Technical Report

UNIVERSITY OF THE BASQUE COUNTRY
Department of Computer Science and Artificial Intelligence

New methods for generating populations in Markov network based EDAs: Decimation strategies and model-based template recombination

Roberto Santana, Alexander Mendiburu, José Antonio Lozano

December 2012

San Sebastian, Spain
www.ccia-kzaa.ehu.es
hdl.handle.net/10810/4562

New methods for generating populations in Markov network based EDAs: Decimation strategies and model-based template recombination

R. Santana, A. Mendiburu, and J. A. Lozano
email:{roberto.santana,alexander.mendiburu,ja.lozano}@ehu.es
Department of Computer Science and Artificial Intelligence
University of the Basque Country (UPV/EHU)

Abstract

Methods for generating a new population are a fundamental component of estimation of distribution algorithms (EDAs). They serve to transfer the information contained in the probabilistic model to the new generated population. In EDAs based on Markov networks, methods for generating new populations usually discard information contained in the model to gain in efficiency. Other methods like Gibbs sampling use information about all interactions in the model but are computationally very costly. In this paper we propose new methods for generating new solutions in EDAs based on Markov networks. We introduce approaches based on inference methods for computing the most probable configurations and model-based template recombination. We show that the application of different variants of inference methods can increase the EDAs' convergence rate and reduce the number of function evaluations needed to find the optimum of binary and non-binary discrete functions.

1 Introduction

Early analysis of genetic algorithms (GAs) [15, 19] based on the building-block hypothesis and the schema theory identified two important issues for successful GA-based optimization. The so-called "building-block identification" and "building-block mixing or exchange" referred to the need of identifying valuable partial solutions, avoid their disruption, and effectively mix them in the generation of new solutions. GAs that apply simple genetic operators are usually unable to identify the problem variables that interact and therefore they are not very efficient at the time of generating new solutions. Extensive research has been devoted to design heuristic recombination operators based on a priori information about the problem domain and adaptive genetic operators that attempt to identify and preserve the linkage between the variables. However, these are not general and robust solutions to these problems.

Estimation of distribution algorithms (EDAs) [27, 41] are evolutionary algorithms (EAs) that deal in a different way with the "building-block identification" and "building-block mixing or exchange" questions. EDAs use machine learning techniques to learn

a probabilistic model from the selected solutions and to sample new solutions from this model. Therefore, they are able to capture, explicitly represent, and exploit the interactions between the problem variables. Although the schema theory may not be the most appropriate framework to analyze the way EDAs behave [52], there is overwhelming evidence that probabilistic modeling of selected solutions is a very competitive approach to the "linkage problem" as known in EAs. EDAs have been successfully applied to a variety of practical problems [4, 26, 29, 31, 63]. For recent surveys on EDAs see [26, 60].

EDAs can be classified according to the type of probabilistic models they use [27]. For discrete problems, research has been mainly focused on the class of EDAs based on Bayesian networks [11, 25, 43], and on EDAs that use Markov networks [49, 58, 62]. The choice of the probabilistic model also determines the machine learning algorithms used to learn the models and to sample from them. It has been argued [44] that although Markov networks may sometimes cover the same distribution that a Bayesian network using fewer edges in the dependency model, their sampling becomes more complicated than the sampling of Bayesian networks. This is indeed the case for the vast majority of Markov-network based EDAs that use Gibbs sampling [14] or similar Markov chain Monte Carlo (MCMC) [36] sampling strategies. These sampling methods can transfer the information contained in the Markov models to the generated solutions but are computationally costly since convergence of the Markov chain to the target distribution may take a long time and many variables updates can be necessary.

In this paper we introduce two methods for generating new solutions in EDAs based on Markov networks. The first method uses a combination of propagation-based inference with decimation strategies [23]. The use of decimation simplifies the process of computing an *approximate* most probable solution of the probabilistic model. This method is related to other algorithms that compute the solutions with maximum a posteriori probability (MAP) from a model and that have been previously applied to EDAs [20, 21, 33, 35, 51, 64]. The second method introduced for generating new solutions is a highly efficient algorithm that uses the Markov network structure as a template for crossover. Although the proposed methods are applied in the context of EDAs based on Markov networks we show that they can be also applied to EDAs based on directed models. The reason is that the inference techniques at the core of these methods can be implemented on a factor graph model [24], which is a general graphical model able to represent Markov and Bayesian networks.

This paper also investigates two other relevant questions related to the effectiveness of the algorithms for generating new solutions: 1) The benefits of combining local and global information about the search space and, 2) The impact of using different inference techniques for computing the MAP. We propose and compare different ways to combine the local information contained in the individuals with the global statistical information stored in the model. Similarly, we empirically analyze the effect that the use of inference techniques, exact and approximate, has in the behavior of EDAs. Furthermore, these two key questions are investigated together, i.e., we look at the identification of the best combination of the variants that addresses questions 1) and 2) above. Extensive experiments on discrete fitness functions with different levels of difficulty have been conducted. Our results show that the introduced procedures are an effective and efficient alternative to the single use of PLS and Gibbs sampling for Markov based EDAs.

The paper is organized as follows. In the next section, we introduce the Markov network factorized distribution algorithm with G-tests (MN-FDA^G) and review the Marko-

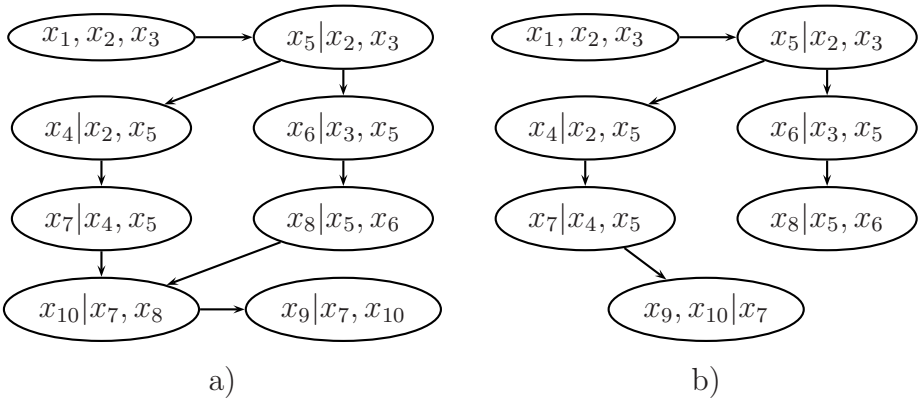


Figure 1: Ordered junction graph a) and junction tree b).

vianity based optimization algorithm (MOA) [59, 61], briefly discussing their learning and sampling steps. Section 3 presents factor graphs and message passing algorithms. In Section 4 we describe how belief propagation can be combined with decimation methods for computing the maximum a posteriori probability (MAP) of a factor graph. Section 5 introduces three variants of algorithms for generating the new solutions. Work in a number of areas related to our paper is surveyed and discussed in Section 6. Details on the algorithms' implementation, the optimization functions and the numerical results of our experiments are given in Section 7. The conclusions of our paper and lines for future work are presented in Section 8.

2 Markov network factorized distribution algorithm and the Markovianity based optimization algorithm

Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote a vector of discrete 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 $\{1, \ldots, n\}$, and X_S (respectively x_S) a subset of the variables of $\mathbf{X}$ (respectively $\mathbf{x}$) determined by the indices in S. We will work with positive distributions denoted by p. $p(x_S)$ will denote the marginal probability for $\mathbf{X}_S = \mathbf{x}_S$. We use $p(x_i \mid x_j)$ to denote the conditional probability distribution of $X_i = x_i$ given $X_j = x_j$.

MN-FDA and MOA are two different approaches to the use of undirected graphical models to represent probabilistic dependencies in EDAs. MOA is a more powerful algorithm in terms of the number of probabilistic dependencies that it is able to capture. However, it requires to use the Gibbs sampling (GS) algorithm to sample this model. GS is a computationally costly algorithm. MN-FDA can capture some of the dependencies that acyclic models are not able to represent, however the number of these dependencies that it is able to represent is rather limited. However, a good characteristic of MN-FDA is that it samples new solutions using PLS, which is a very efficient procedure. In the following sections we describe MN-FDA and MOA.

2.1 MN-FDA

MN-FDA defines a factorization in which the factors correspond to a set of maximal cliques organized as a labeled junction graph (JG). The cliques in the factorization can be ordered in such a way that at least one of the variables in every clique is not contained in the previous nodes in the ordering. MN-FDA allows the existence of cycles between the factors, therefore expanding the class of distributions represented by junction trees. The order of the cliques as represented in the junction graph is crucial for the implementation of the sampling procedure [49]. Figure 1 shows two factorizations defined by a junction graph (figure 1a) and a junction tree (figure 1b).

Algorithm 1: **MNFDA**

```
1 Set t ⇐ 0. Generate N ≫ 0 points randomly.
2 do {
3     Evaluate the points using the fitness function.
4     Select a set D_t^S of k ≤ N points according to a selection method.
5     Learn a labeled JG from D_t^S.
6     Calculate the marginal probabilities for all the cliques in the JG.
7     Generate the new population sampling from the JG.
8     t ⇐ t + 1
9 } until Termination criteria are met
```

The general steps of MN-FDA are shown in Algorithm 1. The steps that describe the way the probabilistic model is learned are shown in Algorithm 2. In [49], the chi-square independence test was used to determine if two binary variables X_i and X_j were dependent with a specified level of significance α. In that case variables were joined by an edge in the graph and the value of the chi-square test was used as a value $w(i, j)$ stressing the pairwise interaction between the variables. In this paper, we extend MN-FDA to deal with discrete non-binary problems. To compute dependencies in this case, we use the G-test of independence or likelihood ratio statistics. G-tests are a subclass of likelihood ratio tests, a general category of tests that have many uses for testing the fit of data to mathematical models [32]. Unlike the chi-square test, G-values are additive.

The G-test general formula is:

$$G = 2 \sum_i O_i \frac{O_i}{E_i} \tag{1}$$

where O_i and E_i are respectively the observed and expected frequencies.

The distribution of G is approximately a chi-squared distribution, with the same number of degrees of freedom as in the corresponding chi-squared test. To compute the test, we use the relationship between the G-test and the mutual information:

$$G(X_i, X_j) = 2NMI(X_i, X_j) \tag{2}$$

where MI is the mutual information and N is the number of samples.

Different papers have compared the chi-square statistics and the G statistics in terms of their asymptotic efficiency [17, 48]. It has been also found that G statistics are adequate and better than chi-square statistics when there are more than five degrees of freedom and expected frequencies greater than or equal to 3 [13, 68]. Since the use of the G-statistics implies a different way to determine the relationships between variables and different characteristics of the learning algorithm, we call the MN-FDA that uses this test as MN-FDA^G.

Algorithm 2: **Model learning**

1 Learn an independence graph $\mathcal{G}$ using the G-test.
2 If necessary, refine the graph.
3 Find the set L of all the maximal cliques of $\mathcal{G}$.
4 Construct a labeled JG from L.
5 Find the marginal probabilities for the cliques in the JG.

Algorithm 3: **Algorithm for learning a** JG

1 Order the cliques in L in a decreasing order according to their weights.
2 Add element $L(1)$ to list L'.
3 Remove element $L(1)$ from L.
4 While L is not empty
5 Find the first element C in L such that $C \cap (L'(1) \cup L'(2) \cdots \cup L'(NCliques)) \neq C$, and the number of variables in $C \cap L'(1) \cap L'(2) \cdots \cap L'(NCliques)$ is maximized.
6 If $C = \emptyset$ remove all the elements in L
7 else insert C in L'.

Dense graphs are made sparser by allowing a maximum number of incident edges to each vertex. If the vertex has more than r incident edges, those with the lowest weights are removed. In this way the size of the maximum clique will be always smaller than or equal to r. To find all the cliques of the graphs the Ken and Kerbash algorithm [6] was used. Once all the cliques have been found they are stored in a list L, and their weights are calculated from the information about dependencies. The weight of any subgraph $\mathcal{G}'$ of $\mathcal{G}$ is calculated as $W(\mathcal{G}') = \sum_{i \sim j \in \mathcal{G}'} w(i,j)$ where $w(i,j)$ is the result from the G-test. Finally, Algorithm 3 receives the list of cliques L with their weights, and outputs a list L' of the cliques in the labeled JG. The first clique in L' is the root, and the labels of cliques in the labeled JG correspond to their position in the list. To sample from the junction graph, MN-FDA uses PLS.

2.2 MOA

MOA [59, 61] exploits the Markovianity or local Markov property of Markov networks. It has a workflow simpler than other global Markov property based EDAs. Furthermore,

in addition to gain in efficiency, it avoids the numerical operations associated to the computation of potentials or Kikuchi approximations [50], which may also represent gains in model accuracy. MOA workflow is described in Algorithm 4.

Algorithm 4: **Markovianity based optimization algorithm**

Set $t \Leftarrow 0$. Generate M points randomly
do {
    Evaluate the points using the fitness function.
    Select a set D_t^S of $N \leq M$ points according to a selection method.
    Estimate the structure of a Markov network from D_t^S.
    Estimate the local Markov conditional probabilities, $p(x_i|N_i)$, for each variable X_i as defined by the undirected structure.
    Generate M new points sampling from the Markov network.
    $t \Leftarrow t + 1$
} **until** Termination criteria are met

Different structural learning algorithms can be applied to learn Markov networks. MOA learns an approximation of the probability using a mutual information based approach. To learn the Markov network structure, first the mutual information for each pair of variables in computed. Then, an edge between two variables is created if the mutual information between them is higher than the given threshold. Here we compute the threshold, TR, as $TR = avg(MI) * sig$, where $avg(MI)$ is the average of the elements of the mutual information matrix and sig is the significance parameter. Notice that the approach to use a threshold on the mutual information is similar to the use of the G-statistics to determine the edges in the graph in MN-FDA^G. In this way, the roles played by α in MN-FDA^G and sig in MOA are similar. Following [61], we set $sig = 1.5$. If the number of neighbors of a variable is higher than an allowed maximum number, N_{neigh}, then only the N_{neigh} neighbors that have the highest mutual information are kept. In our experiments, we used different N_{neigh} values.

Algorithm 5: **Gibbs sampler used by MOA**

Set $t \Leftarrow 0$.
Generate a solution $x = (x_1, x_2, .., x_n)$ at random.
do {
    Choose a variable x_i from $\mathbf{x}$ at random.
    Sample x_i from $p(x_i|N_i) = \frac{e^{p(x_i,N_i)/T}}{\sum_{x_i'} e^{p(x_i',N_i)/T}}$.
    $t \Leftarrow t + 1$
} **until** $t = r$

MOA uses a Gibbs sampler [14] as shown in Algorithm 5, where $r = n \times ln(n) \times IT$, and IT, the *iteration coefficient*, is set to 4. T is the *temperature coefficient* that controls the convergence of the Gibbs probability distribution. Increasing T makes the distribution close to being uniform, and decreasing T converges it to extremal values. In the particular case of binary representation, the probability of $x_i = 1$ given the value of its neighbors N_i is $p(x_i = 1|N_i) = \frac{e^{p(x_i=1,N_i)/T}}{e^{p(x_i=1,N_i)/T}+e^{p(x_i=0,N_i)/T}}$. In [61], a linear schedule for the temperature was set as $T = \frac{1}{g \times CR}$, where g is the current generation of MOA and CR is the *cooling rate* parameter and that was set to $CR = 0.5$. Note that each execution of the Gibbs sampler creates a single solution. Multiple executions of Gibbs sampler should be done in order to create the population of solutions. For more details on MOA see [61].

3 Factor graphs and message passing algorithms

3.1 Factor graphs

A *factor graph* [24] is a bipartite graph that can serve to represent the factorized structure of a distribution. It has two types of nodes: variable nodes (represented as a circle), and factor nodes (represented as a square). In the graphs, factor nodes are named by capital letters starting from A, and variable nodes by numbers starting from 1. Variable nodes are indexed with letters starting with i, and factor nodes with letters starting with a. The existence of an edge connecting variable node i to factor node a means that x_i is an argument of function f_a in the referred factorization.

Figure 2 shows three factor graphs constructed from different additively decomposable functions (ADFs). Notice that the same variable can be in more that one factor and that the same function can be applied to different factors (e.g., in the last sample function h_1 is used to evaluate factors f_a and f_b).

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 $\mathbb{R}^+$. 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

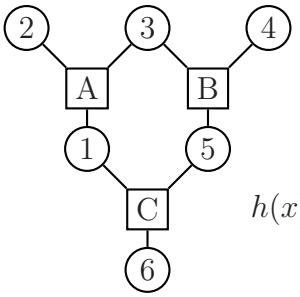
$$p_f(\mathbf{x}) = \frac{1}{Z} \prod_{a=1}^{m} f_a(\mathbf{x}_{S_a}) \tag{3}$$

where $Z = \sum_{\mathbf{x}} \prod_{a=1}^{m} f_a(\mathbf{x}_{S_a})$ is the partition function.

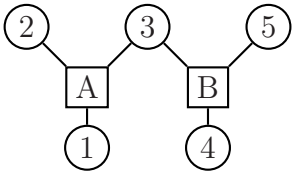
3.2 Belief propagation

Belief propagation (BP) is an inference method used to calculate the marginal probabilities of a probabilistic model. It is usually applied after some evidence about the observed states of the variables has been incorporated to the model. A key characteristic of the BP algorithm is that the inference is done using message-passing between nodes. Each node sends and receives messages until a stable situation is reached. Messages, locally calculated by each node, comprise statistical information concerning neighbor nodes.

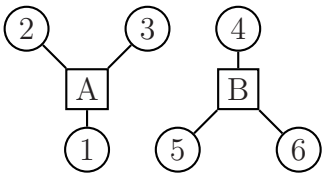
When using BP with factor graphs, two kind of messages are identified: messages $n_{i \to a}(x_i)$ sent from a variable node i to a factor node a, and messages $m_{a \to i}(x_i)$ sent from a factor node a to a variable node i. Note that a message is sent for every value of each variable X_i.



$$h(x) = h_1(x_1, x_2, x_3) + h_2(x_3, x_4, x_5) + h_3(x_1, x_5, x_6)$$



$$h(x) = h_1(x_1, x_2, x_3) + h_2(x_3, x_4, x_5)$$



$$h(x) = h_1(x_1, x_2, x_3) + h_1(x_4, x_5, x_6)$$

Figure 2: Three factor graphs constructed from different additive decomposable functions.

These messages are updated according to the following rules:

$$n_{i\to a}(x_i) := \prod_{c \epsilon N(i)\backslash a} m_{c\to i}(x_i) \tag{4}$$

$$m_{a\to i}(x_i) := \sum_{\boldsymbol{x_a}\backslash x_i} f_a(\boldsymbol{x_a}) \prod_{j \epsilon N(a)\backslash i} n_{j\to a}(x_j) \tag{5}$$

$$m_{a\to i}(x_i) := \arg\max_{\boldsymbol{x_a}\backslash x_i} \{f_a(\boldsymbol{x_a}) \prod_{j \epsilon N(a)\backslash i} n_{j\to a}(x_j)\} \tag{6}$$

where $N(i)\backslash a$ represents all the neighboring factor nodes of node i excluding node a, and $\sum_{\boldsymbol{x_a}\backslash x_i}$ expresses that the sum is completed taking into account all the possible values that all variables but X_i in $\boldsymbol{X}_a$ can take –while variable X_i takes its x_i value.

Equations 4 and 5 are used when marginal probabilities are looked for (sum-product). By contrast, in order to obtain the most probable configurations (max-product), equations 4 and 6 should be applied.

When the algorithm converges (i.e. messages do not change), marginal functions (sum-product) or max-marginals (max-product) are obtained as the normalized product of all messages received by X_i.

$$g_i(x_i) \propto \prod_{a \epsilon N(i)} m_{a\to i}(x_i) \tag{7}$$

Regarding the max-product approach, when the algorithm converges to the most probable value, each variable in the optimal solution is assigned the value given by the

configuration with the highest probability at each max-marginal. Some theoretical results on BP and modifications for maximization can be found in [71].

3.2.1 Probability damping for BP

An important drawback of BP and other message passing algorithms is related to its convergence properties. Although there are results that give necessary conditions for convergence [18], the algorithm is not guaranteed to converge for every problem. One way to improve BP results is to use message damping [47, 65].

In linear message damping, Equations (5) and (6) are respectively replaced by Equations (8) and (9)

$$m_{a\to i}(x_i) := \gamma\, m_{a\to i}(x_i) + (1-\gamma)\left[\sum_{\boldsymbol{x_a}\setminus x_i} f_a(\boldsymbol{x_a}) \prod_{j\epsilon N(a)\setminus i} n_{j\to a}(x_j)\right] \tag{8}$$

$$m_{a\to i}(x_i) := \gamma\, m_{a\to i}(x_i) + (1-\gamma)\left[\arg\max_{\boldsymbol{x_a}\setminus x_i}\{f_a(\boldsymbol{x_a}) \prod_{j\epsilon N(a)\setminus i} n_{j\to a}(x_j)\}\right] \tag{9}$$

where γ is the damping parameter. Increasing γ increases the weight of the old messages in evaluating the new ones, and $\gamma = 1$ implies no evolution at all. A geometric average can be used instead of the linear damping.

4 BP with decimation for computing MAP

Decimation strategies [23] are used to solve constraint satisfaction problems (CSPs). They work by repeatedly fixing variable values and simplifying the constraints without reconsidering earlier decisions. They can be seen as a divide an conquer approach in which the problem is split and simplified at each step. In [37], the survey propagation algorithm, a method combining message-passing and decimation, was introduced. This propagation algorithm has been extensively applied to the solution of hard 3-SAT problems with millions of variables. More recently, Kroc et al. [23] have shown that the combination of BP with decimation is also able to solve CSPs like 3-SAT.

In the context of EDAs, we will apply a decimation strategy to find an approximate solution to the MAP configuration of a factor graph representing a probability distribution. Different variants of the combination of decimation and inference algorithms for MAP have been conceived. From now on we call to any variant of this type of strategies as a *BP-decimation* algorithm. We first describe one of the simplest BP-decimation variants and give an example of how it works. Subsequently, we present the more general implementation used in this paper.

At each step of the simple BP-decimation algorithm the idea is to identify the variable X_i that is almost certainty to have a given value in the problem's most probable configuration. First, max-BP is run. Then, the variable X_i with the largest likelihood on one of its configurations is identified and fixed to its most probable value. In the next step, the variable node X_i is removed from the factor graph and all factors that contains X_i are reduced by eliminating the tuples which force X_i to take a value different from

the one fixed. The cycle is repeated until all variables are fixed. Notice that at each step the structure of the graph can become simpler and split into smaller subgraphs.

As an example we take the first factor graph shown in Figure 2. This model has 6 variables and 3 factors. We will assume these variables are binary and after the first application of max-BP the vector of probabilities $p(X_i = 1)\ i \in \{1, \ldots, 6\}$ is $(0.67, 0.35, 0.92, 0.3, 0.48, 0.2)$. We fix $X_3 = 1$. Consequently, the decimated factors f_a and f_b are now respectively defined on variables (X_1, X_2) and (X_4, X_5). Let us suppose that after the next application of max-BP variable X_5 is selected and $X_5 = 0$. As a result of the following simplification step, X_4 will be fixed (let us suppose as $X_4 = 0$). The remaining factor graph has only two factors, f_a and f_c, of two variables each, (X_1, X_2) and (X_1, X_6). Then max-BP is applied again and variable X_1 is identified and fixed to $X_1 = 1$. As a result of the subsequent simplification, the remaining variables X_2 and X_6 are also fixed and the vector output by the algorithm is $x = (1, 0, 1, 0, 0, 1)$

4.1 Variants of BP-decimation algorithms

Algorithm 6 shows the main steps of the BP-decimation scheme used in this paper. It is based on the *decmap* program included in the libDAI implementation [38] (see Section 7.1 for more details on the characteristics of the used implementation). The algorithm receives a factor graph where some evidence may have been added. A single evidence consists of a joint *observed* configuration for a subset of variables. Multiple evidence can be added to the factor graph prior to the application of the BP-decimation algorithm.

Algorithm 6: **BP-decimation algorithm**

```
1 do {
2     Run an approximate inference algorithm.
3     Clamp the factor with the lowest entropy to its most probable state.
4     Simplify factor graph.
5 } until All variables are fixed
```

In the first step of Algorithm 6 an inference method is applied to compute the MAP. In this paper we use BP with damped messages. BP parameters used in our experiments are presented in Section 7.1. The next step determines which variables should be fixed and to what values. All the variables in the factor with the lowest entropy are fixed to the most probable configuration (they are said to be clamped). Finally, the factor graph is simplified and if there are still variables that have not been fixed the loop is continued.

There are several issues that influence the behavior of the BP-decimation algorithm and that should be taken into account for the kind of applications we envision. Among these issues are the following:

- Choice of the inference algorithm.
- Damping parameter.

- Stop criteria for the inference algorithm (e.g., maximum number of iterations or error tolerance).
- Criterion for variables clamping.

The reason that makes these issues relevant for their application in EDAs is that they allow to balance the accuracy of the MAP result with the computational cost of finding it. Since the MAP configuration will not necessarily be the optimal solution to the optimization problem, some level of inaccuracy can be accepted in the output of the BP-decimation algorithm in exchange for important gains in computational time. Furthermore, recent results [45] in the application of BP-decimation algorithms to CSP show that a small number of sequential updates of the messages per decimation step can be sufficient to obtain relevant information about the MAP probabilities, at least for the topologies and BP equations considered in [45]. Small number of iterations and high error tolerance rates may not be critically harmful for the use of BP-decimation algorithms as methods to generate the new population in EDAs.

The criterion for variables clamping also influences the outcome of the algorithm. The heuristic argument for using greedy decimation as in Algorithm 6 is that in doing so a mistake in the variables assignment is less likely to happen. However, other strategies are possible. For example, to fix some variable randomly chosen from the set of the c percent variables with the largest bias [70], or fitting a set of variables whose configuration probability is above a given threshold [45]. More sophisticated methods for selecting the variables to clamp are also available [9]. From the EDA point of view, the way clamping is done can influence the diversity of the generated solutions. Adding some randomness to this step seems necessary to obtain diversity solutions, unless some different evidence is entered to the factor graph model each time BP-decimation is invoked.

5 New methods for generating new solutions in EDAs based on Markov networks

We propose three different ways for using the information contained in the Markov networks for generating new solutions. In every case, the methods attempt to incorporate the *global* information contained in the Markov network to the new generated set of solutions. The methods introduced are:

1. Adding the most probable configuration.
2. Using Markov networks as template for crossover.
3. Using the most probable configuration as a template for crossover.

5.1 Adding the most probable configuration

In Algorithm 7, the MAP is generated using BP-decimation and it is passed to the new generation together with the elitist solutions E, and the $N - |E| - 1$ solutions generated from the probabilistic model using GS, PLS, or other. This is essentially the same idea used in previous EDA work on the use of the most probable configurations [33, 35]. However, there are also some differences:

- The factor graph is constructed from a Markov network that will represent two different types of structural information. Reduced set of cliques, for MN-FDA^G, and neighborhood based cliques, for MOA.
- Three different methods are used to compute the most probable configuration: junction-tree-based method for computing exact MAP, loopy belief propagation (LBP) for computing approximate MAP, and decimation-based LBP for computing approximate MAP.
- An archive is used as a memory of the EDA to avoid adding MAPs already found in previous generations. The archive stores at most one new MAP in each generation.

Algorithm 7: **Insert-MAP**

```
1  Learn the Markov network model MN.
2  Generate the set S of N − |E| − 1 solutions according to generation method.
3  Compute the factor graph FG.
4  Generate a MAP solution from FG.
5  Form the new population with S, E, and the MAP solution.
```

The effect of the Insert-MAP strategy will depend on the choice of the inference algorithm. Very often for MOA the application of an exact inference procedure is infeasible due to the dimension of the junction tree. In these situations no MAP solution is added to the population.

One limitation of the Insert-MAP approach is that still $N - |E| - 1$ solutions have to be generated using other inference methods, which can be costly when GS is applied.

5.2 Using Markov networks as template for crossover

One straightforward way to use the Markov network structure for generated new solutions is designing a crossover operator that respects the dependencies of the problem captured within the cliques. We use the Markov network as a template for exchanging partial configurations between solutions. Partial configurations corresponding to cliques of the Markov network are exchanged between pairs of solutions. The pseudocode for this approach is shown in Algorithm 8.

Previous results have shown that cliques can capture relevant, and usually interpretable [53], partial information about the problem. By exchanging these subsolutions we combine the global information contained in the probabilistic model with local information contained in the solutions.

Algorithm 8: **Markov-network template crossover**

```
1 Learn the Markov network model MN.
2 for i = 1 to N − |E|
3     Randomly select two solutions x^i, x^j from the selected set.
4     For each maximal clique S in the Markov network, with probability p_c.
5         Exchange configurations x^i_S and x^j_S.
```

To avoid excessively disruptive recombination we set $p_c = \frac{1}{NCliques}$ where $NCliques$ is the number of maximal cliques in the models.

5.3 Using the most probable configuration to bias the generated population

In Algorithm 9, the MAP is used as a template for recombining the information of the solutions in the current population with the global information contained in the Markov model. Recombination guarantees that partial configurations that coincide with the MAP will remain in the next population. In Algorithm 9, inference is applied only once in each generation and PLS or Gibbs sampling are not applied. It has been acknowledged [73] that operators that generate offspring through combination of global statistical information and the location information of solutions found so far (e.g., guided mutation) can improve search efficiency.

Algorithm 9: **Template-MAP**

```
1 Learn the Markov network model MN.
2 Compute the factor graph FG.
3 Obtain MAP solution from FG.
4 for i = 1 to N − |E|
5     Apply uniform crossover between solution x^i and MAP solution.
```

6 Related work

The results presented in this paper are related to previous works that have independently addressed several of the issues we cover. In this section, we review some of this previous work, identifying the links with our proposal.

6.1 Model-based and MAP-based template crossovers

Many researchers have investigated the question of linkage learning in EDAs (see [8] for a survey). EDAs can be included in the class of EAs that apply virtual linkage,

i.e., algorithms that use graph, groupings, matrices, pointers, or other data structures that control the subsequent pairing or clustering organization of decision variables [8]. By means of the strategies used for generating new solutions, EDAs explicitly use the models to create new solutions. The model-based template crossover introduced in this paper also uses the probabilistic model to store information about the dependencies between the variables. However, instead of applying statistical techniques to sample from it, the crossover operator itself uses the structure in a way similar to the way multiple point recombination is applied in GAs. The benefit of this approach is that disruption can be minimized and the pace of recombination can be slowed.

The use of selected individuals to bias the population in a given direction has been also used in EDAs. In the edge histogram based sampling algorithm with template (EHBSA/WT) [69], a template individual is chosen, ($k > 1$) cut points are applied to the template randomly, one of the produced segments is randomly chosen, and then variables are sampled from the segment. The guided mutation [72] also uses a probabilistic model but during the mutation step an existing solution is mutated guided by a probability matrix $P = (p_{ij})_{n \times n}$ and a positive parameter $\delta < 1$. The main difference between the MAP-based template crossover and previous approaches is in the way the individual selected for biasing the population is constructed. The MAP may or not be the best solution in the current population but it condenses information about the best partial configurations contained in the Markov network. Therefore, the MAP could also be used in other contexts beyond EDAs like the recently introduced optimal mixing evolutionary algorithm (OMEA) [66, 67], and related variants, where the currently known best solution is used as a donor for crossover during sampling. Similarly, it could be used to incorporate global information extracted from the probabilistic models to particle swarm optimization algorithms [10].

6.2 Algorithms used for learning the models and generating new solutions from them

MN-FDA applies the G-test on the information computed using the mutual information. It can be seen as a hybrid of a learning algorithm based on the use of metrics, in this case the mutual information, and a learning method based on statistical tests. Recently, Bosman and Thierens [5] have proposed the application of the likelihood-ratio test and discussed its relationship with the mutual information. They consider the application of this test in the context of the linkage tree algorithm and evaluate its application on the optimization of binary problems. Instead of using a threshold value for the likelihood ratio test as done by MN-FDA^G, they apply a threshold on the mutual information, just as MOA [59] does. In [5], different approaches to compute the threshold are investigated.

Although GS has been the most applied method to sample early EDAs that used PGMs with cycles [2, 12, 39, 50], and more recent EDAs based on Markov networks [7, 30, 39, 57], other strategies like blocked GS have been applied with mixed results [46, 55]. There has been also considerable interest in the application of methods that compute the MAPs to generate new solutions from probabilistic models in EDAs. Initial applications of MAPs in EDAs were proposed for univariate models in [64], for which the application of BP is not required, and pairwise Markov networks [51]. These results were extended to cover the more complex estimation of Bayesian network algorithms (EBNAs)

[34, 35].

Using the BP components and parameters as a way to influence the EDA dynamics was discussed in [33]. Two critical questions for the application of MAPs in EDAs are to have a flexible implementation and to be scalable. Flexibility is needed for allowing the manipulation of the MAP components (i.e. initializations methods, scheduling policies, and stopping criteria) according to the search goals, and for the combination of these components with the probabilistic model learning and generation steps used by EDAs. Scalability is essential for addressing hard optimization problems with many variables in a reasonable time. These issues were analyzed in [33] where a flexible parallel framework for BP over factor graphs was introduced.

BP algorithms have been also used for obtaining higher order consistent marginal probabilities in multivariate EDAs [40], as well as the most probable configurations of the model [22, 42]. In these cases, the structure of the problem is known a priori. In [28] BP was applied to perform a substructural local search, looking for the solution with the highest fitness value instead of that with the highest probability.

7 Experiments

In this section we evaluate the behavior of the techniques introduced in the previous sections. First, we provide the implementation details and introduce the functions and experimental benchmarks. Then the numerical results are presented and discussed.

7.1 Algorithms and implementation

We have used the implementation of the inference algorithms contained in the libDAI software package [38], which is a free and open source library that supports various approximations of the partition sum, marginal probability distributions and maximum probability states. In every generation of the EDA, the Markov model is transformed to a factor graph on which the different inference algorithms are applied. In this paper we use three different inference algorithms:

1. Exact inference (JT). A junction tree is constructed. BP is applied using Hugin algorithm [3].

2. Approximate inference (LBP). Messages are updated sequentially using a random sequence. Damping is applied with $\gamma = 0.1$.

3. BP-decimation (BPDEC) as described in Algorithm 6 using LBP as the inference algorithm.

7.2 Function and experimental benchmark

To evaluate our algorithms we use two classes of functions. Separable additively decomposable functions (ADFs) defined on discrete non-binary variables and non-separable decomposable binary functions. Separable ADFs can be solved by EDAs based on marginal product factorizations [16, 54]. However, the difficulty of these functions increases with the cardinality of the variables. Non-separable functions are generally hard to solve due

to the arousal of many dependencies between the variables. Complex probabilistic models might be needed to capture these dependencies by EDAs.

7.2.1 Separable ADFs defined on discrete non-binary variables

Up to now, the analysis of the behavior of EDAs has been focused mainly on binary problems. However, to study the robustness of EDAs and confront a general class of applications, an analysis of non-binary problems is also necessary. To this end, we use a deceptive function defined on non-binary variables.

$$f^c_{deceptivek}(\mathbf{x}) = \sum_{i=1}^{\frac{n}{k}} F_{dec}(x_{k\cdot(i-1)+1} + x_{k\cdot(i-1)+2} + \cdots + x_{k\cdot i}, k, c) \tag{10}$$

$$F_{dec}(x_1, \ldots, x_k, k, c) = \begin{cases} k\cdot(c-1), & for \sum_{i=1}^{k} x_i = k\cdot(c-1) \\ k\cdot(c-1) - \sum_{i=1}^{k} x_i - 1 & otherwise \end{cases} \tag{11}$$

The general deceptive function $f^c_{deceptivek}(\mathbf{x})$ of order k [56] is formed as an additive function composed by the function $F_{dec}(x_1, \ldots, x_k, k, c)$ evaluated on substrings of size k and cardinality c, i.e. $x_i \in \{0, \ldots, c-1\}$. This function is a generalization of the binary $f_{deceptivek}(\mathbf{x})$ to variables with non-binary values.

7.2.2 Non separable ADFs

The generalized Ising model (12) is described by the energy functional (Hamiltonian) where L is the set of sites called a lattice. Each spin variable σ_i at site $i \in L$ either takes the value 1 or value -1. A specific choice of values for the spin variables is called a configuration. The constants J_{ij} are the interaction coefficients. In our experiments we take $h_i = 0,\ \forall i \in L$. The ground state is the configuration with minimum energy.

$$H = -\sum_{i<j\in L} J_{ij}\sigma_i\sigma_j - \sum_{i\in L} h_i\sigma_i \tag{12}$$

We use Ising instances with $L \in \{6, 8, 10\}$. Therefore the number of variables are in $\{36, 64, 100\}$. Four different instances are used for each dimension.

7.2.3 Experimental benchmark

We compare the algorithms in terms of their critical population size to reach the optimum and the average number of evaluations needed. The critical population size is computed as the minimum population size needed to reach the optimum in l successive experiments and the average number of evaluations is computed from a maximum of r independent computations of the critical population size (less than r if it was not possible to find a critical population size in all the runs). For function $f^c_{deceptivek}(\mathbf{x})$, we set $l = 30$ and $r = 30$. For the Ising model, $l = 5$ and $r = 20$. The population size is started at $N = 32$ and doubled if the algorithm does not converge in the l experiments until a maximum of $N = 131072$. The maximum number of generations was $g = 40$ in all the experiments.

Truncation selection, $T = 0.5$, is applied together with best elitism in which the complete selected population is passed to the next generation.

There are two factors that define the different variants we compare:

- EDAs:
  1. MN-FDA^G
  2. MOA
- Strategy for generating new solutions:

  S1. Insert-MAP

  S2. Template-MAP

  S3. Insert-MAP + Template-MAP

  S4. Insert-MAP + Markov-network template crossover

We also evaluated the influence that using different inference methods has in the behavior of the EDA. Four inference methods were compared:

1. NoMAP: No MAP is computed. Instead a solution generated using PLS, GS, or template crossover is added to the population.
2. Exact: Exact inference based on a junction tree
3. BP: Belief propagation
4. Dec-BP: Decimation BP

The choice of MN-FDA^G or MOA influences the type of undirected model that is learned from data but also the strategy used for generating new solutions. When Insert-MAP is combined with NoMAP it means that no MAP is computed and therefore the algorithm used by MN-FDA^G or MOA to generate new solutions is applied. In total, we compare $2 \times 4 \times 4 - 2 = 30$ variants of the algorithms. Notice that when inference is not applied, the combinations (MN-FDA^G,Insert-MAP,NoMAP), (MN-FDA^G,Template-MAP,NoMAP), and (MN-FDA^G,Insert-MAP+Template-MAP,NoMAP) are the same, similarly for (MOA,Insert-MAP,NoMAP), (MOA,Template-MAP,NoMAP), and (MOA,Insert-MAP+Template-MAP,NoMAP).

7.3 Numerical results

7.3.1 Deceptive discrete functions

Although binary deceptive functions are useful to investigate EDAs, many practical problems have a non-binary discrete representation and it is important to determine how are the algorithms influenced by this dimension of difficulty. We investigate the performance of MN-FDA^G and MOA using function $f^c_{deceptivek}(\mathbf{x})$, $k \in \{3, 4, 5\}$, $c \in \{2, 3, 4, 5\}$. We are interested in evaluating the behavior of the EDAs when the size of the definition sets and the number of values of each variable are increased.

Table 1: Results of the different strategies for generating new solutions and inference methods for $f^c_{deceptivek}(\mathbf{x})$, $k \in \{3,4,5\}$, $c \in \{2,3,4,5\}$. Number of runs that achieved the optimum.

Problems	Gen. Method	MN-FDA^G				MOA			
		NoMAP	Exact	BP	Dec-BP	NoMAP	Exact	BP	Dec-BP
$n = 30, k = 3$	$s1$	90	120	120	120	120	120	120	120
	$s2$	90	0	0	0	120	0	0	0
	$s3$	90	120	119	119	120	117	113	118
	$s4$	120	120	120	120	120	120	120	120
$n = 32, k = 4$	$s1$	60	90	90	90	56	57	57	58
	$s2$	60	0	0	0	56	0	0	0
	$s3$	60	85	88	88	54	30	30	30
	$s4$	30	30	30	30	81	78	80	81
$n = 30, k = 5$	$s1$	30	60	60	60	30	30	30	30
	$s2$	30	0	0	0	30	0	0	0
	$s3$	30	60	60	60	30	30	30	30
	$s4$	30	30	30	30	30	30	30	30

Table 1 shows the number of successful runs of different EDAs variants when the definition sets of the functions are increased from $k = 3$ to $k = 5$. Each cell of the table groups the number of successful runs for 4 different cardinality values $c \in \{2,3,4,5\}$. Therefore the maximum number of successful runs is 120. The average number of function evaluations for each combination of k and c are shown in Figures 3, 4, and 5.

A number of observations can be made from the analysis of Table 1:

- As the size of the definition sets is increased the results of all algorithms deteriorate.
- The best strategy for generating new solutions for MN-FDA^G is Insert-MAP with no differences between exact and approximate inference algorithms (Figures 3, 4, and 5).
- The best generation strategy for MOA is Insert-MAP+Markov-network template crossover, although the addition of the MAP configuration does not improve the results of the pure Markov-network template crossover.
- MAP-Template is the worst generation strategy for all the problems.
- Considering together the successful rate and the average number of evaluations, the best overall results are achieved by MN-FDA^G with Insert-MAP.

Using the data from the same experiment, we also compare the algorithms when the cardinality of the variables is increased. The results for function $f^c_{deceptivek}(\mathbf{x})$ are shown in Table 2. It can be seen that, as the variable cardinality is increased, all algorithms dramatically decrease their performance. The combination of a high cardinality and a high definition set is particularly harmful to all the algorithms. For $c = 5$, only functions with $k = 3$ are solved. Observations described below are also confirmed from the analysis of Table 2.

Table 2: Results of the different strategies for generating new solutions and inference methods for $f^c_{deceptivek}(\mathbf{x})$, $n = 30$, $k \in \{3, 4, 5\}$, $c \in \{2, 3, 4, 5\}$. Number of runs that achieved the optimum.

Problems	Gen. method	MN-FDA^G				MOA			
		NoMAP	Exact	BP	Dec-BP	NoMAP	Exact	BP	Dec-BP
$c = 2$	$s1$	90	90	90	90	90	90	90	90
	$s2$	90	0	0	0	90	0	0	0
	$s3$	90	90	90	90	90	90	90	90
	$s4$	90	90	90	90	90	90	90	90
$c = 3$	$s1$	60	90	90	90	56	57	57	58
	$s2$	60	0	0	0	56	0	0	0
	$s3$	60	90	90	90	54	30	30	30
	$s4$	30	30	30	30	60	60	60	60
$c = 4$	$s1$	30	60	60	60	30	30	30	30
	$s2$	30	0	0	0	30	0	0	0
	$s3$	30	55	58	58	30	30	30	30
	$s4$	30	30	30	30	51	48	50	51
$c = 5$	$s1$	0	30	30	30	30	30	30	30
	$s2$	0	0	0	0	30	0	0	0
	$s3$	0	30	29	29	30	27	23	28
	$s4$	30	30	30	30	30	30	30	30

7.3.2 Ising instances

The Ising problem allows to evaluate the EDAs in a problem with multiple, loopy dependencies, where the definition functions are different among them. Each Ising instance corresponds in fact to a different fitness function. The results of the algorithms are summarized in Table 3 where we show the number of times the algorithms were able to solve the problem in 5 consecutive runs. Each cell of the table adds the number of successful runs for the 20 experiments conducted for each of the 4 instances. Solving the problems in every run for every instance means that the cell entry is $4 \times 20 = 80$. The average number of evaluations needed to solve the problem when a critical population size is found, is shown in Figures 6, 7, and 8.

A number of observations can be made from the analysis of Table 3:

- Insert-MAP and Insert-MAP+Markov-network template crossover can always solve the 4 instances for $n = 36$ although differences arise in the average number of evaluations. MOA using the Insert-MAP+Markov-network template crossover is the most efficient algorithm when $n = 36$.

- The best generation strategy for generating new solutions in MN-FDA^G is Insert-MAP with a clear gain in terms of efficiency when using BP and decimation BP with respect to NoMAP or exact inference (See Figures 6, 7, and 8).

- The best strategy for generating new solutions in MOA is Insert-MAP+Markov-network template crossover, and the addition of the MAP configuration improves

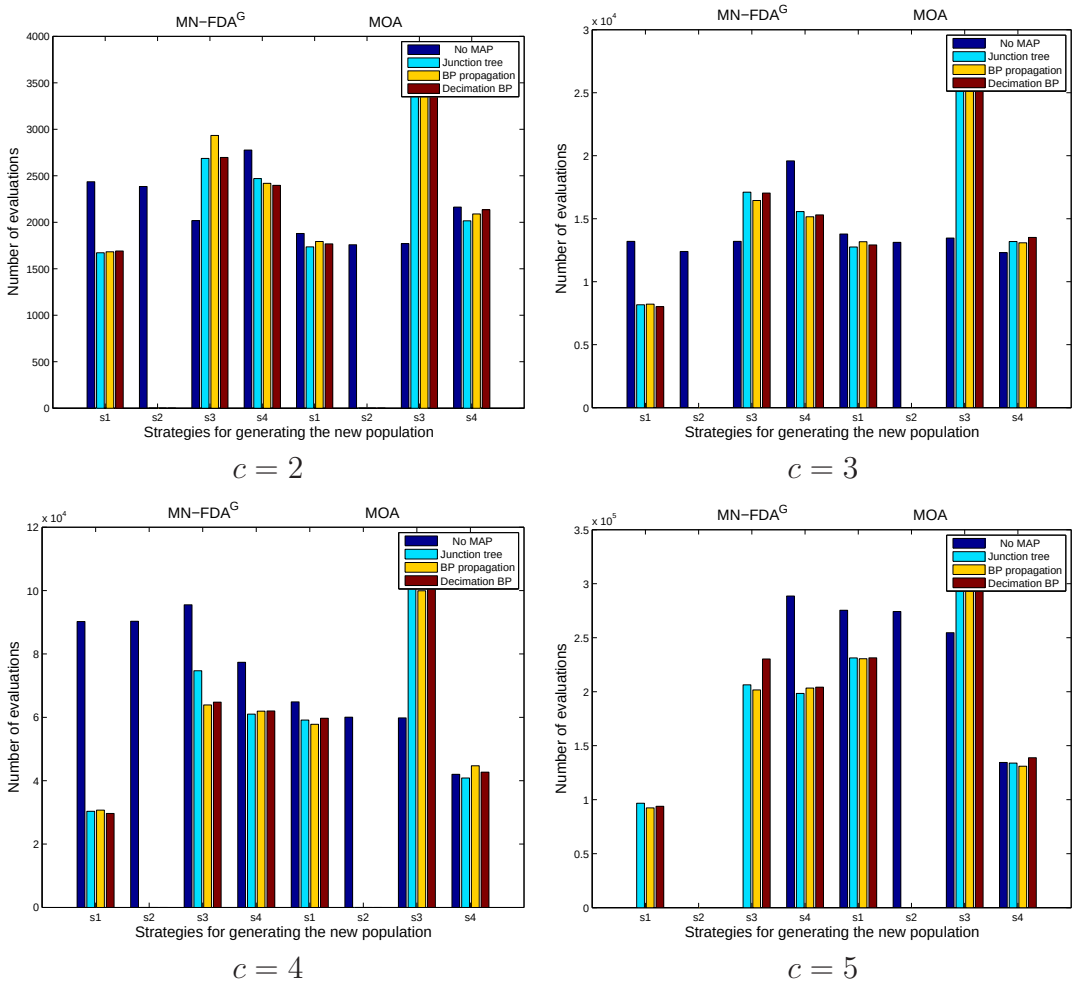


Figure 3: Results of the different strategies for generating new solutions and inference methods for $f^c_{deceptivek}(\mathbf{x})$, $n = 30,\ k = 3,\ c \in \{2, 3, 4, 5\}$.

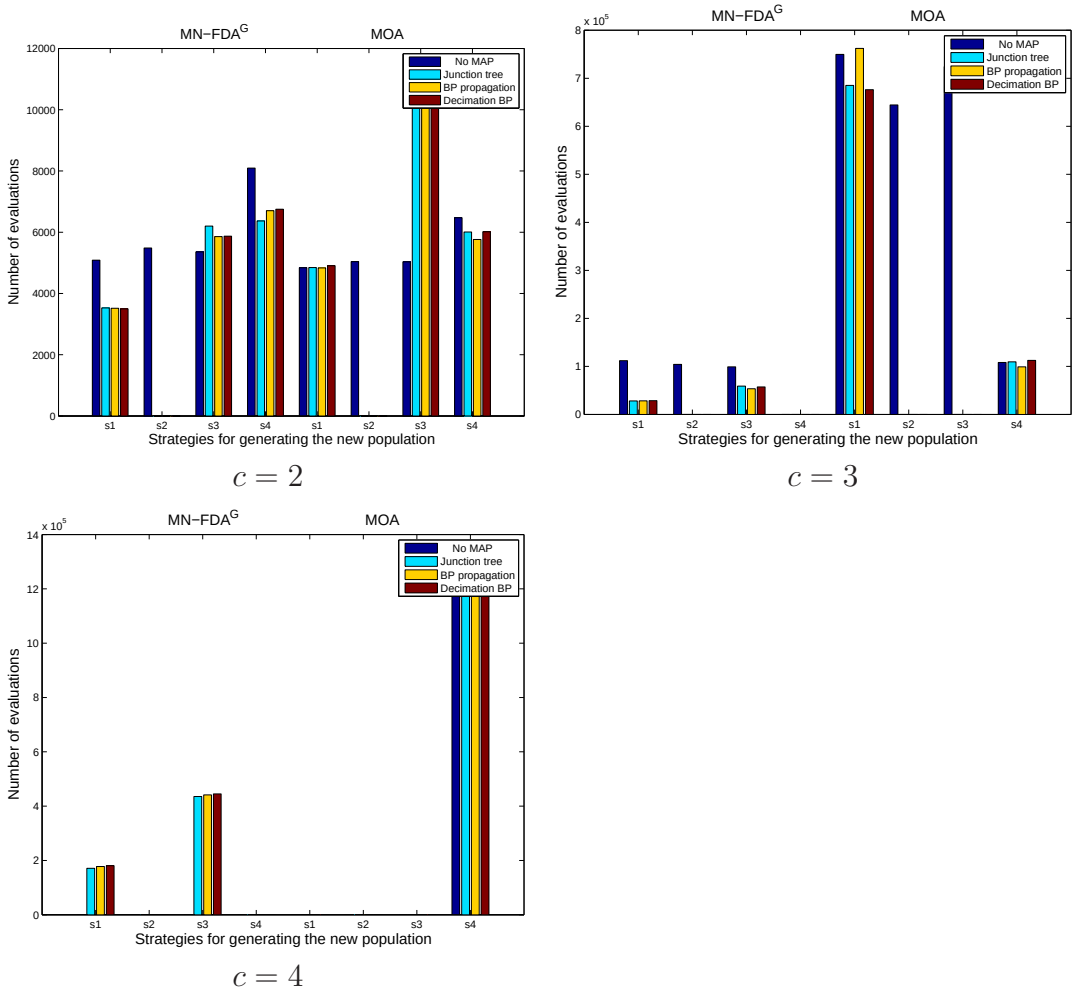
the results of the pure Markov-network template crossover, particularly when BP and Decimation-BP are used (see convergence results for $n = 100$).

- MAP-Template is the worst generation strategy for all the problems although for this strategy different inference methods have a different effect for MN-FDA^G and MOA.

- Considering together the successful rate and the average number of evaluations, the best overall results are achieved by MN-FDA^G with Insert-MAP and Decimation BP.

8 Conclusions

The main contributions presented in this paper are the following:



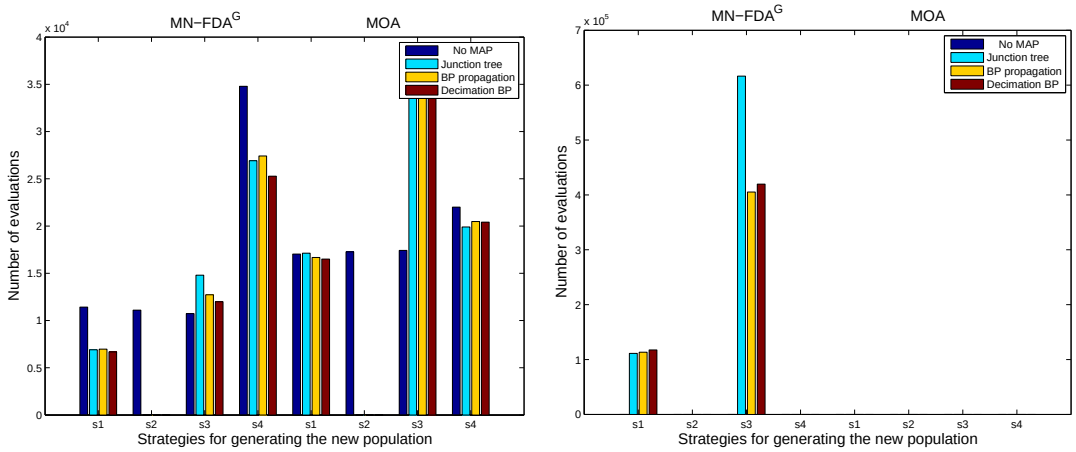
$c = 2$ $c = 3$ $c = 4$

Figure 4: Results of the different strategies for generating new solutions and inference methods for $f^c_{deceptivek}(\mathbf{x})$, $n = 32,\ k = 4,\ c \in \{2, 3, 4\}$. For $c = 5$ none of the algorithms achieved the optimum.

- We have introduced three new strategies generating new solutions in EDAs based on Markov networks, presented different ways they can be combined, and evaluated their effect using two different types of EDAs: MN-FDA^G and MOA. We have empirically shown that some of these strategies can increase the effectiveness of the generation methods usually applied in EDAs and diminish the number of function evaluations needed to find the optimum.

- We have compared exact and approximate inference strategies in the context of EDAs based on Markov networks and we have shown that the choice of the inference procedure can help to reduce the number of evaluations.

- MN-FDA^G, an algorithm that uses the G-test computed from the mutual information has been introduced as an extension to MN-FDA. The statistical test used by MN-FDA^G would allow the algorithm to obtain more accurate estimates from the data for the discrete, non-binary problems considered in our paper. However, we have not compared MN-FDA to MN-FDA^G.



$c = 2$ $c = 3$

Figure 5: Results of the different strategies for generating new solutions and inference methods for $f^c_{deceptivek}(\mathbf{x})$, $n = 30$, $k = 5$, $c \in \{2, 3\}$. For $c \in \{4, 5\}$ none of the algorithms achieved the optimum.

It has been acknowledged [39] that "While the computation of the conditional probabilities and even learning of the Markov network structure can be efficiently done, efficient sampling of Markov networks is still an unsolved problem." The same argument has been exposed or implicitly assumed by other authors from the field. Although we do not claim that the use of model-based template crossover or inference techniques contribute to a more accurate sampling of the Markov network probability distribution, our experiments show that focusing on the MAP of the Markov network can actually improve the optimization results. The lesson to be taken is that the sampling problem could be somewhat relaxed by aiming at directly generating the solutions with highest probabilities (and therefore more likely to have a good fitness) instead of sampling the full distribution using Gibbs sampling. This is what strategies for generating new solutions based on approximate inference methods for MAP are able to do.

Another conclusion of our research is that there exists a close relationship between work done on the combination of inference methods with decimation strategies for CSP and EDAs. It remains an open question to determine whether EDAs, or EAs in general, can be a competitive alternative to BP-Decimation algorithms for CSPs for which the structure is known a priori. In any case, an advantage of EDAs is that they are able to learn an approximation of the problem structure from data. What needs to be determined is the strength of inference methods to deal with the MAP problem when the graphical model is just an approximation of the problem structure. The work presented in this paper could help to identify those problems where EDAs are more likely to provide answers to problems relevant to other research fields.

8.1 Future work

There are a number of lines in which our work could be extended:

- Extend our analysis covering a wider benchmark of objective functions.

Table 3: Results of the different strategies for generating new solutions and inference methods for Ising problems. Number of runs that achieved the optimum.

Problems	Gen. method	MN-FDA^G				MOA			
		NoMAP	Exact	BP	Dec-BP	NoMAP	Exact	BP	Dec-BP
$n = 36$	$s1$	80	80	80	80	80	80	80	80
	$s2$	80	1	59	61	80	49	58	62
	$s3$	80	4	80	80	80	80	80	80
	$s4$	80	80	80	80	80	80	80	80
$n = 64$	$s1$	80	80	80	80	71	80	80	80
	$s2$	80	0	69	59	70	10	10	19
	$s3$	80	0	80	80	73	80	64	80
	$s4$	56	60	80	80	5	80	79	80
$n = 100$	$s1$	0	0	50	41	0	0	0	0
	$s2$	0	0	0	0	0	0	0	0
	$s3$	0	0	50	40	0	0	0	0
	$s4$	0	0	3	8	0	0	23	23

- Evaluate the use of approximate inference algorithms for continuous EDAs.
- Study in more detail the influence of the inference parameters in terms of accuracy and efficiency.
- Evaluate the use of Decimate-BP in EDAs based on Bayesian networks.
- Compare Markov EDAs with the introduced strategies for generating new solutions with inference algorithms (e.g. LBP) applied to the solution of CSPs or problems for which the structure is known.

9 Acknowledgments

This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2010-14931 and Consolider Ingenio 2010 - CSD 2007 - 00018 projects (Spanish Ministry of Science and Innovation) and COMBIOMED network in computational biomedicine (Carlos III Health Institute)

References

[1] P. Abbeel, D. Koller, and A. Y. Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7:1743–1788, 2006.

[2] M. A. Alden. *MARLEDA: Effective Distribution Estimation Through Markov Random Fields*. PhD thesis, Faculty of the Graduate Schoool, University of Texas at Austin, USA, December 2007.

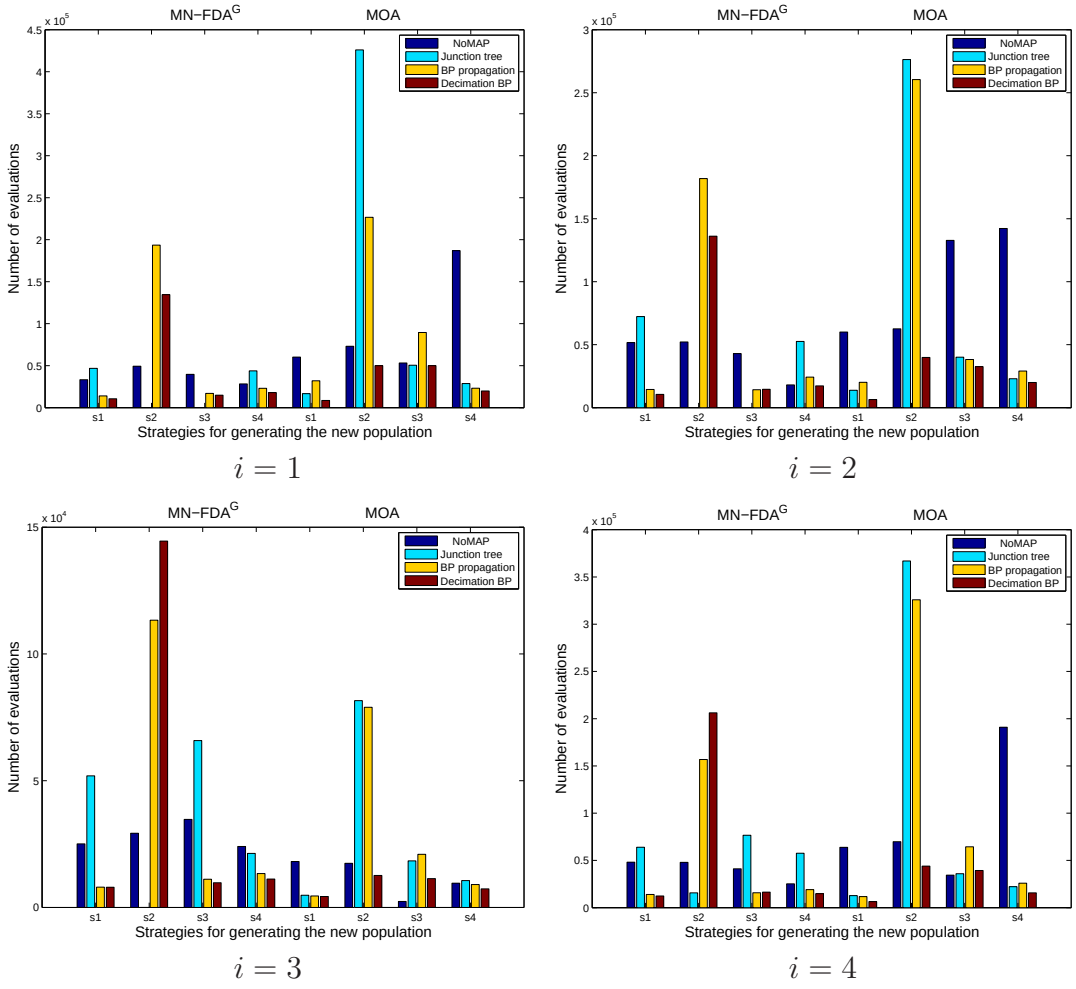


Figure 6: Results of the different strategies for generating new solutions and inference methods for Ising problems.$n = 36$

[3] S. Andersen, K. Olesen, F. Jensen, and F. Jensen. HUGIN– a shell for building Bayesian belief universes for expert systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pages 1080–1085, 1989.

[4] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J. L. Flores, J. A. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, and P. Larrañaga. A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6):doi:10.1186/1756–0381–1–6, 2008.

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

[6] C. Bron and J. Kerbosch. Algorithm 457—finding all cliques of an undirected graph. *Communications of the ACM*, 16(6):575–577, 1973.

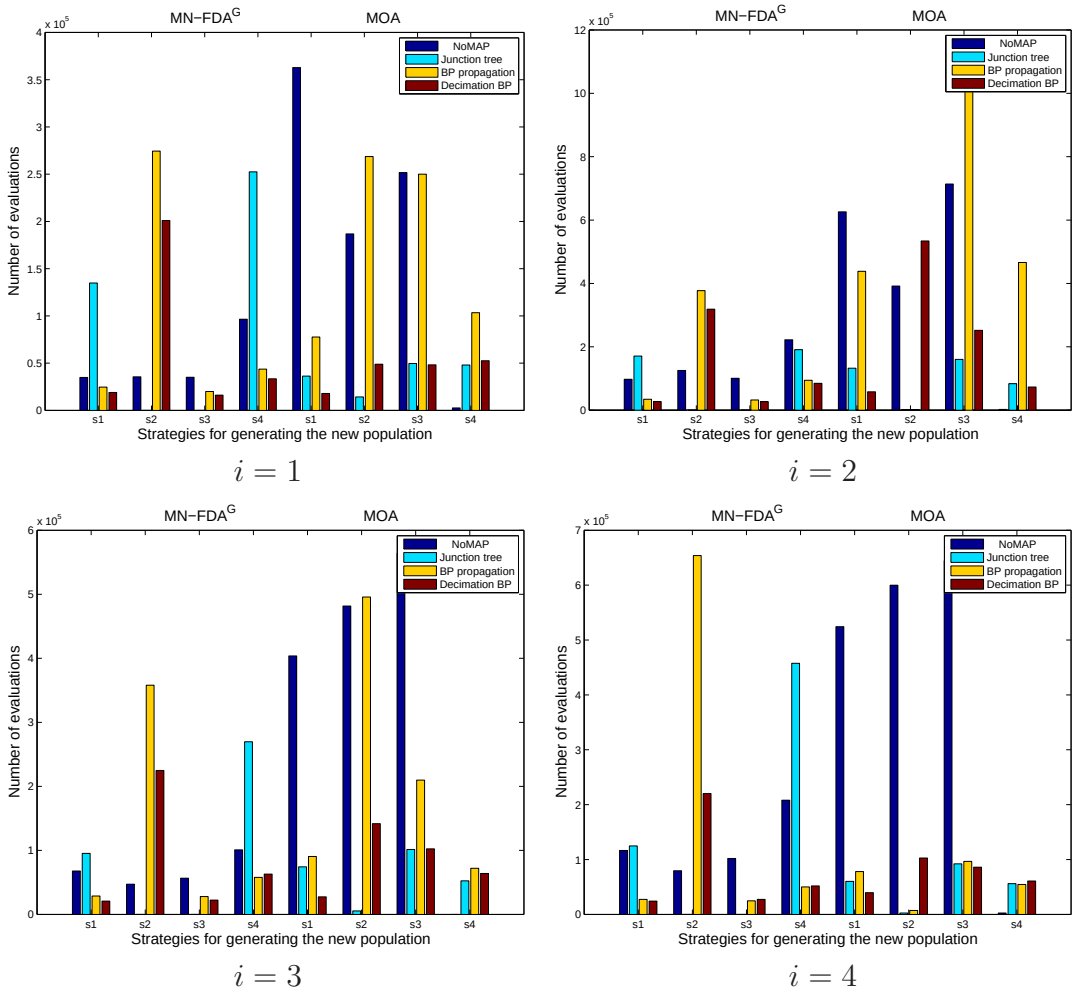


Figure 7: Results of the different strategies for generating new solutions and inference methods for Ising problems. $n = 64$

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

[8] Y. P. Chen, T. L. Yu, K. Sastry, and D. E. Goldberg. A survey of linkage learning techniques in genetic and evolutionary algorithms. IlliGAL Report 2007014, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2007.

[9] F. Eaton and Z. Ghahramani. Choosing a variable to clamp: Approximate inference using conditioned belief propagation. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 145–152, 2009.

[10] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International*

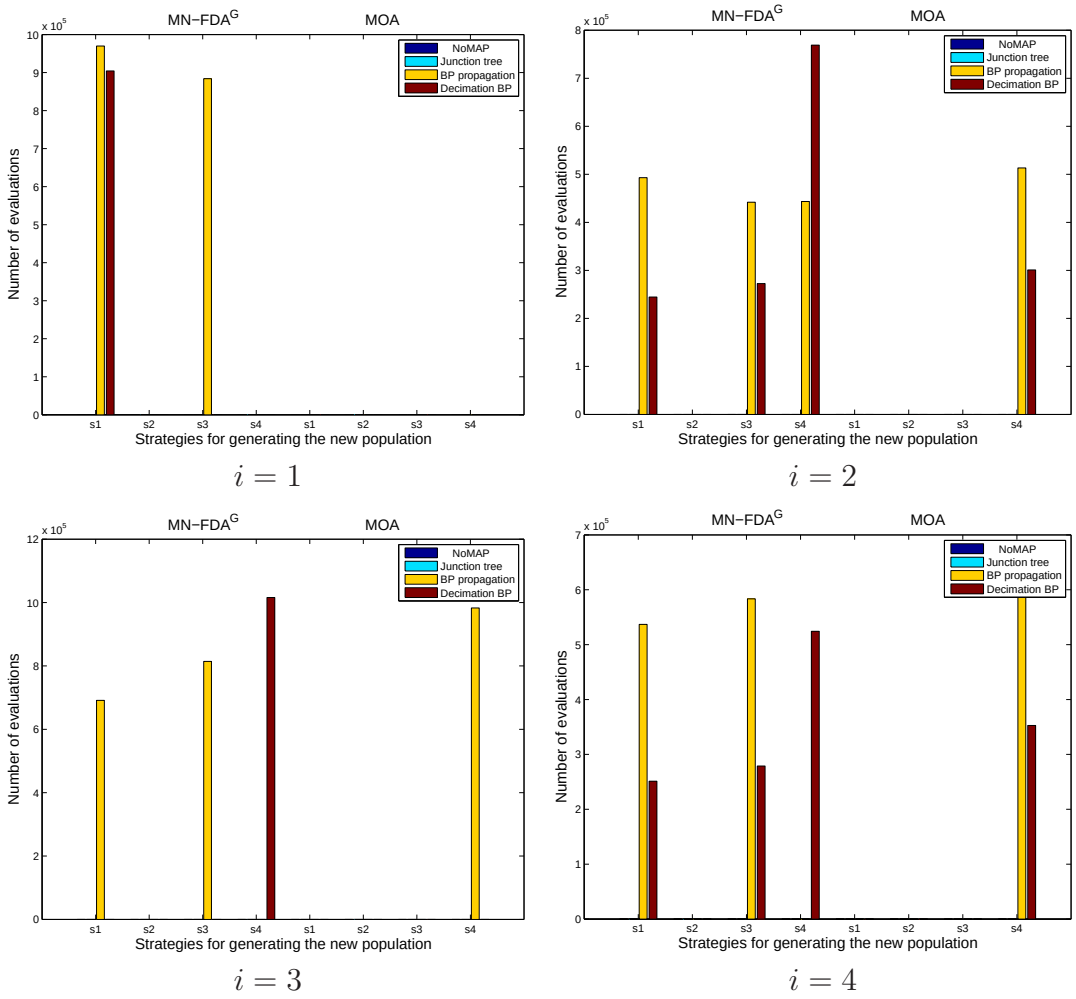


Figure 8: Results of the different strategies for generating new solutions and inference methods for Ising problems.$n = 100$

Symposium on, pages 39–43. IEEE, 1995.

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

[12] J. A. Gámez, J. L. Mateo, and J. M. Puerta. EDNA: Estimation of dependency networks algorithm. In J. Mira and J. R. Álvarez, editors, *Bio-inspired Modeling of Cognitive Tasks, Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007*, volume 4527 of *Lecture Notes in Computer Science*, pages 427–436. Springer, 2007.

[13] A. Gardemann, D. Ohly, M. Fink, N. Katz, H. Tillmanns, F. Hehrlein, W. Haberbosch, et al. Association of the insertion/deletion gene polymorphism of the apolipoprotein B signal peptide with myocardial infarction. *Atherosclerosis*, 141(1):167, 1998.

[14] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741, 1984.

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

[16] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[17] P. Harremoes and G. Tusnady. Information divergence is more χ 2-distributed than the χ 2-statistics. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 533–537. IEEE, 2012.

[18] T. Heskes. On the uniqueness of belief propagation fixed points. *Neural Computation*, 16:2379–2413, 2004.

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

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

[21] R. Höns. *Markov Networks in Evolutionary Computation*, chapter Using Maximum Entropy and Generalized Belief Propagation in Estimation of Distribution Algorithms, pages 175–190. Springer, 2012.

[22] R. Höns, R. Santana, P. Larrañaga, and J. A. Lozano. Optimization by max-propagation using Kikuchi approximations. Technical Report EHU-KZAA-IK-2/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country, November 2007.

[23] L. Kroc, A. Sabharwal, and B. Selman. Message-passing and local heuristics as decimation strategies for satisfiability. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1408–1414. ACM, 2009.

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

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

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

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

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

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

[30] L. Malagó, M. Matteo, and V. Gabriele. Introducing l1-regularized logistic regression in Markov networks based EDAs. In *Proceedings of the 2011 Congress on Evolutionary Computation CEC-2011*, pages 1581–1588. IEEE, 2011.

[31] J. McCall, A. E. I. Brownlee, and S. Shakya. *Markov Networks in Evolutionary Computation*, chapter Applications of distribution estimation using Markov network modelling (DEUM), pages 193–207. Springer, 2012.

[32] J. McDonald. *Handbook of biological statistics*, volume 2. Sparky House Publishing Baltimore, MD, 2009.

[33] A. Mendiburu, R. Santana, E. Bengoetxea, and J. Lozano. A parallel framework for loopy belief propagation. In D. Thierens et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2007*, volume II, pages 2843–2850, London, UK, 2007. ACM Press. Companion material.

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

[35] A. Mendiburu, R. Santana, and J. A. Lozano. *Markov Networks in Evolutionary Computation*, chapter Fast fitness improvements in Estimation of Distribution Algorithms using belief propagation, pages 141–155. Springer, 2012. In press.

[36] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1091, 1953.

[37] M. Mézard, G. Parisi, and R. Zechina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–812, 2002.

[38] J. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *The Journal of Machine Learning Research*, 11:2169–2173, 2010.

[39] H. Mühlenbein. *Markov Networks in Evolutionary Computation*, chapter Convergence Theorems of Estimation of Distribution Algorithms, pages 91–108. Springer, 2012.

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

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

[42] A. Ochoa, R. Höns, M. R. Soto, and H. Mühlenbein. A maximum entropy approach to sampling in EDA- the single connected case. In *Progress in Pattern Recognition, Speech and Image Analysis*, volume 2905 of *Lectures Notes in Computer Science*, pages 683–690. Springer, 2003.

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

[44] M. Pelikan, M. Hauschild, and F. G. Lobo. Introduction to estimation of distribution algorithms. MEDAL Report No. 2012003, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), 2012.

[45] A. Pelizzola, M. Pretti, and J. van Mourik. Palette-colouring: a belief propagation approach. *Journal of Statistical Mechanics: Theory and Experiment*, 2011:P05010, 2011.

[46] E. Ponce de León and E. Díaz. *Markov Networks in Evolutionary Computation*, chapter Adaptive Evolutionary Algorithm based on a Cliqued Gibbs Sampling over Graphical Markov Model Structure, pages 109–123. Springer, 2012.

[47] M. Pretti. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11008+, 2005.

[48] M. Quine and J. Robinson. Efficiencies of chi-square and likelihood ratio goodness-of-fit tests. *The Annals of Statistics*, 13(2):727–742, 1985.

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

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

[51] R. Santana. *Advances in Probabilistic Graphical Models for Optimization and Learning. Applications in Protein Modelling*. PhD thesis, University of the Basque Country, 2006.

[52] R. Santana, P. Larrañaga, and J. A. Lozano. Challenges and open problems in discrete EDAs. Technical Report EHU-KZAA-IK-1/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country, October 2007. Available from http://www.sc.ehu.es/ccwbayes/technical.htm.

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

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

[55] R. Santana and H. Mühlenbein. Blocked stochastic sampling versus Estimation of Distribution Algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC-2002*, volume 2, pages 1390–1395. IEEE press, 2002.

[56] R. Santana, A. Ochoa, and M. R. Soto. Solving problems with integer representation using a tree based factorized distribution algorithm. In *Electronic Proceedings of the First International NAISO Congress on Neuro Fuzzy Technologies*. NAISO Academic Press, 2002.

[57] F. Schlüter and F. Bromberg. Independence-based MAP for Markov networks structure discovery. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence*, 2011. In press.

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

[59] S. Shakya and R. Santana. An EDA based on local Markov property and Gibbs sampling. In M. Keijzer, editor, *Proceedings of the 2008 Genetic and evolutionary computation conference (GECCO)*, pages 475–476, New York, NY, USA, 2008. ACM.

[60] S. Shakya and R. Santana. *Markov Networks in Evolutionary Computation*, chapter A Review of Estimation of Distribution Algorithms and Markov Networks, pages 21–37. Springer, 2012.

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

[62] S. K. Shakya, J. A. McCall, and D. F. Brown. Updating the probability vector using MRF technique for a Univariate EDA. In E. Onaindia and S. Staab, editors, *Proceedings of the Second Starting AI Researchers' Symposium*, pages 15–25, Valencia, Spain, 2004. IOS press.

[63] M. Soto, A. Ochoa, Y. González-Fernández, Y. Milanés, A. Álvarez, D. Carrera, and E. Moreno. *Markov Networks in Evolutionary Computation*, chapter Vine Estimation of Distribution Algorithms with Application to Molecular Docking, pages 175–190. Springer, 2012.

[64] M. R. Soto. *A Single Connected Factorized Distribution Algorithm and its Cost of Evaluation*. PhD thesis, University of Havana, Havana, Cuba, July 2003. In Spanish.

[65] Y. Teh. *Bethe free energy and contrastive divergence approximations for undirected graphical models*. PhD thesis, University of Toronto, 2003.

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

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

[68] A. Tomovic and E. Oakeley. Position dependencies in transcription factor binding sites. *Bioinformatics*, 23(8):933–941, 2007.

[69] S. Tsutsui. Probabilistic model-building genetic algorithms in permutation representation domain using edge histograms. In *Parallel Problem Solving from Nature VIII, Lecture Notes in Computer Science*, volume 2439, pages 224–233. Springer, 2002.

[70] I. Vigan. Understanding belief propagation dialects and their application to optimization problems. Master's thesis, Swiss Federal Institute of Technology. Zurich, August 2009.

[71] M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14:143–166, 2004.

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

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