes when instead of fitting each variable with a separate normal distribution or a mixture of normal distributions, groups of variables are considered together? Let us first consider using a single-peak normal distribution. In multivariate domains, a joint normal distribution can be defined by a vector of n means (one mean per variable) and a covariance matrix of size $n \times n$. Diagonal elements of the covariance matrix specify the variances for all variables, whereas nondiagonal elements specify linear dependencies between pairs of variables. Considering each variable separately corresponds to setting all nondiagonal elements in a covariance matrix to 0. Using different deviations for different variables allows for "squeezing" or "stretching" the distribution along the axes. On the other hand, using nondiagonal entries in the covariance matrix allows rotating the distribution around its mean. Figures 6(b) and 6(c) illustrate the difference between a joint normal distribution using only diagonal elements of the covariance matrix and a distribution using the full covariance matrix. Therefore, using a covariance matrix introduces another degree of freedom and improves the expressiveness of a distribution. Again, one can use a number of joint normal distributions in a mixture, where each component consists of its mean, covariance matrix, and weight.

A joint normal distribution including a full or partial covariance matrix was used within the IDEA framework [21] and in the estimation of Gaussian networks algorithm (EGNA) [83]. Both these algorithms can be seen as extensions of EDAs that model each variable by a single normal distribution to use also nondiagonal elements of the covariance matrix.

Bosman and Thierens [22] proposed *mixed IDEAs* as an extension of EDAs that use a mixture of normal distributions to model each variable. Mixed IDEAs allow multiple variables to be modeled by a separate mixture of joint normal distributions. At one extreme, each variable can have a

separate mixture; at another extreme, one mixture of joint distributions covering all the variables is used. Despite that learning such a general class of distributions is quite difficult and a large number of samples is necessary for reasonable accuracy, good results were reported on single-objective [22] as well as multiobjective problems [189, 77, 85]. Using mixture models for all variables was also proposed as a technique for reducing model complexity in discrete EDAs [119].

Real-valued EDAs presented so far are applicable to real-valued optimization problems without requiring differentiability or continuity of the underlying problem. However, if it is possible to at least partially differentiate the problem, gradient information can be used to incorporate some form of gradient-based local search and the performance of real-valued EDAs can be significantly improved. A study on combining real-valued EDAs within the IDEA framework with gradient-based local search can be found in ref. [24].

One of the crucial limitations of using estimation of real-valued distributions is that real-valued EDAs have a tendency to lose diversity too fast even when the problem is relatively easy to solve [19]; for example, maximum likelihood estimation and sampling of a normal distribution will lead to diversity loss even while climbing a simple linear slope. That is why several EDAs were proposed that aim to control variance of the probabilistic model so that the loss of variance is avoided and yet the effective exploration is not hampered by an overly large variance of the model. For example, the adapted maximum-likelihood Gaussian model iterated density-estimation evolutionary algorithm (AMaLGaM) scales up the covariance matrix to prevent premature convergence on slopes [17, 18].

4.2.4 Other real-valued EDAs

Using normal distributions is not the only approach to modeling real-valued distributions. Other density functions are frequently used to model real-valued probability distributions, including histogram distributions, interval distributions, and others. A brief review of real-valued EDAs that use other than normal distributions follows.

In the algorithm proposed by Servet et al. [174], an interval (a_i, b_i) and a number $z_i \in (0, 1)$ are stored for each variable. By z_i, the probability that the ith variable is in the lower half of (a_i, b_i) is denoted. Each z_i is initialized to 0.5. To generate a new candidate solution, the value of each variable is selected randomly from the corresponding interval. The best solution is then used to update the value of each z_i. If the value of the ith variable of the best solution is in a lower half of (a_i, b_i), z_i is shifted toward 0; otherwise, z_i is shifted toward 1. When z_i gets close to 0, interval (a_i, b_i) is reduced to its lower half; if z_i gets close to 1, interval (a_i, b_i) is reduced to its upper half.

EDAs proposed in refs. [21, 195] use empirical histograms to model each variable as opposed to using a single normal distribution or a mixture of normal distributions. In these approaches, a histogram for each single variable is constructed. New points are then generated according to the distribution encoded by the histograms for all variables. The sampling of a histogram proceeds by first selecting a particular bin based on its relative frequency, and then generating a random point from the interval corresponding to the bin. It is straightforward to replace the histograms in the above methods by various classification and discretization methods of statistics and machine learning (such as k-means clustering) [28].

Pelikan et al. [126, 135] use an adaptive mapping from the continuous domain to the discrete one in combination with discrete EDAs. The population of promising solutions is first discretized using equal-width histograms, equal-height histograms, k-means clustering, or other classification techniques. A population of promising discrete solutions is then selected. New points are created by applying a discrete recombination operator to the selected population of promising discrete solutions. For example, new solutions can be generated by building and sampling a Bayesian

network like in BOA. The resulting discrete solutions are then mapped back into the continuous domain by sampling each class (a bin or a cluster) using the original values of the variables in the selected population of continuous solutions (before discretization). The resulting solutions are perturbed using one of the adaptive mutation operators of evolution strategies [143, 144, 172, 57]. In this way, competent discrete EDAs can be combined with advanced methods based on adaptive local search in the continuous domain. A related approach was proposed by Chen and Chen [30], who propose a split-on-demand adaptive discretization method to use in combination with ecGA.

The mixed Bayesian optimization algorithm (mBOA) developed by Ocenasek and Schwarz [110] models vectors of continuous variables using an extension of Bayesian networks with local structures. A model used in mBOA consists of a decision tree for each variable. Each internal node in the decision tree for a variable is a test on the value of another variable. Each test on a variable is specified by a particular value, which is also included in the node. The test considers two cases: the value of the variable is greater or equal than the value in the node or it is smaller. Each internal node has two children, each child corresponding to one of the two results of the test specified in this node. Leaves in a decision tree thus correspond to rectangular regions in the search space. For each leaf, the decision tree for the variable specifies a single-variable mixture of normal distributions centered around the values of this variable in the solutions consistent with the path to the leaf. Thus, for each variable, the model in mBOA divides the space reduced to other variables into rectangular regions, and it uses a single-variable normal kernels distribution to model the variable in each region. The adaptive variant of mBOA (amBOA) [114] extends mBOA by employing variance adaptation with the goal of maximizing effectiveness of the search for the optimum on real-valued problems.

4.3 EDAs for Genetic Programming

In genetic programming [80], the task is to solve optimization problems with candidate solutions represented by labeled trees that encode computer programs or symbolic expressions. Internal nodes of a tree represent functions or commands; leaves represent functions with no arguments, variables, and constants. There are two key challenges that one must deal with when applying EDAs to genetic programming. Firstly, the length of programs is expected to vary and it is difficult to estimate how large the solution will be without solving the problem first. Secondly, small changes in parent-child relationships often lead to large changes in the performance of a candidate solution, and often the relationship between nodes in the program trees is more important than their actual position. Despite these challenges, even in this problem domain, EDAs have been quite successful. In this section we briefly outline some EDAs for genetic programming.

The probabilistic incremental program evolution (PIPE) algorithm [153, 154] uses a probabilistic model in the form of a tree of a specified maximum allowable size. Nodes in the model specify probabilities of functions and terminals. PIPE does not capture any interactions between the nodes in the model. The model is updated by adjusting the probabilities based on the population of selected solutions using an update rule similar to the one in PBIL [7]. New program trees are generated in a top-down fashion starting in the root and continuing to lower levels of the tree. More specifically, if the model generates a function in a node and that function requires additional arguments, the successors (children) of the node are generated to form the arguments of the function. If a terminal is generated, the generation along this path terminates. An extension of PIPE named H-PIPE was later proposed [155]. In H-PIPE, nodes of a model are allowed to contain subroutines, and both the subroutines as well as the overall program are evolved.

Handley [56] used tree probabilistic models to represent populations of programs (trees) in

genetic programming. Although the goal of this work was to compress the population of computer programs in genetic programming, Handley's approach can be used within the EDA framework to model and sample candidate solutions represented by computer programs or symbolic expressions. A similar model was used in estimation of distribution programming (EDP) [199], which extended PIPE by employing parent-child dependencies in candidate labeled trees. More specifically, in EDP the content of each node is conditioned on the node's parent.

The extended compact genetic programming (ECGP) [164] assumes a maximum tree of maximum branching like PIPE. Nonetheless, ECGP uses a marginal product model which partitions nodes into clusters of strongly correlated nodes. This allows ECGP to capture and exploit interactions between nodes in program trees, and solve problems that are difficult for conventional genetic programming and PIPE. There are four main characteristics that distinguish ECGP and EDP. ECGP is able to capture dependencies between more than two nodes, it learns the dependency structure based on the promising candidate trees, and it is not restricted to the dependencies between parents and their children. On the other hand, ECGP is somewhat limited in the ability of efficiently encoding long-range interactions compared to probabilistic models that do not assume that groups of variables must be fully independent of each other.

Looks et al. [96] proposed to use Bayesian networks to model and sample program trees. Combinatory logic is used to represent program trees in a unified manner. Program trees translated with combinatory logic are then modeled with Bayesian networks of BOA, EBNA, and LFDA. Contrary to most other EDAs for genetic programming presented in this section, in the approach of Looks et al. the size of computer programs is not limited, but solutions are allowed to grow over time. Looks later developed a more powerful framework for competent program evolution using EDAs, which was named meta-optimizing semantic evolutionary search (MOSES) [94, 93, 95]. The key facets of MOSES include the division of the population into demes, the reduction of the problem of evolving computer programs to the one of building a representation with tunable features (knobs), and the use of hierarchical BOA [120] or another competent evolutionary algorithm to model demes and sample new candidate program solutions.

Several EDAs for genetic programming used probabilistic models based on grammar rules [13, 179, 180, 142]. Most grammar-based EDAs for genetic programming use a context-free grammar. The stochastic grammar-based genetic programming (SG-GP) [141, 142] started with a fixed context-free grammar with a default probability for each rule; the probabilities attached to the different rules were gradually adjusted based on the best candidate programs. The program evolution with explicit learning (PEEL) [179] used a probabilistic L-system with rules applicable at specific depths and locations; the probabilities of the rules were adapted using a variant of ant colony optimization (ACO) [35]. Another grammar-based EDA for genetic programming was proposed by Bosman and de Jong [13], who used a context-free grammar that is initialized to a minimum stochastic context-free grammar and adjusted to better fit promising candidate solutions by expanding rules and incorporating depth information into the rules. Grammar model-based program evolution (GMPE) [181, 180] also uses a probabilistic context-free grammar. In GMPE, new rules are allowed to be created and old rules may be eliminated from the model. A variant of the minimum-message-length metric is used in GMPE to compare grammars according to their quality. Tanev [186] incorporated stochastic context-sensitive grammars into the grammar-guided genetic programming [198, 197, 55].

4.4 EDAs for Permutation Problems

In many problems, candidate solutions are most naturally represented by permutations. This is the case for example in many scheduling or facility location problems. These types of problems often contain two specific types of features or constraints that EDAs need to capture. The first is the *absolute* position of a symbol in a string and the second is the *relative* ordering of specific symbols. In some problems, such as the traveling-salesman problem, relative ordering constraints matter the most. In others, such as the quadratic assignment problem, both the relative ordering and the absolute positions matter.

One approach to permutation problems is to apply an EDA for problems not involving permutations in combination with a mapping function between the EDA representation and the admissible permutations. For example, one may use the random key encoding [10] to transfer the problem of finding a good permutation into the problem of finding a high-quality real-valued vector, allowing the use of EDAs for optimization of real-valued vectors in solving permutation-based problems [25, 149]. Random key encoding represents a permutation as a vector of real numbers. The permutation is defined by the reordering of the values in the vector that sorts the values in ascending order. The main advantage of using random keys is that any real-valued vector defines a valid permutation and any EDA capable of solving problems defined on vectors of real numbers can thus be used to solve permutation problems. However, since EDAs do not process the aforementioned types of regularities in permutation problems directly their performance can often be poor [25, 23]. That is why several EDAs were developed that aim to encode either type of constraints for permutation problems explicitly.

To solve problems where candidate solutions are permutations of a string, Bengoetxea et al. [12] start with a Bayesian network model built using the same approach as in EBNA [39]. However, the sampling method is changed to ensure that only valid permutations are generated. This approach was shown to have promise in solving the inexact graph matching problem. In much the same way, the dependency-tree EDA (dtEDA) of Pelikan et al. [136] starts with a dependency-tree model [8, 32] and modifies the sampling to ensure that only valid permutations are generated. dtEDA for permutation problems was used to solve structured quadratic assignment problems with great success [136]. Bayesian networks and tree models are capable of encoding both the absolute position and the relative ordering constraints, although for some problem types, such models may turn out to be rather inefficient.

Bosman and Thierens [25] extended the real-valued EDA to the permutation domain by storing the dependencies between different positions in a permutation in the induced chromosome element exchanger (ICE). ICE works by first using a real-valued EDA, which encodes permutations as real-valued vectors using the random keys encoding. ICE extends the real-valued EDA by using a specialized crossover operator. By applying the crossover directly to permutations instead of simply sampling the model, relative ordering is taken into account. The resulting algorithm was shown to outperform many real-valued EDAs that use the random key encoding alone [25].

The edge histogram based sampling algorithm (EHBSA) [190, 193] works by creating an edge histogram matrix (EHM). For each pair of symbols, EHM stores the probabilities that one of these symbols will follow the other one in a permutation. To generate new solutions, EHBSA starts with a randomly chosen symbol. EHM is then sampled repeatedly to generate new symbols in the solution, normalizing the probabilities based on what values have already been generated. EHM does not take into account absolute positions at all; in order to address problems in which absolute positions are important, a variation of EHBSA that involved templates was proposed [190]. To generate new solutions, first a random string from the population was picked as a template. New

solutions were then generated by removing random parts of the template string and generating the missing parts with sampling from EHM. The resulting algorithm was shown to be better than most other EDAs on the traveling salesman problem. In another study, the node histogram sampling algorithm (NHBSA) of Tsutsui et al. [193] considers a model capable of storing node frequencies at each position (thereby encoding absolute position constraints) and also uses a template.

Zhang [206, 207] proposed to use guided mutation to optimize both permutation problems [152] as well as graph problems [207]. In guided mutation, the parts of the solution that are to be modified using a stochastic neighborhood operator are identified by analyzing a probabilistic model of the population of promising candidate solutions.

5 EDA Theory

Along with the design and application of EDAs, the theoretical understanding of these algorithms has improved significantly since the first EDAs were proposed. One way to classify key areas of theoretical study of EDAs follows [66]:

1. **Convergence proofs.** Some of the most important results in EDA theory focus on the number of iterations of an EDA on a particular class of problems or the conditions that allow EDAs to provably converge to a global optimum. The convergence time (number of iterations until convergence) of UMDA on onemax for selection methods with fixed selection intensity was derived by Mühlenbein and Schlierkamp-Voosen [101]. The convergence of FDA on separable additively decomposable functions (ADFs) was explored by Mühlenbein and Mahnig [104], who developed an exact formula for convergence time when using fitness-proportionate selection. Since in practice fitness-proportionate selection is rarely used because of its sensitivity to linear transformations of the objective function, truncation selection was also examined and an equation was derived giving the approximate time to convergence from the analysis of the onemax function. Later, Mühlenbein and Mahnig [105] adapted the theoretical model to the class of general ADFs where subproblems were allowed to interact. Under the assumption of Boltzmann selection, theory of graphical models was used to derive sufficient conditions for an FDA model so that FDA with a large enough population is guaranteed to converge to a model that generates only the global optima. Zhang [205] analyzed stability of fixed points of limit models of UMDA and FDA, and showed that at least for some problems the chance of converging to the global optimum is indeed increased when using higher order models of FDA rather than only the probability vector of UMDA. Convergence properties of PBIL were studied for example in refs. [52, 73, 82].

2. **Population sizing.** The convergence proofs mentioned above assumed infinite populations in order to simplify calculations. However, in practice using an infinite population is not possible and the choice of an adequate population size is crucial, similarly as for other population-based evolutionary algorithms [45, 46, 61, 58]. Using a population that is too small can lead to convergence to solutions of low quality and inability to reliably find the global optimum. On the other hand, using a population that is too large can lead to an increased complexity of building and sampling probabilistic models, evaluating populations, and executing other EDA components. Similar to genetic algorithms, EDAs must have a population size sufficiently large to provide an adequate initial supply of partial solutions in an adequate problem decomposition [46, 131] and to ensure that good decisions are made between competing partial solutions [58]. However, the population must also be large enough for EDAs to make good decisions about presence or absence of statistically significant variable interactions. To examine this topic, Pelikan et al. [131] analyzed the population size required for BOA to solve decomposable problems of

bounded difficulty with uniformly and nonuniformly scaled subproblems. The results showed that the population sizes required grew nearly linearly with the number of subproblems (or problem size). The results also showed that the approximate number of evaluations grew subquadratically for uniformly scaled subproblems but was quadratic on some nonuniformly scaled subproblems. Yu et al. [203] refined the model of Pelikan et al. [131] to provide a more accurate bound for the adequate population size in multivariate entropy-based EDAs such as ecGA and BOA, and also examined the effects of the selection pressure on the population size. Population sizing was also empirically analyzed in FDA by Mühlenbein [103].

3. **Diversity loss.** Stochastic errors in sampling can lead to a loss of diversity that may sometimes hamper EDA performance. Shapiro [182] examined the susceptibility of UMDA to diversity loss and discussed how it is necessary to set the learning parameters in such a way that this does not happen. Bosman et al. [14] examined diversity loss in EDAs for solving real-valued problems and the approaches to alleviating this difficulty. The results showed that due to diversity loss some of the state-of-the-art EDAs for real-valued problems could still fail on slope-like regions in the search space. The authors proposed using anticipated mean shift (AMS) to shift the mean of new solutions each generation in order to effectively maintain diversity.

4. **Memory complexity.** Another factor of importance in EDA problem solving is the memory required to solve the problem. Gao and Culberson [42] examined the space complexity of the FDA and BOA on additively decomposable functions where overlap was allowed between subfunctions. Gao and Culberson [42] proved that the space complexity of FDA and BOA is exponential in the problem size even with very sparse interaction between variables. While these results are somewhat negative, the authors point out that this only shows that EDAs have limitations and work best when the interaction structure is of bounded size. Note that one way to reduce the memory complexity of EDAs is to use incremental EDAs, such as PBIL [7], cGA [62] or iBOA [133].

5. **Model accuracy.** Model accuracy studies examine the accuracy of models in EDAs. Hauschild et al. [68] analyzed the models generated by hBOA when solving concatenated traps, random additively decomposable problems, hierarchical traps and two-dimensional Ising spin glasses. The models generated were then compared to the underlying problem structure by analyzing the number of spurious and correct dependencies. The results showed that the models corresponded closely to the structure of the underlying problems and that the models did not change significantly between consequent iterations of hBOA. The relationship between the probabilistic models learned by BOA and the underlying problem structure was also explored by Lima et al. [89]. One of the most important contributions of this study was to demonstrate the dramatic effect that selection has on spurious dependencies. The results showed that model accuracy was significantly improved when using truncation selection compared to tournament selection. Motivated by these results, the authors modified the complexity penalty of BOA model building to take into account tournament sizes when using binary tournament selection. Echegoyen et al. [37] also analyzed the structural accuracy of the models using EBNA on concatenated traps, two variants of Ising spin glass and MAXSAT. In this work two variations of EBNA were compared, one that was given the complete model structure based on the underlying problem and another that learned the approximate structure. The authors then examined the probability at any generation that the models would generate the optimal solution. The results showed that it was not strictly necessary to have all the interactions that were in the complete model in order to solve the problems. It was also discovered that in order for the algorithm to reach a solution, the probability of an optimal solution must always exceed a certain threshold. Finally, the effects

of spurious linkages on EDA performance were examined by Radetic and Pelikan [139]. The authors started by proposing a theoretical model to describe the effects of spurious (unnecessary) dependencies on the population sizing of EDAs. This model was then tested empirically on one-max and the results showed that while it would be expected that spurious dependencies would have little effect on population size, when niching was included the effects were substantial.

6 Efficiency enhancement techniques for EDAs

EDAs can solve many classes of important problems in a robust and scalable manner, oftentimes requiring only a low-order polynomial growth of the number of function evaluations with respect to the number of decision variables [50, 84, 97, 107, 131, 117, 130]. However, even a low-order polynomial complexity is sometimes insufficient for practical application of EDAs especially when the number of decision variables is extremely large, when evaluation of candidate solutions is computationally expensive, or when there are many conflicting objectives to optimize. The good news is that a number of approaches exist that can be used to further enhance efficiency of EDAs. Some of these techniques can be adopted from genetic and evolutionary algorithms with little or no change. However, some techniques are directly targeted at EDAs because these techniques exploit some of the unique advantages of EDAs over most other metaheuristics. Specifically, some efficiency enhancements capitalize on the facts that the use of probabilistic models in EDAs provides a rigorous and flexible framework for incorporating prior knowledge about the problem into optimization, and that EDAs provide practitioners with a series of probabilistic models that reveal a lot of information about the problem. This section reviews some of the most important efficiency enhancement techniques for EDAs with main focus on techniques designed specifically for EDAs.

6.1 Parallelization

One of the most straightforward approaches to speeding up any algorithm is to distribute the computation over a number of computational nodes so that several computational tasks can be executed in parallel. There are two main bottlenecks of EDAs that are typically addressed by parallelization: (1) fitness evaluation, and (2) model building and sampling. If fitness evaluation is computationally expensive, a master-slave architecture can be used for distributing fitness evaluations and collecting the results [27]. If most computational time is spent in model building and sampling, model building and sampling should be parallelized [84, 115, 112].

Many parallelization techniques and much of the theory can be adopted from research on parallelization in genetic and evolutionary algorithms [27]. In the context of EDAs, parallelization of model building was discussed for example by Ocenasek et al. [111, 113, 115, 112] who proposed the parallel BOA and by Larrañaga et al. [84] who parallelized model building in EBNA. One of the most impressive results in parallelization of EDAs was published by Sastry et al. [166, 51] who proposed a highly efficient, fully parallelized implementation of cGA to solve large-scale problems with millions to billions of variables even with a substantial amount of external noise in the objective function.

6.2 Hybridization

An optimization hybrid combines two or more optimizers in a single procedure [72, 183, 54]. Typically, a *global* procedure and a *local* procedure are combined; the global procedure is expected to find promising regions and the local procedure is expected to find local optima quickly within

reasonable basins of attraction. Global and local search are used in concert to find good solutions faster and more reliably than would be possible using either procedure alone.

Numerous studies have proposed to combine EDAs with variants of local search both in the discrete domain [117, 121, 140] and in the real-valued domain [16]. The main reason for combining EDAs with local search is that by reducing the search space to the local optima, the structure of the problem can be identified more easily and the population-sizing requirements can be significantly decreased [117, 121]. Furthermore, the search reduces to the space of basins of attraction around each local optimum as opposed to the space of all admissible solutions.

However, hybridization of EDAs is not restricted to the combination of an EDA with simple local search. As was already pointed out, probabilistic models often contain a lot of information about the problem. By mining these models for information about the structure and other properties of the problem landscape, decisions can be made about the nature and likely effectiveness of particular local search procedures and appropriate neighborhood structures for those procedures [91, 90, 100, 116, 159]. In turn, subsequent local search as well as the coordination of the global and local search in a hybrid can be managed so that excellent solutions are found quickly, reliably and accurately.

There are two main approaches to the design of EDA-based (model-directed) hybrids with advanced neighborhoods: (1) Belief propagation, which uses the probabilistic model to generate the maximum likely instance [90, 100, 116] and (2) local search with an advanced neighborhood structure derived from an EDA model [91, 159]. However, it is important to note that the use of EDA models is not limited to advanced neighborhood structures or belief propagation, and one may envision the use of probabilistic models to control the division of time resources between the global and local searcher and in a number of other tasks.

Local search based on advanced neighborhood structures in a hill-climbing like procedure [75, 137] is strongly related to model-directed hybridization using EDAs, although in this approach no estimation of distributions takes place. The basic idea is to use a linkage learning approach to detect important interactions between problem variables, and then run a local search based on a neighborhood defined by the underlying problem decomposition.

6.3 Time Continuation

To achieve the same solution quality, one may run an EDA or another population-based metaheuristic with a large population for one convergence epoch, or run the algorithm with a small population for a large number of convergence epochs with controlled restarts between these epochs [49]. Similar tradeoffs are involved in the design of efficient and reliable hybrid procedures where an appropriate division of computational resources between the component algorithms is critical. The term *time continuation* is used to refer to the tradeoffs involved [47].

Two important studies related to time continuation in EDAs were published by Sastry et al. [160, 161]. Based on a theoretical model of an ECGA-based hybrid, Sastry et al. showed that under certain assumptions, the neighborhoods created from EDA-built models provide sufficient information for local search to succeed on its own even on classes of problems for which local search with standard neighborhoods performs poorly. However, in many other cases, EDA-driven search in a hybrid with local search based on the adaptive neighborhood should perform better, especially if the structure of the problem is complex and the problem is affected by external noise.

One of the promising research directions related to time continuation in EDAs is to mine probabilistic models discovered by EDAs to find an optimal way to exploit time continuation tradeoffs, be it in an EDA alone or in an EDA-based hybrid.

6.4 Using Prior Knowledge and Learning from Experience

The use of prior knowledge has had longstanding study and use in optimization. For example, promising partial solutions may be used to bias the initial population of candidate solutions, specialized search operators can be designed to solve a particular class of problems, or representations can be biased in order to make the search for the optimum an easier task. However, one of the limitations of most of these approaches is that the prior knowledge must be incorporated by hand and the approaches are limited to one specific problem domain.

The use of probabilistic models provides EDAs with a unique framework for incorporating prior knowledge into optimization because of the possibility of using Bayesian statistics to combine prior knowledge with data in the learning of probabilistic models [6, 64, 171]. Furthermore, the use of probabilistic models in EDAs provides a basis for learning from previous runs in order to solve new problem instances of similar type with increased speed, accuracy and reliability [64, 67, 117]. For example, Hauschild and Pelikan [65, 67] proposed to use a probability coincidence matrix to store probabilities of Bayesian-network dependencies between different pairs of problem variables in prior hBOA runs and to bias the model building in hBOA on future problem instances of similar type using the matrix.

6.5 Fitness Evaluation Relaxation

To reduce the number of objective (fitness) function evaluations, a model of the fitness function can be built [129, 167, 168]. If an advanced EDA is used that contains a complex probabilistic model, the model itself can be mined to provide a set of statistics that can be estimated for an accurate, efficient computational model of the objective function. The model is then used to replace some of the evaluations, possibly most of them. It was shown that the use of adequate models of the objective function can yield multiplicative speedups of several tens [129, 167, 168].

6.6 Incremental and Sporadic Model Building

With sporadic model-building, the structure of the probabilistic model is built once every few generations and the probabilities are updated every generation [134]. With incremental model building, the model is built incrementally starting from the structure discovered in the previous iteration [39]. This allows for models that are ideally both more accurate and quicker to learn.

7 Starting Points for Obtaining Additional Information

This section provides pointers for obtaining additional information about EDAs.

7.1 Introductory Books and Tutorials

Numerous books and other publications exist that provide introduction to estimation of distribution algorithms and additional starting points. The following list of references includes some of them: [53, 66, 84, 97, 117, 118, 124, 130].

7.2 Software

The following list includes some of the popular EDA implementations available online. These implementations should provide a good starting point for the interested reader. Entries in the list

are ordered alphabetically. Note that the list is *not* exhaustive.

- Adapted maximum-likelihood Gaussian model iterated density estimation evolutionary algorithm (AMaLGaM) [18]:
  `http://homepages.cwi.nl/~bosman/source_code.php`
- Bayesian optimization algorithm (BOA) [123]; BOA with decision graphs [125]; dependency-tree EDA [8]:
  `http://medal.cs.umsl.edu/`
- Demos of aggregation pheromone system (APS) [191] and histogram-based EDAs for permutation-based problems (EHBSA) [193]:
  `http://www.hannan-u.ac.jp/~tsutsui/research-e.html`
- Distribution estimation using Markov random fields (DEUM) [177, 176]:
  `http://sidshakya.com/Downloads/Main.html`
- Extended compact genetic algorithm [60], ξ-ary ECGA, BOA [123], BOA with decision trees/graphs [125], and others:
  `http://illigal.org/`
- Mixed BOA (mBOA) [110], adaptive mBOA (amBOA) [114]:
  `http://jiri.ocenasek.com/`
- Probabilistic incremental program evolution (PIPE) [154]:
  `ftp://ftp.idsia.ch/pub/rafal/`
- Real-coded BOA (rBOA) [2], multiobjective rBOA [1]:
  `http://www.evolution.re.kr/`
- Regularity model based multiobjective EDA (RM-MEDA) [208]; hybrid of differential evolution and EDA [87]; model-based multiobjective evolutionary algorithm (MMEA) [206], and others:
  `http://cswww.essex.ac.uk/staff/qzhang/mypublication.htm`

7.3 Journals

The following journals are key venues for papers on EDAs and evolutionary computation, although papers on EDAs can be found in many other journals focusing on optimization, artificial intelligence, machine learning, and applications.

- *Evolutionary Computation* (MIT Press):
  `http://www.mitpressjournals.org/loi/evco`
- *Evolutionary Intelligence* (Springer):
  `http://www.springer.com/engineering/journal/12065`
- *Genetic Programming and Evolvable Machines* (Springer):
  `http://www.springer.com/computer/ai/journal/10710`
- *IEEE Transactions on Evolutionary Computation* (IEEE Press):
  `http://ieeexplore.ieee.org/servlet/opac?punumber=4235`

- *Natural Computing* (Springer):
  `http://www.springer.com/computer/theoretical+computer+science/journal/11047`
- *Swarm and Evolutionary Computation* (Elsevier):
  `http://www.journals.elsevier.com/swarm-and-evolutionary-computation/`

7.4 Conferences

The following conferences provide the most important venues for publishing papers on EDAs and evolutionary computation, although similarly as for journals, papers on EDAs are often published in other venues.

- *ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO)*
- *European Workshops on Applications of Evolutionary Computation (EvoWorkshops)*
- *IEEE Congress on Evolutionary Computation (CEC)*
- *Main European Events on Evolutionary Computation (EvoStar)*
- *Parallel Problem Solving in Nature (PPSN)*
- *Simulated Evolution and Learning (SEAL)*

8 Summary and Conclusions

EDAs are a class of stochastic optimization algorithms that have been gaining popularity due to their ability to solve a broad array of complex problems with excellent performance and scalability. Moreover, while many of these algorithms have been shown to perform well with little or no problem-specific information, such information can be used advantageously if available.

EDAs have their roots in the fields of evolutionary computation and machine learning. From evolutionary computation EDAs borrow the idea of using a population of solutions that evolves through iterations of selection and variation. From machine learning EDAs borrow the idea of learning models from data, and they use the resulting models to guide the search for better solutions. This approach is powerful especially because it allows the search algorithm to adapt to the problem being solved, giving EDAs the possibility of being an effective *black-box* search algorithm. Since most real world problems have some sort of inherent structure (as opposed to being completely random), there is a hope that EDAs can learn such a structure, or at least parts of it, and put that knowledge to good use in searching for optima.

Another key characteristic of EDAs, and one that sets them apart from other metaheuristics, lies in the fact that the sequence of probabilistic models learned along a particular run (or a sequence or runs) yields important information that can be exploited for other means. For example, such information can be used for building surrogate models of the objective function leading to significant performance speedups, for designing effective neighborhoods for local search when conventional neighborhoods fail, and even for learning about characteristics of an entire class of problems that can in turn be used to solve other instances of the same problem class.

This chapter gave an introduction and reviewed both the history and the state of the art in EDA research. The basic concepts of these algorithms were presented and a taxonomy was outlined from the views based on the model decomposition and the type of local distributions. The most popular

EDAs proposed in the literature were then surveyed according to the most common representations for candidate solutions. Finally, the major theoretical research areas and efficiency enhancement techniques for EDAs were highlighted. This chapter should be valuable both for those who want to grasp the basic ideas of EDAs as well as for those who want to have a coherent view of EDA research.

Acknowledgments

This project was sponsored by the National Science Foundation under grants ECS-0547013 and IIS-1115352, by the University of Missouri in St. Louis through the High Performance Computing Collaboratory sponsored by Information Technology Services, and by the University of Missouri Bioinformatics Consortium (UMBC). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Ahn, C.W., Ramakrishna, R.S.: Multiobjective real-coded Bayesian optimization algorithm revisited: Diversity preservation. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 593–600

[2] Ahn, C.W., Ramakrishna, R.S., Goldberg, D.E.: Real-coded Bayesian optimization algorithm: Bringing the strength of BOA into the continuous world. Genetic and Evolutionary Computation Conference (GECCO-2004) (2004) 840–851

[3] Armañanzas, R., Saeys, Y., Inza, I., García-Torres, M., Bielza, C., de Peer, Y.V., Larrañaga, P.: Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms. IEEE/ACM Trans. Comput. Biology Bioinform. **8**(3) (2011) 760–774

[4] Arst, R., Minsker, B.S., Goldberg, D.E.: Comparing advanced genetic algorithms and simple genetic algorithms for groundwater management. Proceedings of the American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) 2002 Water Resources Planning & Management Conference (2002) Roanoke, VA

[5] Bacardit, J., Stout, M., Hirst, J.D., Sastry, K., Llorà, X., Krasnogor, N.: Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 346–353

[6] Baluja, S.: Incorporating a priori knowledge in probabilistic-model based optimization. In Cantú-Paz, E., Pelikan, M., Sastry, K., eds.: Scalable optimization via probabilistic modeling: From algorithms to applications. Springer (2006) 205–219

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

[8] Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Proceedings of the International Conference on Machine Learning (1997) 30–38

[9] Baluja, S., Davies, S.: Fast probabilistic modeling for combinatorial optimization. Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98) (1998) 469–476

[10] Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6**(2) (1994) 154–160

[11] Belda, I., Madurga, S., Llorà, X., Martinell, M., Tarragó, Piqueras, M.G., Nicolás, E., Giralt, E.: ENPDA: An evolutionary structure-based de novo peptide design algorithm. Journal of Computer Aided Molecular Design (2005)

[12] Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., Boeres, C.: Inexact graph matching using learning and simulation of Bayesian networks. In: Workshop on Bayesian and Causal Networks: From Inference to Data Mining at ECAI-2000. (2000)

[13] Bosman, P.A.N., de Jong, E.D.: Learning probabilistic tree grammars for genetic programming. Parallel Problem Solving from Nature (2004) 192–201

[14] Bosman, P.A.N., Grahl, J., Thierens, D.: Enhancing the performance of maximum-likelihood Gaussian EDAs using anticipated mean shift. Parallel Problem Solving from Nature (2008) 133–143

[15] Bosman, P.A.N.: On empirical memory design, faster selection of Bayesian factorizations and parameter-free Gaussian EDAs. Genetic and Evolutionary Computation Conference (GECCO-2009) (2009) 389–396

[16] Bosman, P.A.N.: On gradients and hybrid evolutionary algorithms for real-valued multi-objective optimization. IEEE Transactions on Evolutionary Computation **16**(1) (2012) 51–69

[17] Bosman, P.A.N., Grahl, J., Thierens, D.: AMaLGaM IDEAs in noiseless black-box optimization benchmarking. In: Black Box Optimization Benchmarking BBOB Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2009). (2009) 2247–2254

[18] Bosman, P.A.N., Grahl, J., Thierens, D.: AMaLGaM IDEAs in noisy black-box optimization benchmarking. In: Black Box Optimization Benchmarking BBOB Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2009). (2009) 2351–2358

[19] Bosman, P.A.N., Grahl, J., Rothlauf, F.: SDR: A better trigger for adaptive variance scaling in normal EDAs. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 492–499

[20] Bosman, P.A.N., Thierens, D.: Linkage information processing in distribution estimation algorithms. Genetic and Evolutionary Computation Conference (GECCO-99) **I** (1999) 60–67

[21] Bosman, P.A.N., Thierens, D.: Continuous iterated density estimation evolutionary algorithms within the IDEA framework. Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000) (2000) 197–200

[22] Bosman, P.A.N., Thierens, D.: Mixed IDEAs. Utrecht University Technical Report UU-CS-2000-45, Utrecht University, Utrecht, Netherlands (2000)

[23] Bosman, P.A.N., Thierens, D.: Crossing the road to efficient IDEAs for permutation problems. Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 219–226

[24] Bosman, P.A.N., Thierens, D.: Exploiting gradient information in continuous iterated density estimation evolutionary algorithms. Proc. of the Belgium-Netherlands Conf. on Artificial Intelligence (BNAIC-2001) (2001) 69–76

[25] Bosman, P.A.N., Thierens, D.: New IDEAs and more ICE by learning and using unconditional permutation factorizations. In: Late-Breaking Papers of the Genetic and Evolutionary Computation Conf. (GECCO 2001). (2001) 13–23

[26] Cantú-Paz, E.: Comparing selection methods of evolutionary algorithms using the distribution of fitness. Technical Report UCRL-JC-138582, University of California Lawrence Livermore National Laboratory (2000)

[27] Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer, Boston, MA (2000)

[28] Cantú-Paz, E.: Supervised and unsupervised discretization methods for evolutionary algorithms. Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 213–216

[29] Černý, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications **45** (1985) 41–51 10.1007/BF00940812.

[30] Chen, Y.P., Chen, C.H.: Enabling the extended compact genetic algorithm for real-parameter optimization by using adaptive discretization. Evolutionary Computation **18**(2) (2010) 199–228

[31] Chen, Y., Yu, T.L., Sastry, K., Goldberg, D.E.: A survey of genetic linkage learning techniques. IlliGAL Report No. 2007014, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2007)

[32] Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory **14** (1968) 462–467

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

[34] De Bonet, J.S., Isbell, C.L., Viola, P.: MIMIC: Finding optima by estimating probability densities. Advances in Neural Information Processing Systems (NIPS-97) **9** (1997) 424–431

[35] Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial Life **5**(2) (1999) 137–172

[36] Ducheyne, E., De Baets, B., De Wulf, R.: Probabilistic models for linkage learning in forest management. In Jin, Y., ed.: Knowledge incorporation in evolutionary computation. Springer (2004) 177–194

[37] Echegoyen, C., Mendiburu, A., Santana, R., Lozano, J.A.: Toward understanding EDAs based on Bayesian networks through a quantitative analysis. IEEE Transactions on Evolutionary Computation **99** (2011) 1–17

[38] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2010)

[39] Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. Second Symposium on Artificial Intelligence (CIMAF-99) (1999) 332–339

[40] Gallagher, M., Frean, M.: Population-based continuous optimization, probabilistic modelling and mean shift. Evolutionary Computation **13**(1) (2005) 29–42

[41] Gallagher, M., Frean, M., Downs, T.: Real-valued evolutionary optimization using a flexible probability density estimator. Genetic and Evolutionary Computation Conference (GECCO-99) **1** (13-17 July 1999) 840–846

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

[43] Goldberg, D.E.: Real-coded genetic algorithms, virtual alphabets, and blocking. Complex Systems **5**(2) (1991) 139–167

[44] Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms **1** (1991) 69–93

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

[46] Goldberg, D.E., Sastry, K., Latoza, T.: On the supply of building blocks. Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 336–342

[47] Goldberg, D.E., Voessner, S.: Optimizing global-local search hybrids. Genetic and Evolutionary Computation Conference (GECCO-99) (1999) 220–228

[48] Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA (1989)

[49] Goldberg, D.E.: Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. Genetic and Evolutionary Computation Conference (GECCO-99) (1999) 212–219

[50] Goldberg, D.E.: The design of innovation: Lessons from and for competent genetic algorithms. Kluwer (2002)

[51] Goldberg, D.E., Sastry, K., Llorà, X.: Toward routine billion-variable optimization using genetic algorithms. Complexity **12**(3) (2007) 27–29

[52] Gonzalez, C., Lozano, J., Larrañaga, P.: Analyzing the PBIL algorithm by means of discrete dynamical systems. Complex Systems **4**(12) (2001) 465–479

[53] Grahl, J., Minner, S., Bosman, P.: Learning structure illuminates black boxes: An introduction into estimation of distribution algorithms. In Michalewicz, Z., Siarry, P., eds.: Advances in Metaheuristics for Hard Optimization. Springer (2008) 365–396

[54] Grosan, C., Abraham, A., Ishibuchi, H., eds.: Hybrid Evolutionary Algorithms. Studies in Computational Intelligence. Springer (2007)

[55] Gruau, F.: On using syntactic constraints with genetic programming. In Angeline, P.J., Kinnear, Jr., K.E., eds.: Advances in Genetic Programming 2. MIT Press (1996) 377–394

[56] Handley, S.: On the use of a directed acyclic graph to represent a population of computer programs. International Conference on Evolutionary Computation (ICEC-94) (1994) 154–159

[57] Hansen, N., Ostermeier, A., Gawelczyk, A.: On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. International Conference on Genetic Algorithms (ICGA-95) (1995) 57–64

[58] Harik, G., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation **7**(3) (1999) 231–253

[59] Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Transactions on Evolutionary Computation **3**(4) (1999) 287–297

[60] Harik, G.: Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)

[61] Harik, G.R., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. International Conference on Evolutionary Computation (ICEC-97) (1997) 7–12

[62] Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. International Conference on Evolutionary Computation (ICEC-98) (1998) 523–528

[63] Hasegawa, Y., Iba, H.: Estimation of distribution algorithm based on probabilistic grammar with latent annotations. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. (sept. 2007) 1043 –1050

[64] Hauschild, M.W., Pelikan, M.: Enhancing efficiency of hierarchical BOA via distance-based model restrictions. Parallel Problem Solving from Nature (2008) 417–427

[65] Hauschild, M.W., Pelikan, M.: Intelligent bias of network structures in the hierarchical BOA. Genetic and Evolutionary Computation Conference (GECCO-2009) (2009) 413–420

[66] Hauschild, M.W., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation **1**(3) (2011) 111–128

[67] Hauschild, M.W., Pelikan, M., Sastry, K., Goldberg, D.E.: Using previous models to bias structural learning in the hierarchical BOA. Evolutionary Computation (2011) In Press.

[68] Hauschild, M.W., Pelikan, M., Sastry, K., Lima, C.F.: Analyzing probabilistic models in hierarchical BOA. IEEE Transactions on Evolutionary Computation **13**(6) (2009) 1199–1217

[69] Hayes, M.S., Minsker, B.S.: Evaluation of advanced genetic algorithms applied to groundwater remediation design. Proceedings of the American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) World Water & Environmental Resources Congress 2005 & Related Symposia (2005) Anchorage, AK

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

[71] Heckerman, D., Geiger, D., Chickering, D.M.: Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning **20**(3) (1995) 197–243

[72] Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. Complex Systems **1** (1987) 495–502

[73] Höhfeld, M., Rudolph, G.: Towards a theory of population-based incremental learning. International Conference on Evolutionary Computation (ICEC-97) (1997) 1–6

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

[75] Iclanzan, D., Dumitrescu, D.: Overcoming hierarchical difficulty by hill-climbing the building block structure. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 1256–1263

[76] Juels, A., Baluja, S., Sinclair, A.: The equilibrium genetic algorithm and the role of crossover. Unpublished manuscript (1993)

[77] Khan, N., Goldberg, D.E., Pelikan, M.: Multi-objective Bayesian optimization algorithm. IlliGAL Report No. 2002009, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2002)

[78] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220** (1983) 671–680

[79] Kollat, J.B., Reed, P.M., Kasprzyk, J.R.: A new epsilon-dominance hierarchical Bayesian optimization algorithm for large multi-objective monitoring network design problems. Advances in Water Resources **31**(5) (2008) 828–845

[80] Koza, J.R.: Genetic programming: On the programming of computers by means of natural selection. The MIT Press, Cambridge, MA (1992)

[81] Kullback, S., Leibler, R.A.: On information and sufficiency. Annals of Math. Stats. **22** (1951) 79–86

[82] Kvasnicka, V., Pelikan, M., Pospichal, J.: Hill climbing with learning (An abstraction of genetic algorithm). Neural Network World **6** (1996) 773–796

[83] Larrañaga, P., Etxeberria, R., Lozano, J.A., Pena, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000) (2000) 201–204

[84] Larrañaga, P., Lozano, J.A., eds.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Boston, MA (2002)

[85] Laumanns, M., Ocenasek, J.: Bayesian optimization algorithms for multi-objective optimization. Parallel Problem Solving from Nature (2002) 298–307

[86] Lawler, E.L.: The quadratic assignment problem. Management Science **9**(4) (1963) 586–599

[87] Li, H., Zhang, Q.: A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages. Parallel Problem Solving from Nature (2006) 583–592

[88] Li, J., Aickelin, U.: A Bayesian optimization algorithm for the nurse scheduling problem. IEEE Congress on Evolutionary Computation (CEC-2003) (2003) 2149–2156

[89] Lima, C., Lobo, F., Pelikan, M., Goldberg, D.E.: Model accuracy in the Bayesian optimization algorithm. Soft Computing - A Fusion of Foundations, Methodologies and Applications **15** (2011) 1351–1371

[90] Lima, C.F., Pelikan, M., Lobo, F.G., Goldberg, D.E.: Loopy substructural local search for the Bayesian optimization algorithm. In Stützle, T., Birattari, M., Hoos, H.H., eds.: Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, Springer (2009) 61–75

[91] Lima, C.F., Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E., Lobo, F.G.: Substructural neighborhoods for local search in the Bayesian optimization algorithm. Parallel Problem Solving from Nature (2006) 232–241

[92] Lipinski, P.: ECGA vs. BOA in discovering stock market trading experts. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 531–538

[93] Looks, M.: Competent Program Evolution. PhD thesis, Washington University, St. Louis, MO (2006)

[94] Looks, M.: Levels of abstraction in modeling and sampling: The feature-based Bayesian optimization algorithm. Genetic and Evolutionary Computation Conference (GECCO-2006) (2006) 429–430

[95] Looks, M.: Scalable estimation-of-distribution program evolution. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 539–546

[96] Looks, M., Goertzel, B., Pennachin, C.: Learning computer programs with the Bayesian optimization algorithm. Genetic and Evolutionary Computation Conference (GECCO-2005) (2005) 747–748

[97] Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E., eds.: Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer (2006)

[98] Marascuilo, L.A., McSweeney, M.: Nonparametric and distribution-free methods for the social sciences. Brooks/Cole Publishing Company, CA (1977)

[99] McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: A survey. Genetic Programming and Evolvable Machines **11**(3-4) (2010) 365–396

[100] Mendiburu, A., Santana, R., Lozano, J.A.: Introducing belief propagation in estimation of distribution algorithms: A parallel approach. Tech. Rep. EHU-KAT-IK-11-07, Department of Computer Science and Artificial Intelligence, University of the Basque Country (2007)

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

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

[103] Mühlenbein, H.: Convergence of estimation of distribution algorithms for finite samples. Technical report, Fraunhofer Institut Autonomous intelligent Systems, Sankt Augustin, Germany (2008)

[104] Mühlenbein, H., Mahnig, T.: Convergence theory and applications of the factorized distribution algorithm. Journal of Computing and Information Technology **7**(1) (1998) 19–32

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

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

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

[108] Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In Eiben, A., Bäck, T., Shoenauer, M., Schwefel, H., eds.: Parallel Problem Solving from Nature, Berlin, Springer Verlag (1996) 178–187

[109] Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. Parallel Problem Solving from Nature (1996) 178–187

[110] Ocenasek, J., Schwarz, J.: Estimation of distribution algorithm for mixed continuous-discrete optimization problems. In: 2nd Euro-International Symposium on Computational Intelligence, Kosice, Slovakia, IOS Press (2002) 227–232

[111] Ocenasek, J.: Parallel Estimation of Distribution Algorithms. PhD thesis, Faculty of Information Technology, Brno University of Technology, Brno (2002)

[112] Ocenasek, J.: Parallel Estimation of Distribution Algorithms: Principles and Enhancements. Lambert Academic Publishing (2010)

[113] Ocenasek, J., Cantú-Paz, E., Pelikan, M., Schwarz, J.: Design of parallel estimation of distribution algorithms. In Pelikan, M., Sastry, K., Cantú-Paz, E., eds.: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Springer (2006)

[114] Ocenasek, J., Kern, S., Hansen, N., Koumoutsakos, P.: A mixed Bayesian optimization algorithm with variance adaptation. Parallel Problem Solving from Nature (2004) 352–361

[115] Ocenasek, J., Schwarz, J.: The parallel Bayesian optimization algorithm. In: Proc. of the European Symposium on Computational Inteligence, Physica-Verlag (2000) 61–67

[116] Ochoa, A., Höns, R., Soto, M., Mühlenbein, H.: A maximum entropy approach to sampling in eda: The single connected case. In: Progress in Pattern Recognition, Speech and Image Analysis. Volume 2905 of LNCS., Springer (2003) 683–690

[117] Pelikan, M.: Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms. Springer (2005)

[118] Pelikan, M.: Probabilistic model-building genetic algorithms. In: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation. GECCO '11, New York, NY, USA, ACM (2011) 913–940

[119] Pelikan, M., Goldberg, D.E.: Genetic algorithms, clustering, and the breaking of symmetry. Parallel Problem Solving from Nature (2000) 385–394

[120] Pelikan, M., Goldberg, D.E.: Escaping hierarchical traps with competent genetic algorithms. Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 511–518

[121] Pelikan, M., Goldberg, D.E.: Hierarchical BOA solves Ising spin glasses and maxsat. Genetic and Evolutionary Computation Conference (GECCO-2003) **II** (2003) 1275–1286

[122] Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: Linkage problem, distribution estimation, and Bayesian networks. IlliGAL Report No. 98013, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1998)

[123] Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. Genetic and Evolutionary Computation Conference (GECCO-99) **I** (1999) 525–532

[124] Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications **21**(1) (2002) 5–20

[125] Pelikan, M., Goldberg, D.E., Sastry, K.: Bayesian optimization algorithm, decision graphs, and Occam's razor. Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 519–526

[126] Pelikan, M., Goldberg, D.E., Tsutsui, S.: Combining the strengths of the Bayesian optimization algorithm and adaptive evolution strategies. Genetic and Evolutionary Computation Conference (GECCO-2002) (2002) 512–519

[127] Pelikan, M., Hartmann, A.K.: Searching for ground states of Ising spin glasses with hierarchical BOA and cluster exact approximation. In Cantú-Paz, E., Pelikan, M., Sastry, K., eds.: Scalable optimization via probabilistic modeling: From algorithms to applications. Springer (2006)

[128] Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. Advances in Soft Computing—Engineering Design and Manufacturing (1999) 521–535

[129] Pelikan, M., Sastry, K.: Fitness inheritance in the Bayesian optimization algorithm. Genetic and Evolutionary Computation Conference (GECCO-2004) **2** (2004) 48–59

[130] Pelikan, M., Sastry, K., Cantú-Paz, E., eds.: Scalable optimization via probabilistic modeling: From algorithms to applications. Springer-Verlag (2006)

[131] Pelikan, M., Sastry, K., Goldberg, D.E.: Scalability of the Bayesian optimization algorithm. International Journal of Approximate Reasoning **31**(3) (2002) 221–258

[132] Pelikan, M., Sastry, K., Goldberg, D.E.: Multiobjective hBOA, clustering, and scalability. Genetic and Evolutionary Computation Conference (GECCO-2005) (2005) 663–670

[133] Pelikan, M., Sastry, K., Goldberg, D.E.: iBOA: The incremental Bayesian optimization algorithm. Genetic and Evolutionary Computation Conference (GECCO-2008) (2008) 455–462

[134] Pelikan, M., Sastry, K., Goldberg, D.E.: Sporadic model building for efficiency enhancement of the hierarchical BOA. Genetic Programming and Evolvable Machines **9**(1) (2008) 53–84

[135] Pelikan, M., Sastry, K., Tsutsui, S.: Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert. Information Sciences **156**(3-4) (2003) 147–171

[136] Pelikan, M., Tsutsui, S., Kalapala, R.: Dependency trees, permutations, and quadratic assignment problem. MEDAL Report No. 2007003, Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri–St. Louis, St. Louis, MO (2007)

[137] Posík, P., Vanícek, S.: Parameter-less local optimizer with linkage identification for deterministic order-k decomposable problems. Genetic and Evolutionary Computation Conference (GECCO-2011) (2011) 577–584

[138] Prim, R.: Shortest connection networks and some generalizations. Bell Systems Technical Journal **36** (1957) 1389–1401

[139] Radetic, E., Pelikan, M.: Spurious dependencies and EDA scalability. Genetic and Evolutionary Computation Conference (GECCO-2010) (2010) 303–310

[140] Radetic, E., Pelikan, M., Goldberg, D.E.: Effects of a deterministic hill climber on hBOA. Genetic and Evolutionary Computation Conference (GECCO-2009) (2009) 437–444

[141] Ratle, A., Sebag, M.: Avoiding the bloat with probabilistic grammar-guided genetic programming. The 5th International Conference, Evolution Artificielle (2001) 255–266

[142] Ratle, A., Sebag, M.: Avoiding the bloat with stochastic grammar-based genetic programming. CoRR **abs/cs/060** (2006) 255–266

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

[144] Rechenberg, I.: Evolutionsstrategie '94. Frommann-Holzboog Verlag, Stuttgart (1994)

[145] Reed, P.M., Shah, R., Kollat, J.B.: Assessing the value of environmental observations in a changing world: Nonstationarity, complexity, and hierarchical dependencies. In Swayne, D.A., Yang, W., Voinov, A.A., Rizzoli, A., Filatova, T., eds.: Fifth Biennial Meeting, International Congress on Environmental Modelling and Software Modelling for Environments Sake, International Environmental Modelling and Software Society (iEMSs) (2010) S.12.04

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

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

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

[149] Robles, V., P. de Miguel, P.L.n.: Solving the traveling salesman problem with edas. In Larrañaga, P., Lozano, J.A., eds.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers (2002) 227–238

[150] Rothlauf, F., Goldberg, D.E., Heinzl, A.: Bad codings and the utility of well-designed genetic algorithms. IlliGAL Report No. 200007, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2000)

[151] Rudlof, S., Köppen, M.: Stochastic hill climbing with learning by vectors of normal distributions. In: First On-line Workshop on Soft Computing, Nagoya, Japan (1996)

[152] Salhi, A., Rodríguez, J.A.V., Zhang, Q.: An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 570–576

[153] Salustowicz, R.P., Schmidhuber, J.: Probabilistic incremental program evolution. Evolutionary Computation **5**(2) (1997) 123–141

[154] Salustowicz, R.P., Schmidhuber, J.: Probabilistic incremental program evolution: Stochastic search through program space. Proceedings of the European Conference of Machine Learning (ECML-97) **1224** (1997) 213–220

[155] Salustowicz, R., Schmidhuber, J.: H-PIPE: Facilitating hierarchical program evolution through skip nodes. Technical Report IDSIA-08-98, Instituto Dalle Molle di Studi sull' Intelligenza Artificiale (IDSIA), Lugano, Switzerland (1998)

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

[157] Santana, R., Larraaga, P., Lozano, J.A.: Protein folding in simplified models with estimation of distribution algorithms. IEEE Transactions on Evolutionary Computation **12**(4) (2008) 418–438

[158] Santarelli, S., Yu, T.L., Goldberg, D.E., Altshuler, E.E., O'Donnell, T., Southall, H., Mailloux, R.: Military antenna design using simple and competent genetic algorithms. Mathematical and Computer Modelling **43**(9-10) (2006) 990–1022

[159] Sastry, K., Goldberg, D.E.: Designing competent mutation operators via probabilistic model building of neighborhoods. Genetic and Evolutionary Computation Conference (GECCO-2004) (2004) 114–125 Also IlliGAL Report No. 2004006.

[160] Sastry, K., Goldberg, D.E.: Let's get ready to rumble: Crossover versus mutation head to head. Genetic and Evolutionary Computation Conference (GECCO-2004) (2004) 126–137

[161] Sastry, K., Goldberg, D.E.: Let's get ready to rumble redux: Crossover versus mutation head to head on exponentially scaled problems. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 114–125 Also IlliGAL Report No. 2004006.

[162] Sastry, K.: Efficient atomic cluster optimization using a hybrid extended compact genetic algorithm with seeded population. IlliGAL Report No. 2001018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2001)

[163] Sastry, K., Goldberg, D.E.: On extended compact genetic algorithm. IlliGAL Report No. 2000026, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2000)

[164] Sastry, K., Goldberg, D.E.: Probabilistic model building and competent genetic programming. In Riolo, R.L., Worzel, B., eds.: Genetic Programming Theory and Practise. Kluwer (2003) 205–220

[165] Sastry, K., Goldberg, D.E., Johnson, D.D.: Scalability of a hybrid extended compact genetic algorithm for ground state optimization of clusters. Materials and Manufacturing Processes **22**(5) (2007) 570–576

[166] Sastry, K., Goldberg, D.E., Llorà, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 577–584

[167] Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. IEEE Congress on Evolutionary Computation (CEC-2004) (2004) 720–727

[168] Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency enhancement of estimation of distribution algorithms. In Pelikan, M., Sastry, K., Cantú-Paz, E., eds.: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Springer (2006) ?–?

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

[170] Schwarz, J., Ocenasek, J.: Experimental study: Hypergraph partitioning based on the simple and advanced algorithms BMDA and BOA. Mendel '99: International Conference on Soft Computing (1999) 124–130

[171] Schwarz, J., Ocenasek, J.: A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning. Proceedings of the Fourth Joint Conference on Knowledge-Based Software Engineering (2000) 51–58

[172] Schwefel, H.P.: Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie. Volume 26 of Interdisciplinary Systems Research. Birkhäuser, Basle, Switzerland (1977)

[173] Sebag, M., Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces. Parallel Problem Solving from Nature (1998) 418–427

[174] Servet, I., Trave-Massuyes, L., Stern, D.: Telephone network traffic overloading diagnosis and evolutionary computation techniques. Proceedings of the European Conference on Artificial Evolution (AE-97) (1997) 137–144

[175] Shah, R., Reed, P.: Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems. European Journal of Operations Research **211**(3) (2011) 466–479

[176] Shakya, S., Brownlee, A.E.I., McCall, J.A.W., Fournier, F.A., Owusu, G.: A fully multivariate DEUM algorithm. IEEE Congress on Evolutionary Computation (CEC-2009) (2009) 479–486

[177] Shakya, S.K.: DEUM: A Framework for an Estimation of Distribution Algorithm based on Markov Random Fields. PhD thesis, Robert Gordon University, Aberdeen, UK (2006)

[178] Shakya, S.K., McCall, J.A., Brown, D.F.: Solving the Ising spin glass problem using a bivariate EDA based on Markov random fields. IEEE Congress on Evolutionary Computation (CEC-2006) (2006) 908– 915

[179] Shan, Y., McKay, R., Abbass, H.A., Essam, D.: Program evolution with explicit learning: A new framework for program automatic synthesis. IEEE Congress on Evolutionary Computation (CEC-2003) (2003) 1639–1646

[180] Shan, Y.: Program Distribution Estimation with Grammar Models. PhD thesis, Wuhan Cehui Technical University, China (2005)

[181] Shan, Y., McKay, R.I., Baxter, R.: Grammar model-based program evolution. IEEE Congress on Evolutionary Computation (CEC-2004) (2004) 478–485

[182] Shapiro, J.L.: Drift and scaling in estimation of distribution algorithms. Evolutionary Computation **13** (January 2005) 99–123

[183] Sinha, A., Goldberg, D.E.: A survey of hybrid genetic and evolutionary algorithms. IlliGAL Report No. 2003004, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2003)

[184] Soto, M., Ochoa, A.: A factorized distribution algorithm based on polytrees. IEEE Congress on Evolutionary Computation (CEC-2000) (2000) 232–237

[185] Sun, J., Zhang, Q., Li, J., Yao, X.: A hybrid estimation of distribution algorithm for cdma cellular system design. International Journal of Computational Intelligence and Applications **7**(2) (2007) 187–200

[186] Tanev, I.: Incorporating learning probabilistic context-sensitive grammar in genetic programming for efficient evolution and adaptation of snakebot. Proceedings of the 8th European Conference on Genetic Programming (EuroGP-2005) (2005) 155–166

[187] Thierens, D.: Analysis and design of genetic algorithms. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1995)

[188] Thierens, D.: Scalability problems of simple genetic algorithms. Evolutionary Computation **7**(4) (1999) 331–352

[189] Thierens, D., Bosman, P.A.N.: Multi-objective mixture-based iterated density estimation evolutionary algorithms. Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 663–670

[190] Tsutsui, S., Goldberg, D.E., Pelikan, M.: Solving sequence problems by building and sampling edge histograms. IlliGAL Report No. 2002024, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL (2002)

[191] Tsutsui, S., Pelikan, M., Ghosh, A.: Performance of aggregation pheromone system on unimodal and multimodal problems. IEEE Congress on Evolutionary Computation (CEC-2005) (2005) 880–887

[192] Tsutsui, S., Pelikan, M., Goldberg, D.E.: Probabilistic model-building genetic algorithms using histogram models in continuous domain. Journal of the Information Processing Society of Japan **43** (2002)

[193] Tsutsui, S., Pelikan, M., Goldberg, D.E.: Node histogram vs. edge histogram: A comparison of pmbgas in permutation domains. MEDAL Report No. 2006009, Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri–St. Louis, St. Louis, MO (2006)

[194] Tsutsui, S., Pelikan, M., Goldberg, D.E.: Evolutionary algorithm using marginal histogram models in continuous domain. Knowledge-Based Intelligent Information Engineering Systems & Allied Thechnologies (KES-2001) (2001) 112–121

[195] Tsutsui, S., Pelikan, M., Goldberg, D.E.: Evolutionary algorithm using marginal histogram models in continuous domain. Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 230–233

[196] Watson, R.A., Hornby, G.S., Pollack, J.B.: Modeling building-block interdependency. Parallel Problem Solving from Nature (1998) 97–106

[197] Whigham, P.: Grammatically-based genetic programming. Proceedings of the Workshop on Genetic Programming: From Theory to Real- World Applications (1995) 33–41

[198] Wong, M.L., Leung, K.S.: Genetic logic programming and applications. IEEE Expert **10**(5) (1995) 68–76

[199] Yanai, K., Iba, H.: Estimation of distribution programming based on Bayesian network. IEEE Congress on Evolutionary Computation (CEC-2003) (2003) 1618–1625

[200] Yu, T.L.: A matrix approach for finding extrema: Problems with modularity, hierarchy, and overlap. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL (2006)

[201] Yu, T.L., Goldberg, D.E., Chen, Y.P.: A genetic algorithm design inspired by organizational theory: A pilot study of a dependency structure matrix driven genetic algorithm. IlliGAL Report No. 2003007, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2003)

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

[203] Yu, T.L., Sastry, K., Goldberg, D.E., Pelikan, M.: Population sizing for entropy-based model building in estimation of distribution algorithms. Genetic and Evolutionary Computation Conference (GECCO-2007) (2007) 601–608

[204] Zhang, B.T., Shin, S.Y.: Bayesian evolutionary optimization using Helmholtz machines. Parallel Problem Solving from Nature (2000) 827–836

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

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

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

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