Adaptive Estimation of Distribution Algorithms

Roberto Santana[1], Pedro Larrañaga[1], and José A. Lozano[1]

Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizabal 1, 20080 Donostia - San Sebastian, Spain
rsantana@si.ehu.es, pedro.larranaga@ehu.es, ja.lozano@ehu.es

Summary. Estimation of distribution algorithms (EDAs) are evolutionary methods that use probabilistic models instead of genetic operators to lead the search. Most of current proposals on EDAs do not incorporate adaptive techniques. Usually, the class of probabilistic model employed as well as the learning and sampling methods are static. In this paper, we present a general framework for introducing adaptation in EDAs. This framework allows the possibility of changing the class of probabilistic models during the evolution. We present a number of measures, and techniques that can be used to evaluate the effect of the EDA components in order to design adaptive EDAs. As a case of study we present an adaptive EDA that combines different classes of probabilistic models and sampling methods. The algorithm is evaluated in the solution of the satisfiability problem.

Keywords: Estimation of distribution algorithm, adaptive probabilistic model, SAT.

1 Introduction

Estimation of distribution algorithms (EDAs) [9] are evolutionary methods that use probabilistic models to represent relevant information about the search space. The idea is to capture, in the form of probabilistic dependencies between the variables, information about promising areas of the search space that can be used to improve the search for better solutions. Machine learning techniques are used to learn the probabilistic models and sample new solutions from them. EDAs have shown to solve problems where genetic algorithms exhibit a poor performance [9, 12].

A characteristic feature of EDAs is the type of probabilistic model used. Different models come associated with different capacities of representation and the computational complexity of the algorithms used to learn and sample from them also changes accordingly. Although probabilistic models provide EDAs with an important degree of flexibility, usually the class of the models is fixed at the beginning of the evolution and will not change during the search process. This fact may compromise the flexibility of the algorithm. More efficient EDAs are expected to exhibit a wider amount of adaptation with more flexible frameworks for probabilistic modeling.

In this chapter, we present our initial results on the conception of adaptive EDAs. We identify a number of ways in which adaptation can be added to EDAs and focus on the use of adaptive probabilistic models and sample algorithms. Our findings lead to the introduction of an EDA that uses a combination of probabilistic models and which is evaluated in the optimization of a number of instances of the satisfiability problem.

The chapter is organized as follows. In the next section, EDAs are presented and some of their main characteristics are discussed. In Section 3, we accomplish a brief review of previous work on the design of adaptive genetic algorithms. Section 4 introduces a general framework for the analysis and design of adaptive EDAs. In Section 5 we focus on the analysis of feasible ways of incorporating adaptive probabilistic models to EDAs. Section 6 presents factor graph based factorizations and Kikuchi approximations as our case of study. The design of the experiments and the numerical results are presented in Section 7. The paper ends with Section 8 where the conclusions of our paper are presented.

2 Estimation of Distribution Algorithms

2.1 Notation

Let X be a random variable. A value of X is denoted x. $\mathbf{X} = (X_1, \ldots, X_n)$ will denote a vector of random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $N = \{1, \ldots, n\}$, and $\mathbf{X}_S$ (respectively, $\mathbf{x}_S$) a subset of the variables of $\mathbf{X}$ (respectively, a subset of values of $\mathbf{x}$) determined by the indices in S. We will work with discrete variables.

The joint probability mass function of $\mathbf{x}$ is represented as $p(\mathbf{X} = \mathbf{x})$ or $p(\mathbf{x})$. $p(\mathbf{x}_S)$ will denote the marginal probability distribution for $\mathbf{X}_S$. We use $p(X_i = x_i \mid X_j = x_j)$ or, in a simplified form, $p(x_i \mid x_j)$, to denote the conditional probability distribution of X_i given $X_j = x_j$.

A *graphical model* for $\boldsymbol{X} = (X_1, \ldots, X_n)$ encodes a graphical factorization of a joint probability distribution. Commonly used graphical models include Bayesian networks, Markov networks and factor graphs.

2.2 EDAs

Estimation of distribution algorithms (EDAs) [9,13] are evolutionary algorithms that work with a set (or population) of points. Initially, a random sample of points is generated. These points are evaluated using the objective function, and a subset of points is selected based on this evaluation. Hence, points with better function values have a higher chance to be selected. Then a probabilistic model of the selected solutions is built, and a new set of points is sampled from the model. The process is iterated until an optimum has been found or another termination criterion is fulfilled.

Algorithm 1. Estimation of distribution algorithm

```
1 Set $t \Leftarrow 0$. Generate $M$ points randomly.
2 do {
3     Evaluate the points using the fitness function.
4     Select a set $S$ of $N \leq M$ points according to a selection method.
5     Calculate a probabilistic model of $S$.
6     Generate $M$ new points sampling from the distribution
      represented in the model.
7     $t \Leftarrow t + 1$
8 } until Termination criteria are met.
```

The general scheme of the EDA approach is shown in Algorithm 1. There are a number of selection methods that can be used. In the literature, truncation, Boltzmann, and tournament selection are commonly used with EDAs.

One essential assumption of these algorithms is that it is possible to build a probabilistic model of the search space that can be used to guide the search for the optimum. Thus, a key characteristic and crucial step of EDAs is the construction of this probabilistic model. If there is available information about the function (e.g. variable dependencies), this can be exploited by including parametrical and/or structural prior information in the model. Otherwise, the model is learned exclusively using the selected population. Several probabilistic models with different expressive power and complexity can be applied. These models may differ in the order and number of the probabilistic dependencies that they represent.

3 Work on Adaptive Genetic Algorithms

In this section, we make a short review of previous work on the design of adaptive genetic algorithms, emphasizing some of the issues that will be considered in the presentation of our proposal of adaptive EDAs.

In [24], an analysis of the way adaptation has been used in genetic algorithms (GAs) is done. Three principles that allow to study the role of adaptation are presented. These are:

- What is being adapted? (operators, parameters, etc.).
- The scope of the adaptation (i.e. does it apply to all the population, just to individual members, or just sub-components?).
- Basis for change (externally imposed schedule, fuzzy logic, etc).

One of the most important benefits of adaptive reproductive operators is that they permit a more flexible tuning between the goals of exploration and exploitation of the search space. This can be done by modifying the parameters associated to the operators, or by changing their frequency of application. The simplest adaptive GAs use a fixed set of operators and adapt the probability of application of those operators. Another class of adaptive GAs change the behavior of the operators over time.

Different techniques have been used to extract information from the search in order to determine adapting schedules of decision rules for adaptive GAs. This includes the use of fuzzy logic [6], inductive learning [23], and reinforcement learning [16].

There are important similarities between the study of adaptation in GAs and EDAs. Research done on those components common to GAs and EDAs can be applied to the second class of algorithms with minor modifications. This research comprises, for instance, the use of variable population sizes, adaptive selection schedules, etc. The techniques employed to extract information about the search are also of direct application to the conception of adaptive EDAs.

The study of adaptation in EDAs must take into account some main differences between EDAs and GAs. One of these differences is that reproduction, as implemented in GAs, can provide more fine grained information about the effect of the reproduction operator that the way reproduction is accomplished in EDAs. For instance, the concept of "safety ratios" [22] refers to the probability that a new point generated by the application of reproductive operators would be fitter than its parent(s). This probability, that can be used as a measure of the operators effect, is calculated using information about the fitness of the parents. Since the influence of the parents in EDAs (the whole selected set) is mediated by the existence of a probability model, it is not possible to define a parent-to-offspring correspondence. Instead, macroscopic measures (e.g. average fitness of the population) must be used to describe the effect of the reproductive operators.

But even if detailed information about the relationship parent-offsprings will not be available in EDAs, these algorithms expand both, the sources of statistical information about the search, and the range and variety of applications of this information. Probabilistic models are the main specific source of statistical information in EDAs, but also the information collected during the learning of these models (the model search step) could be used for adaptation.

4 Improving the Search: Adaptive EDAs

Our initial analysis will be led to the identification of the particular features of EDAs that should be modified for the design of adaptive algorithms.

There are a number of issues that need to be addressed in order to accomplish the conception of adaptive EDAs. The following questions will help us to guide our analysis:

- Which EDA components can be adapted to the search?
- How can an EDA adapt its components and parameters by itself?
- How to obtain relevant information from the search for adaptation?
- Which is the available repertoire of possibilities for adapting EDAs?

Among the components of EDAs that can be adaptively modified during the search are the fitness evaluation, the selection method, the elitism method and the parameters (i.e. population size, selection and elitism parameters). Even if research focused on these components is important we will consider in our analysis those other components that are specific to EDAs. These include: the class of probabilistic models, the methods used for learning them and the sampling methods. In Section 5, we will analyze how these components can be modified in order to guarantee an adaptive behavior of EDAs.

Once the parts of the algorithms that can adapt to changes during the search have been identified, it is important to define the ways they can be adapted to the search and which the requirements to implement the changes are. Two essential issues have to be considered for the definition of such strategies. They are: the information about the search history and the general decision rules based on this information.

The first issue involves the collection, storing and interpretation of the data generated during the search. Not all the data available from the algorithm behavior is relevant to the purpose of taking decisions about the search strategies. Therefore, it is needed to set the sort of data that will be stored and eventually used by the algorithm. Furthermore, some data may require a preprocessing step before being used. The computational cost of this step should be estimated in order to guarantee that the gain due to increasing the algorithm adaptability is not achieved at the expense of an unbearable computational cost of preprocessing step.

The second issue, which is very related with information about the search history, it the strategy conceived for using this information. As in the case of adaptive GAs, this strategy can be defined using different machine learning techniques that will employ a particular class of the information available. The selection of the relevant information for adaptive EDAs presupposes that a strategy that will use this class of information has been determined.

We describe in some detail which the possible sources of information for adaptive EDAs and the preprocessing steps needed to use this information are. Main sources of information are the following:

- Fitness related measures:
  1. Measures of convergence.
  2. Measures of exploration and exploitation.
- Information about the interactions captured in the graphical model.

Fitness related measures are obtained from the fitness values of the solutions so far visited by the algorithm. Let $f(t)$ be the fitness function at generation t. Examples of these measures include:

- Average fitness and variance of the population $(\bar{f}(t),\sigma^2(f(t)))$.
- Response to selection: $R(t) = \bar{f}(t+1) - \bar{f}(t)$.
- Amount of selection: $S(t) = \bar{f}_s(t+1) - \bar{f}(t+1)$.
- Realized heritability $b(t) = \frac{R(t)}{S(t)}$.

The average fitness is used to compute the response to selection which is a general measure of the improvements obtained in the average fitness of the population by the application of the learning and sampling steps. However, an increase of $R(t)$ can hide a loss of diversity in the population. In these cases, the change in the fitness variance can support additional information about whether the population is really diverse. The mathematical framework that involves the use of $R(t)$, $S(t)$ and $b(t)$ was originally proposed in population genetics and has been applied before to the analysis of the breeder genetic algorithm [14]. We propose to apply these measures to evaluate the role of different operators (e.g. different classes of probabilistic models and sampling methods) and parameters used by EDAs.

Other measures that can be used as a source of information about the search are the average fitness of individuals in the selected population, the number of different solutions in the selected population and the number of generations spent without improvement.

Among the measures related to the probabilistic model that can be used for adaptation are: the number of edges, the size of the maximum clique, the number of maximal cliques, and the number of connected components of the graph.

Alternatives for adaptation in EDAs include the following:

- Varying the strength of selection according to the diversity of solutions.
- Choice of the probability model class according to the graph complexity.
- Determination of the sampling algorithm according to the graph topology.
- Increasing the population size to avoid stagnation of the search.

In [10], an adaptive schedule for the Boltzmann selection was introduced and compared with the truncation selection. Although both methods showed similar dynamics, EDAs with truncation selection reached better convergence rates and required less number of fitness evaluations. In [17], adaptive priors that relate the rate of variation of the population to the quality of the approximation learned by the model are proposed in the context of the mixture of trees factorized learning algorithm (MT-FDA) [21]. Better results than when using MT-FDA with static learning methods are achieved.

There is some recent work on the incorporation of adaptive techniques in EDAs [2,5,26]. This work has focused on optimization problems with continuous representation and the mechanisms of adaptation have been constrained to the change in the parameters governing the learning process for the probabilistic model of choice. Although some of the general issues we treat in this paper can be extended to problems with continuous representation, our proposal is introduced in the context of optimization problems with discrete representation. On the other hand, the scheme of adaptation we present allows to change the class of the probabilistic models during the evolution, expanding the class of components and the scope of actions available to deal with the exploration of the search space.

In the next section, we will focus on the definition of a framework that allows to change the class of probabilistic model and the sampling algorithm during the search.

5 Adapting the Class of Probabilistic Models in EDAs

In order to explain the ways adaptation can be introduced in EDAs we start by presenting a generalized EDA that comprises different types of probabilistic models, learning and sampling algorithms. We constrain our analysis to EDAs based on undirected graphical models [15, 19, 20]. The pseudocode of the generalized EDA is shown in Algorithm 2.

Algorithm 2. Generalized EDA

```
1  Set $t \Leftarrow 0$. Generate $M$ points randomly.
2  do {
3      Select a set $S$ of $N \leq M$ points according to a selection method.
4      Learn an undirected-graph-based representation of the dependencies
       in $S$.
5      Using the graph, determine a class of graphical models or
       approximation strategy to approximate the distribution
       of points in $S$.
6      Determine an inference algorithm to be applied in the graphical
       model.
7      Generate $M$ new points from the model using the inference method.
8      $t \Leftarrow t + 1$
9  } until Termination criteria are met.
```

The most relevant feature of the generalized EDA is that it allows the use of different classes of graphical models at each generation. The model choice should be related to the complexity of the data and to the patterns of interaction between the components of the problem. In situations in which there are few interactions between the variables, we could choose a simple class of models and avoid more complex learning algorithms (e.g. those required by Bayesian networks). Choosing a simpler model can thus lead to an advantage in terms of computational time. Additionally the marginal probabilities of a probabilistic model with lower order dependencies could be more accurately estimated from small data samples.

Using different classes of graphical models during the search will also allow to incorporate different sampling techniques that determine different ways of searching for solutions. Therefore, the dynamic change of the probabilistic model will need an automatic procedure to select among the different types of graphical models. The topological characteristics of the undirected graphs learned are plausible information for this decision. The number, size, and cardinality of the variables (number of values) of each clique are three of the issues that influence the feasibility of the model for estimating the marginal probabilities and sampling new solutions.

We will assume that an undirected graph that encodes the (in)dependence relationships between the variables is given. Given the structure, we face two problems: 1) To decide which candidate probabilistic models could be used as approximations, and 2) To define which criteria to take into account to choose among them.

5.1 Alternatives for Probabilistic Modeling

Table 1 shows a number of alternatives for selecting a probability model according to the graph structure. Column 1 (Graphs) describes whether the approximation comprises all and only those dependencies in the graph (exact), a subgroup of the dependencies (subgraph) or all the dependencies and additional dependencies (triangulated graph). Column 2 (Graphical models) describes different situations that could be faced (e.g. univariate –there are not dependencies–, junction tree –valid factorization–, etc.). Column 3 (Inference) shows different sampling algorithms that can be used according to the model. They comprise: probabilistic logic sampling (PLS), Gibbs sampler (GS), most probable configurations (MPC), most probable configurations with belief propagation (MPC-BP), most probable configurations with loopy belief propagation (MPC-loopy BP), and most probable configurations with generalized belief propagation (MPC-generalized BP).

Table 1. Approximation strategies, graphical models, and inference methods to be employed by EDAs based on undirected graphs

Graphs	Graphical models	Inference
exact graph	univariate	PLS,MPC
	junction tree	PLS,MPC-BP
	junction graph	PLS,MPC-BP
	clique-based Kikuchi approximation	GS
	Bethe approximation	MPC-loopy BP
	Kikuchi approximation	MPC-generalized BP
subgraph	univariate	PLS,MPC
	junction tree	PLS,MPC-BP
	junction graph	PLS,MPC-BP
	clique-based Kikuchi approximation	GS
	Bethe approximation	MPC-loopy BP
	Kikuchi approximation	MPC-generalized BP
triangulated graph	junction tree	PLS,MPC-BP

5.2 Decision Criteria for Choosing the Model

The second question is the definition of the decision criteria for selecting among the alternatives. Without considering information about the search state, something that will be required for adaptive EDAs, the two main criteria to take into account are the accuracy and the complexity of the approximation. The accuracy of the approximation can be roughly estimated by measuring the number of interactions of the original graph covered by the approximation and the strength of the interactions covered.

Complexity is related to the size of the factors involved in the factorization. One way to choose between the classes of possible approximations according to their complexity is to constrain the size of the largest marginal table as well as the number of factors. To do this, a first step is to calculate all the maximal cliques of the graph and determine the size of the probability tables. To simplify our analysis, we will assume that all the variables have the same cardinality and, therefore, the largest marginal table will correspond to the maximum clique of the graph. The analysis can be generalized to the case where variables have different number of values.

If the graph is triangulated, and the maximum clique of the graph fulfills the complexity constraint, any of the alternatives listed in Table 1 as *exact graph* could be applied. These alternatives respect all the original dependencies that exist in the graph. Nevertheless, the chosen sampling method may determine that only an approximation is achieved.

If the graph is not triangulated, then one possibility is to triangulate it, compute the maximum cliques of the graph, evaluate whether the complexity constraint is fulfilled for the triangulated graph, and in that case, apply any of the alternatives listed in Table 1 as *triangulated graph*.

If the complexity constraint is not fulfilled in the original or in the triangulated graph, then other types of approximation must be tried. One possibility is to simplify the graph by splitting the largest cliques, something that can be done by removing edges. Another possibility is to make the graph sparser in one step previous to the calculation of the cliques.

The most common method applied for inference in the context of EDAs is the PLS. It starts from an order of the variables imposed by the structure of the graphical model. Each variable is sampled given the values assigned to its ascendants in the order. PLS can be applied to the junction tree and junction graph, but it cannot be applied to any other approximation listed in Table 1 because, in the general case it is not possible to find an order of the variables for these approximations.

For Kikuchi approximations that use clique-based decompositions [19], GS can be employed. In this case, the conditional probability distributions serve to determine the transitions in the Markov chain. The drawback of using Gibbs sampling is that if the most probable configuration has an exponentially small probability a large number of configurations will need to be visited to hit the optimum. A partial remedy to this situation is the combination of Kikuchi approximations with propagation based inference methods [7].

6 A Case Study: Generalized Factorized Distribution Algorithms

We will focus now on the class of EDAs that explicitly construct a factorization of the distribution.

6.1 Factorizations

In simple terms, a factorization of a distribution $p(\mathbf{x})$ will be a product of probability distribution $p(\mathbf{x}_s)$ each of which will be called a factor. Factorizations are important because they allow us to obtain a condensed representation of otherwise very difficult to store probability distributions. Generally, graphical models serve to define one or more factorizations of $p(\mathbf{x})$.

The structure of a factorization can be directly recovered from a chordal graph as done in the factorized distribution algorithm (FDA) [12] or learned from data [15, 19, 20]. Factorizations that satisfy the running intersection property (RIP) are called *valid* [12]. In [19], invalid factorization were further classified in "ordered" and "messy" regarding the number of factors that are part of the factorization. Most of EDAs employ valid factorizations. EDAs that work with messy factorizations were presented in [19, 20].

FDA can work with invalid factorizations [12] but in this case, the convergence properties proved for when valid factorizations are employed do not hold. Valid factorizations can also be obtained from directed graphs as those used by EDAs based on Bayesian networks [4].

6.2 Factor Graphs and Factorizations

The analysis of the EDAs presented in this chapter will be based on the use of factor graphs. One convenient way of representing factorizations are factor graphs.

Factor graphs

A *factor graph* [8] is a bipartite graph that can serve to represent the factorized structure of a distribution. It has two types of nodes: variable nodes (which we draw as a circle), and factor nodes (which we draw as a square). In the graphs, factor nodes are named by capital letters starting from A, and variable nodes by numbers starting from 1. We will index variable nodes with letters starting with i, and factor nodes with letters starting with a. The existence of an edge connecting variable node i to factor node a means that x_i is an argument of function f_a in the referred factorization. Figure 1 (left) shows a factor graph with two factor nodes and five variable nodes. The associated undirected graph (right) have two maximal cliques.

In [1], Gibbs distributions are associated with factor graphs. A factor f with scope $\mathbf{X}_S$ is a mapping from $\mathbf{x}_S$ to $\mathcal{R}^+$. A Gibbs distribution $p(\mathbf{x})$ is associated with a set of factors $\{f_a\}_{a=1}^m$ with scopes $\{\mathbf{X}_{S_a}\}_{a=1}^m$, such that

$$p_f(\mathbf{x}) = \frac{1}{Z} \prod_{a=1}^{m} f_a(\mathbf{x}_{S_a}) \tag{1}$$

Factorizations commonly used by EDAs can be represented by factor graphs. For a given function, if its definition sets and the corresponding subfunctions are known then it is possible to associate a factor to each definition set. The

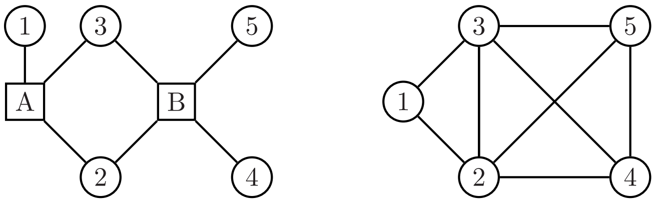

Fig. 1. Factor graph (left) and associated undirected graph with two maximal cliques (right)

corresponding factor graph distribution would be given by (1). At each generation of the EDA, a different factor graph distribution can be learned by taking $f_a(\mathbf{x}_{S_a}) = p_a(\mathbf{x}_{S_a})$ where $p_a(\mathbf{x}_{S_a})$ are the marginal probability distributions learned from the data.

If the factorization is valid then $Z = 1$, and the factorization given by the factor graph is exact. But in the general case, a factorization represented by a factor graph does not have to satisfy the RIP. As a consequence, $Z \neq 1$ and inference of points from the factorization is not straightforward. One alternative in these cases is to learn an approximation. One example of such approximations is the Kikuchi approximation of the distribution.

6.3 Kikuchi Approximation of a Distribution

The Kikuchi approximation of a distribution has three essential components:

1. An initial representation of the interactions of the variables given by a graphical model.
2. A set of regions comprising sets of variables.
3. An overcounting number associated to each region.

In [19], the Kikuchi approximation of a distribution was defined from an independence graph. Initial regions corresponded to the maximal cliques of the graph and the rest of regions were found using the cluster variation method [25]. Overcounting numbers c_R corresponding to each region R were constrained to be calculated using the following recursive formula:

$$c_R = 1 - \sum_{\substack{S \in \mathcal{R} \\ R \subset S}} c_S, \tag{2}$$

where c_S is the overcounting number of any region S in $\mathcal{R}$ such that S is a superset of R. c_R values corresponding to the initial regions are equal to 1.

Given a factor graph, a straightforward generalization of Kikuchi approximations for factor graphs will associate each factor of the graph with an initial region of the Kikuchi approximation. From the set of initial regions the Kikuchi approximation is constructed using the cluster variation method. The overcounting numbers are also calculated using Equation (2).

Given a set of regions $\mathcal{R}$ calculated as explained before, the Kikuchi approximation, $k(\mathbf{x})$, of the probability distribution $p(\mathbf{x})$ is defined as:

$$k(\mathbf{x}) = \prod_{R \in \mathcal{R}} p(\mathbf{x}_R)^{c_R} \tag{3}$$

An important property of the Kikuchi approximation is that, if the factorization is valid, the corresponding Kikuchi approximation is exact, i.e. it will give the original factor graph distribution constructed from the marginal probabilities. On the other hand, a probability function $\tilde{p}(\mathbf{x})$ based on the Kikuchi approximation can be found by normalizing k.

$$\tilde{p}(\mathbf{x}) = \frac{k(\mathbf{x})}{\sum_{x'} k(\mathbf{x}')}$$

Another alternative to deal with factor graph distributions is the use of Markov blanket canonical factorizations [1]. In this case, factor graph distributions are parameterized as a product of local probabilities only. These local probabilities are defined over factor scopes and their Markov blankets [1].

6.4 Learning and Sampling the Kikuchi Approximation from a Factor Graph Distribution

The complexity of learning a Kikuchi approximation from a factor graph distribution is related to whether the structure is previously known, or both the structure and the parameters of the distribution have to be learned. In the first case, and assuming that the maximum size of the factors is feasible regarding the cost of computing and storing the parameters, learning is reduced to estimate the parameters from the data. In the second case, structural learning is required.

The complexity of sampling a factor graph distribution depends on whether the factorization is valid or invalid. In the first case, probabilistic logic sampling could be applied. In second case, more costly techniques like Gibbs sampling [19] and belief propagation [11] could be employed.

To learn the structure of the factor graph we follow the approach described in Algorithm 3.

Algorithm 3. Algorithm for learning a factor graph representation

```
1  Learn an independence graph $G$ from the data (the selected set of
   solutions).
2  If necessary, refine the graph.
3  Find the set $\mathcal{C}$ of all the maximal cliques of $G$.
4  Associate a factor to each maximal clique of the graph.
5  Find the set of regions $\mathcal{R}$.
6  Find the marginal probabilities for the regions.
```

Given an undirected graph $G = (V, E)$, a clique in G is a subset of V for which there exists an edge between every pair of vertices. A clique is the maximum clique of the graph if it is a clique with the highest number of vertices. The choice of taking maximal cliques as factors is related to the properties of the Kikuchi approximation for clique based decompositions shown in [19].

The independence graph is learned using independence tests. We use the Chi-square independence test. If two variables X_i and X_j are dependent with a specified level of significance α, they are joined by an edge. α is a parameter of the algorithm. The algorithm weights each edge $i \sim j$ in the independence graph with a value $w(i, j)$ stressing the pairwise interaction between the variables. We use the value of the Chi-square test to set $w(i, j)$.

If the independence graph is very dense, the dimension of the cliques will increase beyond a feasible limit. It is important to impose a limit r to the size of the maximum clique. An alternative solution to this problem is to make the graph sparser in one step previous to the calculation of the cliques. This has been done by allowing a maximum number $r - 1$ of incident edges to each vertex. If one vertex has more than $r - 1$ incident edges, those with the lowest weights are removed. In this way, the size of the maximum clique will always be smaller or equal than r. To find all the maximal cliques of the graphs the Bron and Kerbosch algorithm [3] is used. Junction graphs and junction trees can be constructed using a subset of these cliques [18].

Since in the general case, the partition function Z is not known, we use GS to sample points from $k(\mathbf{x})$. VS, Cy, and In are defined as the parameters of the GS algorithm. VS is the type of visitation scheme, and defines the way in which the variables are selected for update. Random ($VS = 0$), or fixed ($VS = 1$) visitation schemes can be used. Cy is the number of cycles of the GS algorithm. One cycle comprises the update of n variables. In is a parameter that determines the way the initial vector of the GS is constructed. The vector where the GS starts from can be randomly selected ($In = 0$), or sampled from an approximate factorization found using a chordal subgraph of the independence graph ($In = 1$). More details about the GS algorithm can be found in [19].

6.5 Probabilistic Operators

From a given independence graph we will define five different classes of factorizations. We will call these classes *probabilistic operators*. To further specify and control their behavior we will employ parameters r, α, Cy and In. Parameters r and α are general parameters because they impose constraints to the independence graph. These constraints influence the type of factorizations. For instance, if $r = 1$ the graph will be disconnected and the only possible factorization is the univariate, similarly if $r = 2$, the graph will be a set of isolated nodes, paths and cycles. Notice that r represents a constraint on the *maximum clique* of the graph. Manipulating α the density of the graph can be changed, influencing the number and size of the factors. Parameters Cy and In will only affect the Kikuchi approximations. A description of the probabilistic operators follows.

- MK0: A Kikuchi approximation that uses as starting vector for GS a vector sampled from an invalid junction-graph-based factorization.
- MK1: An invalid junction-graph-based factorization.
- MK2: A Kikuchi approximation that uses as starting vector for GS a random vector.
- MK3: An invalid junction-tree-based factorization.
- MK4: A Kikuchi approximation that uses as starting vector for GS a vector sampled a valid junction-tree-based factorization.

7 Experiments

The objectives of our experiments are twofold. The first is to study the influence of the different probabilistic operators in the dynamics of the search and the way they interact. Our analysis will be based on the descriptive measures presented in Section 4. The second goal is to extract from this analysis a number of rules that can be translated to the definition of adaptive EDAs. To evaluate the algorithms, we have selected some difficult instances of the satisfiability (SAT) problem.

7.1 SAT Problem

Let $U = \{u_1, \cdots, u_n\}$ be a set of n Boolean variables. A (partial) truth assignment for U is a (partial) function $T : U \rightarrow \{true, false\}$. Corresponding to each variable u are two literals, u and $\neg u$. A literal u (resp. $\neg u$) is $true$ under T if $T(u) = true$ (resp. $T(u) = false$). We call a set of literals a clause, and a set or sequence (tuple) of clauses a formula. Let ϕ be a formula and C a clause in ϕ. We say that a truth assignment T for U satisfies C if at least one literal $u \in C$ is true under T, and T satisfies ϕ if it satisfies every clause in ϕ. The satisfiability problem is the problem of finding a solution for a formula.

In our representation, variable X_i is associated to the Boolean variable u_i, and $(u_i = true) \Leftrightarrow (x_i = 1)$. As the objective function we use the sum of clauses satisfied by the solution.

As as set of instances, we have used the uniform random-3-SAT, which is a family of SAT problems distributions obtained by randomly generating 3-CNF formulae. The test-set $uf-75$ comprises 1000 instances sampled from the phase transition region of uniform Random-3-SAT. The instances, as well as a detailed explanation about the way they were generated, can be found in the SATLIB benchmark[1]. Each instance in $uf - 75$ has 75 variables with 325 clauses.

7.2 Parameters of the Algorithms

In all the experiments, we use truncation selection with parameter $T = 0.15$. The population size was $N = 500$. The best solution in each generation is passed to

[1] http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/Benchmarks/SAT/RND3SAT/descr.html

the new generated population. The maximum number of generations was set to 250. The algorithm stops when the optimum is found or the maximum number of generations is reached. We notice that the maximum number of evaluations is relatively small for reaching the optimum of some of the instances considered. However, our goal was not optimize the parameters of the algorithms but to analyze, for these parameters, the effect of the probabilistic operators. Otherwise noticed, we execute 100 runs of each algorithm.

In the experiments we considered three different scenarios. *Random EDA*, in which at each generation, the probabilistic operators and/or the parameters that will be applied are randomly determined. Otherwise stated, the uniform distribution is used for randomly selecting from the parameter set of possible values. This random scenario is conceived for collecting information about the role of the different probabilistic operators. In the *static EDA* scenario, an EDA with fixed probabilistic operator and parameters is run. These cases are considered as a reference for contrasting results. Finally, in the *adaptive EDA* scenario, an EDA with varying probabilistic operators and/or parameters is run.

7.3 Numerical Experiments

In the first experiment, we consider a random EDA scenario where, at each generation, one of the five probabilistic operators presented in Section 6.5 is randomly selected. Fixed parameters were $\alpha = 0.7$ and $r = 8$. Once the operator has been determined, the parameter Cy is randomly selected. For MK0 and MK4, $Cy \in \{1, \dots, 6\}$, for MK2, $Cy \in \{10, 20, 30\}$. From 100 runs of the algorithm, information about 88369 generations is collected. For each generation, we have the operator applied and it is possible to compute the response to selection it causes. Figure 2 shows the histogram of the number of times that each of the probabilistic operators causes a positive response to selection (i.e. an increase in the average fitness of the population is achieved). Two main features can be noticed from the graph. First, in terms of $R(t)$, the performance of $MK2$ operator is the worst. On the other hand, the effects operators $MK0$ and $MK4$ are very similar. The same fact holds for operators $MK1$ and $MK3$. This may indicate that, for this experiment, a factor graph does not support more relevant information about the interactions in the graph than that that can be represented by a junction tree.

In Table 2, the change in $R(t)$ is detailed. We compute the average of $R(t)$ for each of the probabilistic operators and each of the corresponding Cy values. The best values of the response to selection are reached for operators MK1 and MK3. For operators MK0 and MK4, $R(t)$ decreases with higher values of Cy. For MK2, the decrease in the response of selection is slightly slowed down when Cy is increased. In general, the average results seem to indicate that valid factorizations guarantee higher values of $R(t)$. However, average results can be deceptive. Therefore, we have computed a classification tree to determine the best suited probabilistic operators according to the variance of the population. To compute the classification tree, the treefit procedure implemented in the Matlab software has been employed.

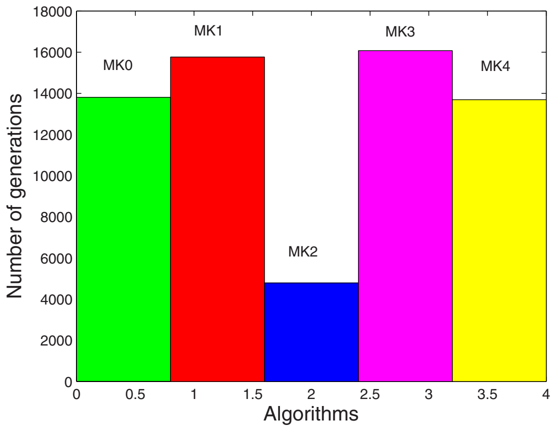

Fig. 2. Average response to selection for different probabilistic operators

Table 2. Average response to selection for different probabilistic models of an EDA with multiple models and sampling algorithms

Cy/EDAs	MK0		MK1		MK2		MK3		MK4	
	mean	σ^2	*mean*	σ^2	*mean*	σ^2	*mean*	σ^2	*mean*	σ^2
1	3.51	9.26	3.68	8.78			3.68	9.00	3.47	9.60
2	3.27	11.62	3.61	8.95			3.65	9.20	3.28	12.67
3	2.75	15.62	3.64	8.79			3.70	9.24	2.91	14.95
4	2.53	18.74	3.75	8.96			3.78	9.39	2.58	18.53
5	2.12	22.50	3.69	8.79			3.78	9.06	2.29	21.10
6	1.92	25.62	3.77	9.26			3.66	9.10	2.06	23.97
10					−12.72	297.0				
20					−12.64	327.2				
30					−12.23	328.6				
all	2.69	17.50	3.69	8.92	−12.53	317.69	3.72	9.18	2.92	15.53

We have taken as predictor variables, the (discretized) variance and a variable Rs, such that $Rs = 0$ if $R(t) < 0$, and $Rs = 1$ if $R(t) > 0$. The categorical dependent class is the type of probabilistic operator, taking into account the value of Cy. Since only three different values where considered for the operator MK2, we have grouped the values for MK0 and MK4 in three groups ($Cy \leq 2$, $Cy \in \{3, 4\}$, and $Cy \geq 5$). Similarly, occurrences of MK1 and MK3 have been equally divided in three groups, but in this case the membership to the group has no implications for the classification.

Figures 3 and 4 show the computed classification tree. It can be observed that most, although not all, of the choices of probabilistic operators that cause a negative value of $R(t)$ (Figure 3) correspond to the MK2 operators. Conversely,

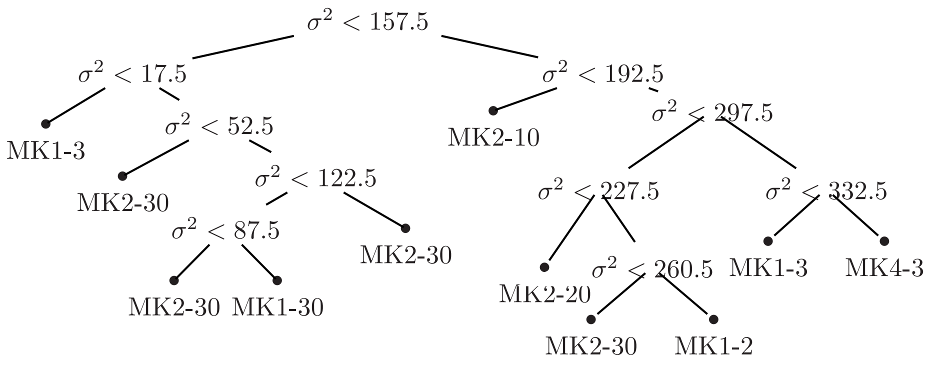

Fig. 3. Classification tree showing the relationship between the variance and the probabilistic operators when the response to selection is negative ($Rs < 0.5$)

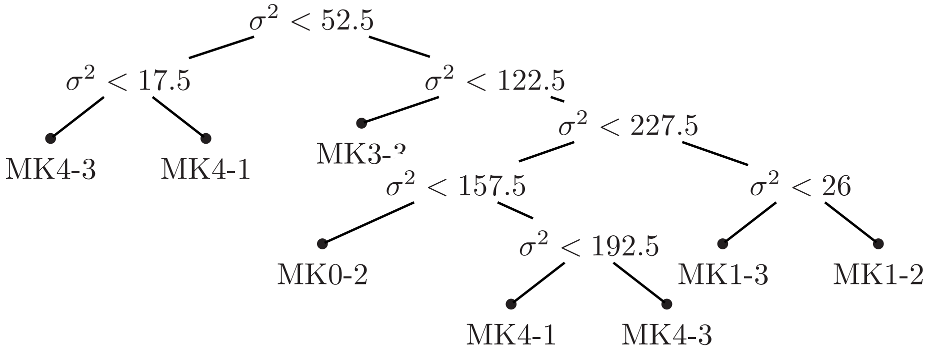

Fig. 4. Classification tree showing the relationship between the variance and the probabilistic operators when the response to selection is positive or zero ($Rs > 0.5$)

MK2 does not appear associated to any value of the variance in the main right branch of the tree (Figure 4).

We have also analyzed the effects that the different probabilistic operators have in the variance of the algorithm. Figure 5 shows the relationship between the response of selection and the variance for MK1 and MK2 operators. This figure reveals an important result. Even if the average value of $R(t)$ is negative for MK2, this operator has a higher variance and may obtain an improvement in the fitness average superior to operator MK1. In simpler terms, operator MK1 regularly improves the solutions but this improvement is constrained. On the other hand, operator MK2 seldom improves solutions but when it does it, the improvement can be important. Additionally, the improvement achieved by operator MK1 is achieved at the cost of an important loss of variance. This is not the case for operator MK2, when the response to selection is improved, also the variance of the generated solutions is increased.

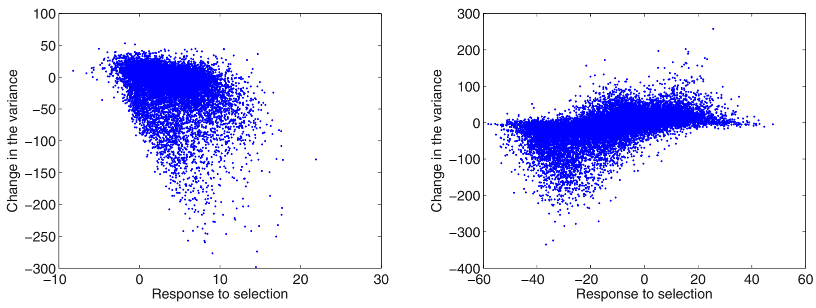

Fig. 5. Relationship between the response to selection and the variance for probabilistic operators MK1 (left) and MK2 (right)

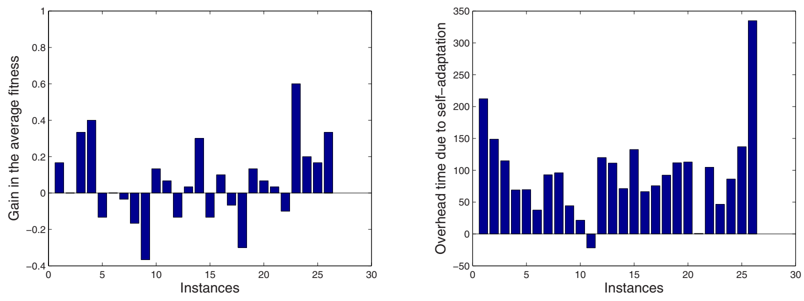

Fig. 6. Gain in the average fitness (left) and overhead time (right) due to the self-adaptation process

We have investigated in our experiments the influence of parameters α and r (data not shown) and extracted decision rules using classification trees. We have evaluated EDAs that incorporate these rules but, for the instances considered, these results are not statistically significant. It turned out, that, at least for the instances of the SAT problem, adaptation based on a combination of the exploratory effect of the MK2 operators with the rest of the operators gives the best results. The resulting algorithm alternates the application of the operators pursuing the goal of balancing exploration and exploitation.

Three different criteria are used to identify a loss of diversity in the population and change the type of probabilistic operator applied. These criteria are: the fitness variance of the selected population is zero, the number of different individuals in the selected set is below half the size of the selected set, and if two consecutive generations have equal average fitness of the selected population.

When one of these criteria is fulfilled, operator MK2 is applied with a randomly selected value of $Cy \in \{1, \dots, 5\}$. In Table 3, the results for different static and adaptive EDAs and the four instances considered in our experiments are presented. In the table, S is the number of times the optimum has been

Table 3. Success rate and average number for different variants of static and adaptive EDAs

EDAs	scheme	α	r	steps	uf001		uf002		uf003		uf004	
					S	$\bar{f}$	S	$\bar{f}$	S	$\bar{f}$	S	$\bar{f}$
	Random	0.70	8	–	39	324.09	40	324.26	0	322.63	1	322.73
MK1	static	0.92	8	–	43	324.11	46	324.29	0	323.03	0	322.09
MK2	static	0.92	8	2	39	324.06	52	324.49	0	323.11	0	322.74
	adaptive	0.92	8		67	324.66	87	324.87	0	323.80	0	323.64
	adaptive	0.92	6		47	324.40	57	324.53	0	323.39	0	323.16
	adaptive	0.92	4		62	324.59	67	324.66	1	323.40	2	323.23
	adaptive	0.92	2		62	324.59	59	324.58	0	323.36	2	323.35
	adaptive	0.70	1		35	323.90	67	324.63	0	323.27	0	323.00
	adaptive	0.70	3		50	324.43	68	324.68	1	323.34	2	323.20
	adaptive	0.70	6		45	324.19	50	324.46	0	323.36	0	322.99

found and $\bar{f}$ the average fitness of the best found solution. Notice, that for instances $uf003$ and $uf004$ the optimum is very difficult to find. In these cases, we take $\bar{f}$ to evaluate the performance of the algorithms. The random EDA is the algorithm for which previous results have been presented in this section. The adaptive EDAs ($r = 8$) clearly outperforms the other algorithms. The analysis of the table also reveals that factors α and r can play an important role in the performance of the algorithms. Finding schedules for adaptively changing these values during the search should produce more efficient algorithms.

Additional experiments have been conducted for instances from $uf005$ to $uf030$ of the uf-75 benchmark. For these problems, we have compared the performance of the MK1 operator and the adaptive EDA with $\alpha = 0.92$ and $r = 8$ parameters. For each problem, 30 experiments has been conducted from which the average fitness of the best solution found and the overhead time due to the adaptation process have been computed. The results are shown in Figure 6. The adaptive EDA improves the results in 15 of the 26 instances. However, in 9 of the instances worse results are achieved. In 25 of the 26 instances there is a cost due to the adaptation process. Although, the application of the adaptive schedule does not always guarantee an improvement of the results, the improvement achieved can be very important for some of the instances, justifying the additional time spent for the adaptation.

8 Conclusions

In this chapter, we have proposed a general framework for the analysis and design of adaptive EDAs. We have analyzed the main differences between GAs and EDAs regarding the ways adaptation can be incorporated to the algorithms. We have focused on feasible ways of adaptively combining different types probabilistic models in EDAs. Using probabilistic operators based on factor graph based factorizations and Kikuchi approximations we have introduced an adaptive schedule and evaluated its performance in the optimization of different SAT instances. Our preliminary results show that adaptive EDAs can outperform static EDAs.

The design of flexible, adaptive EDAs, is a difficult challenge that in order to be overcome may require the combination of results from different fields (e.g. data mining, machine learning, automatic control, etc.). However, the benefits to be obtained from this type of algorithms justify the efforts on this trend. We consider the work presented in this chapter as an initial step in this direction.

Acknowledgements

The authors thank the reviewers for useful comments on the paper. This work was supported by the SAIOTEK-Autoinmune (II) 2006 and Etortek research projects from the Basque Government. It has been also supported by the Spanish Ministerio de Ciencia y Tecnología under grant TIN 2005-03824.

References

1. Abbeel, P., Koller, D., Ng, A.Y.: Learning factor graphs in polynomial time and sample complexity. Journal of Machine Learning Research 7, 1743–1788 (2006)
2. Bosman, P.A., Grahl, J.: Matching inductive search bias and problem structure in continuous estimation of distribution algorithms. European Journal of Operational Research (to appear, 2007)
3. Bron, C., Kerbosch, J.: Algorithm 457—finding all cliques of an undirected graph. Communications of the ACM 16(6), 575–577 (1973)
4. Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, pp. 151–173 (1999)
5. Grahl, J., Bosman, P.A., Rothlauf, F.: The correlation-triggered adaptive variance scaling idea. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. GECCO 2006, pp. 397–404. ACM Press, New York (2006)
6. Herrera, F., Lozano, M.: Adaptive genetic algorithms based on fuzzy techniques. In: Proceedings of Information Processing and Management of Uncertainty Conference. IPMU 1996, pp. 775–780 (1996)
7. Höns, R., Santana, R., Larrañaga, P., Lozano, J.A.: Optimization by max-propagation using Kikuchi approximations, (submitted for publication, 2007)
8. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
9. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
10. Mahnig, T., Mühlenbein, H.: Comparing the adaptive Boltzmann selection schedule SDS to truncation selection. In: Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS 2001), Havana, Cuba, March 2001, pp. 121–128 (2001)
11. Mühlenbein, H., Höns, R.: The estimation of distributions and the minimum relative entropy principle. Evolutionary Computation 13(1), 1–27 (2005)
12. Mühlenbein, H., Mahnig, T., Ochoa, A.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 213–247 (1999)

13. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
14. Mühlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm (BGA). Evolutionary Computation 1(4), 335–360 (1993)
15. Ochoa, A., Soto, M.R., Santana, R., Madera, J.C., Jorge, N.: The Factorized Distribution Algorithm and the junction tree: A learning perspective. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, March 1999, pp. 368–377 (1999)
16. Pettinger, J.E., Everson, R.M.: Controlling genetic algorithms with reinforcement learning. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, p. 692. Morgan Kaufmann Publishers Inc., San Francisco (2002)
17. Santana, R.: An analysis of the performance of the mixture of trees factorized distribution algorithm when priors and adaptive learning are used. Technical Report ICIMAF 2002-180, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (March 2002)
18. Santana, R.: A Markov network based factorized distribution algorithm for optimization. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 337–348. Springer, Heidelberg (2003)
19. Santana, R.: Estimation of distribution algorithms with Kikuchi approximations. Evolutionary Computation 13(1), 67–97 (2005)
20. Santana, R., Larrañaga, P., Lozano, J.A.: Mixtures of Kikuchi approximations. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 365–376. Springer, Heidelberg (2006)
21. Santana, R., Ochoa, A., Soto, M.R.: The mixture of trees factorized distribution algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2001, pp. 543–550. Morgan Kaufmann Publishers, San Francisco (2001)
22. Schaffer, J.D., Eshelman, L.J.: On crossover as an evolutionarily viable strategy. In: Belew, R.K., Booker, L.B. (eds.) Proceedings of the 4th International Conference on Genetic Algorithms, pp. 61–68. Morgan Kaufmann, San Francisco (1991)
23. Sebag, M., Schoenauer, M.: Controlling crossover through inductive learning. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) Parallel Problem Solving from Nature – PPSN III, pp. 209–218. Springer, Berlin (1994)
24. Smith, J.E., Fogarty, T.C.: Operator and parameter adaptation in genetic algorithms. Soft Computing - A Fusion of Foundations, Methodologies and Applications 2, 81–87 (1997)
25. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory 51(7), 2282–2312 (2005)
26. Zhou, A., Zhang, Q., Jin, Y., Sendhoff, B.: Adaptive modelling strategy for continuous multiobjective optimization. In: Proceedings of the 2007 Congress on Evolutionary Computation CEC 2007. IEEE Press, Singapore (2007)