Estimation of Distribution Algorithms with Kikuchi Approximations

Roberto Santana rsantana@si.ehu.es
Institute of Cybernetics, Mathematics, and Physics,
Calle 15, e/ C y D, Vedado, Cp-10400, Havana, Cuba

Abstract

The question of finding feasible ways for estimating probability distributions is one of the main challenges for Estimation of Distribution Algorithms (EDAs). To estimate the distribution of the selected solutions, EDAs use factorizations constructed according to graphical models. The class of factorizations that can be obtained from these probability models is highly constrained. Expanding the class of factorizations that could be employed for probability approximation is a necessary step for the conception of more robust EDAs. In this paper we introduce a method for learning a more general class of probability factorizations. The method combines a reformulation of a probability approximation procedure known in statistical physics as the Kikuchi approximation of energy, with a novel approach for finding graph decompositions. We present the Markov Network Estimation of Distribution Algorithm (MN-EDA), an EDA that uses Kikuchi approximations to estimate the distribution, and Gibbs Sampling (GS) to generate new points. A systematic empirical evaluation of MN-EDA is done in comparison with different Bayesian network based EDAs. From our experiments we conclude that the algorithm can outperform other EDAs that use traditional methods of probability approximation in the optimization of functions with strong interactions among their variables.

Keywords

Estimation of distribution algorithms, Factorized distribution algorithms, Kikuchi approximations, factorizations, graphical models.

1 Introduction

Two main contributions of EDAs (Mühlenbein and Paaß, 1996; Larrañaga and Lozano, 2002) to Evolutionary Computation are:

- To demonstrate, based on a solid theoretical foundation, that an efficient evolutionary search can be achieved by estimating the distribution associated with the selected set, and using the estimated distribution in the generation of new solutions.
- To identify that one solution to the estimation problem is to find a set of marginal distributions, the combination of which gives a good approximation of the target distribution.

In their seminal work on EDAs, Mühlenbein and Paaß (1996) studied the first contribution, relating their findings with previous work on Genetic Algorithms (GAs). They found that the modeling of the selected individuals by means of the estimation of their probability distribution was one way to create more efficient evolutionary algorithms. Nevertheless, they did not provide a practical method for estimating the distributions. The second contribution was presented in (Mühlenbein et al., 1999), where the authors identified the factorization of the probability distribution according to a probability model, as a practical method that permits the computation of an estimation. This method for computing probability estimations has been the core of most known EDAs. Concerning the estimation of the probabilities, EDAs mainly differ in the types of probability models according to which the probability is factorized.

After the initial success of EDAs in the solution of difficult optimization problems, there arises the question of whether there exist other feasible methods for probability approximation, and how they can be inserted in the EDA framework. The investigation of new approximation methods could lead to the creation of more efficient and robust algorithms. Moreover, one lesson from the use of EDAs is that we can establish a close connection between the question of finding an adequate probability estimation, and the more general goal of achieving an appropriate problem decomposition. The estimation techniques used by EDAs, are not only tools for finding the optimal solutions, but also serve as a source of knowledge about the hidden structure of the optimization problem, usually reflected in the interactions among the variables which are captured by the probability models. The question of finding feasible ways for estimating probability distributions remains one of the main challenges for EDAs.

One route to the conception of new methods for probability estimation goes via the expansion of the class of factorizations that can be used in the approximation of distributions. The class of factorizations that can be obtained from traditional probability models is highly constrained. Expanding one that could be employed for probability approximation implies the design of algorithms for their identification or construction. In the case of EDAs, the methods for inferring the approximations have to be accompanied by efficient sampling algorithms, that could be used in the generation of new points.

This paper introduces a method that approximates probability distributions using what we have called “messy factorizations”. The algorithm presented here for learning the factorizations combines a reformulation of a probability approximation procedure used in statistical physics, with a novel approach for selecting the initial inputs required by the procedure. Then, a method is presented for sampling solutions from the approximation. The learning and sampling methods are the primary components of the Markov Network EDA, which is the main contribution of this paper. Finally it is shown that the introduced EDA is able to solve very difficult problems by means of representing complex interactions among the variables of the problem.

The remainder of this paper is presented as follows. The next section discusses the use of factorizations in EDAs. Section 3, moves from the description of the Markov Random Field (MRF) to an introduction of the concept of a clique based decomposition of a graph, and finally to a presentation of our Kikuchi approximation of the probability distribution. Section 4 shows a number of properties satisfied by the Kikuchi approximation, and section 5 describes a procedure for sampling points from the approximation. Section 6 presents an algorithm for learning Kikuchi approximations from the data. Section 7 introduces the Markov Network EDA, and discusses its computational cost. In section 8 we present a number of experiments that describe the dynamics of MN-EDA. We test its performance, comparing it with other EDAs. The paper concludes in section 9 by relating these findings to current research in EDAs and other research fields. A number of further enhancements to MN-EDA are also given.

2 Factorized distribution algorithms

This section begins by introducing some basic notations.

2.1 Preliminaries

Let $X = (X_1, \ldots, X_n)$ denote a vector of discrete random variables. We will use $x = (x_1, \ldots, x_n)$ 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 X (respectively 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 $X_S = x_S$. We use $p(x_i \mid x_j)$ to denote the conditional probability distribution of $X_i = x_i$ for given $X_j = x_j$.

An undirected graph $G = (V, E)$ is defined by a non-empty set of vertices V, and a set of edges E. An edge between vertices i and j will be represented by $i \sim j$.

Definition 1. *Given a graph $G = (V, E)$, a clique in G is a fully connected subset of V. We reserve the letter C to refer to a clique. The collection of all cliques in G is denoted as $\mathcal{C}$. C is maximal when it is not contained in any other clique. C is the maximum clique of the graph if it is the clique in $\mathcal{C}$ with the highest number of vertices.*

Definition 2. *A junction graph constructed from a set of maximal cliques is a graph where each node corresponds to a maximal clique, and there could be an edge between two nodes if their corresponding cliques overlap. A labeled junction graph is a junction graph that has an associated ordering of the nodes with a distinguished node called the root, and that has the property that a node belongs to the graph if at least one of the variables in the clique is not contained in the previous nodes in the ordering.*

An undirected graph is said to be chordal if every cycle of length four or more has a chord. A fundamental property of chordal graphs is that their maximal cliques can be joined to form a tree, called the junction tree, such that any two cliques containing a node α are either adjacent in the junction tree, or connected by a chain made entirely of cliques that contain α. This property is called the running intersection property (Lauritzen, 1996). A junction tree is an acyclic junction graph.

Independence graphs serve to display the probability dependencies that exist among the variables of a given probability distribution. Two vertices are joined in the independence graph if the corresponding variables are not conditionally independent given the rest of the variables. Throughout this paper we will only consider independence graphs defined on undirected graphs. Each variable X_i is associated to vertex V_i in the corresponding graph. To emphasize this relationship, vertices may be named as the variables $V = \{X_1, \ldots, X_n\}$. Whether we refer to a variable or a vertex will be made clear from the context.

Now, the problem of probability approximation using factorizations will be discussed. The next section briefly reviews EDAs that are based on undirected graphical models and that are relevant to our research.

2.2 Factorizations and FDAs

In simple terms, a factorization of a distribution p is a product of marginals of p. Factorizations are important because they allow us to obtain a condensed representation of otherwise very difficult to store probability distributions. We say that a factorization is valid when the maximal cliques in the factorization satisfy the running intersection property. Otherwise the factorization is said to be invalid.

There exist methods for finding valid factorizations based on the independence graph. It is very simple to construct an invalid factorization, but we can not guaran-

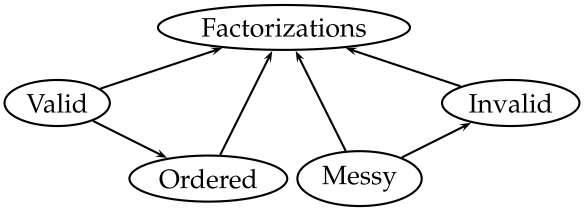

Figure 1: Classifications for different types of factorizations

tee the accuracy of an approximation of the distribution based on an arbitrary, invalid factorization. A criterion for the search of invalid factorizations is needed as is a more refined classification of the overly broad and ill defined class of invalid factorizations.

In (Santana, 2003), a method for constructing invalid factorizations has been proposed. The method consists of constructing a labeled junction graph from the maximal cliques of the independence graph of the distribution. Even if the factorization is invalid, it exhibits a convenient feature: 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. This property is crucial for the implementation of the sampling procedure as explained in (Santana, 2003).

The factorization based on the junction graph serves to propose a second classification of factorizations. We say that a factorization is "ordered" when an ordering of all the maximal cliques in the factorization exists such that for every clique there exists at least one variable that is not contained in the previous cliques in the ordering. When it is impossible to find such an ordering, we say that the factorization is "messy". The satisfaction of the running intersection property determines that every valid factorization is an ordered one. Figure 1 depicts the relationship between our two classifications of factorizations.

Example 1. *In this example, p_0 is an ordered, valid factorization. The cliques where marginals are calculated from, can be joined to form a junction tree. p_1 is an ordered, invalid factorization. In this case, cliques can not be arranged in a junction clique, but a labeled junction graph can be formed.*

$$p_0(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{p(x_1, x_2, x_3) \cdot p(x_1, x_2, x_5) \cdot p(x_1, x_3, x_6) \cdot p(x_2, x_3, x_4)}{p(x_1, x_2) \cdot p(x_1, x_3) \cdot p(x_2, x_3)}$$

$$p_1(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{p(x_1, x_2, x_3) \cdot p(x_1, x_2, x_5) \cdot p(x_1, x_3, x_6) \cdot p(x_4, x_5, x_6)}{p(x_1, x_2) \cdot p(x_1, x_3) \cdot p(x_5, x_6)}$$

One special subclass of EDAs includes the algorithms that use ordered factorizations of the probability distribution. In this paper we name this subclass Factorized Distribution Algorithms (FDAs). In the literature, the term FDA is frequently used to name a particular type of Factorized Distribution Algorithms introduced in (Mühlenbein et al., 1999). To avoid confusion, in this paper we name this algorithm FDA*. Our definition of FDAs covers FDA* and other algorithms that use ordered factorizations, whether they are based on directed or undirected graphical models.

One idea behind the development of FDAs is the assumption that an FDA can be more efficient as it is able to represent a higher number of the relevant interactions among the variables. It has been acknowledged that the existence of dependencies is not per se a source of difficulty, only malign or frustrating interactions can be difficult to deal with (Kallel et al., 2000). However, for many difficult problems, the real chal-

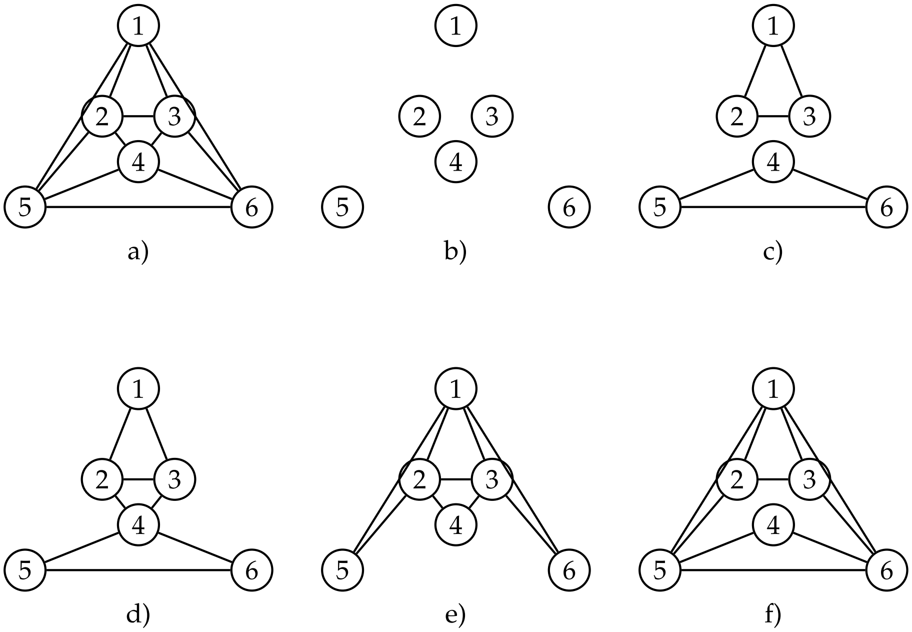

Figure 2: Independence graphs illustrating different examples of factorizations used by FDAs. a) Original independence graph, b) Valid factorization used by UMDA, c) Valid factorization used by ECGA, d), e) Valid factorizations used by FDA* and FDA-learning, f) Invalid factorization used by MN-FDA.

lenge is to find models able to represent these types of complex relationships, without incurring much computational overhead.

Consider the independence graph shown in figure 2a). It represents the dependencies that arise in a given probability distribution. The simplest approximate factorization of this distribution is represented by the independence graph shown in figure 2b). There are no edges between any pair of vertices because the model assumes that all the variables are independent. This is the model used by the Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein, 1997).

More complex factorizations move beyond the assumption of variable independence. They can be obtained by grouping subsets of variables with strong probability dependencies (figure 2c)). The Extended Compact Genetic Algorithm (ECGA) (Harik, 1999) uses a factorization of the probability where variables are separated in non overlapping clusters. FDA* (Mühlenbein et al., 1999), and FDA-learning (Ochoa et al., 1999) use valid factorizations with overlapping cliques. While FDA* uses a fixed model of interactions, and learns the parameters of the cliques in each generation, FDA-learning is able to learn the structure of the model from the data. Figure 2d) and figure 2e) show models that can also be handled by these algorithms.

The Markov Network FDA (MN-FDA) (Santana, 2003) extends the representation capabilities of previous FDAs. The main difference with respect to previous FDAs based on undirected models is that it uses a junction graph as its probabilistic model, being able to represent ordered invalid factorizations, like the one presented in example 1. Note that from the approximate factorization shown in figure 2b), to the one shown in figure 2f), there is an increment in the power of expression of the models in the sense that they are able to represent a higher number of those interactions that exist in the original model (figure 2a)). Note too, that none of the approximations can represent all the original interactions. In the next section we introduce a method to deal with this limitation of the models.

3 Kikuchi approximations of a probability distribution

The aim of this section is to introduce the method for estimating the probability distributions by means of messy factorizations. The approximation achieved by the method can be considered itself as an approximation of a Gibbs field. Therefore, we begin by giving the main concepts related to MRFs. Later, the path from MRFs to the use of Kikuchi approximations is briefly explained. The connection between the Boltzmann distribution, and the Kikuchi approximation is rather complex, and due to space limitations it cannot be described here. The interested reader is referred to (Yedidia et al.,2001;2002) for more details.

3.1 Markov Random Fields

Definition 3. *(Neighborhood, N): Given a graph G, the neighborhood $N(X_i)$ of a vertex $X_i \in X$ is defined as $N(X_i) = \{X_j : X_j \sim X_i \in E\}$. The set of edges uniquely determines a neighborhood system on G.*

Definition 4. *(Boundary, bd): The boundary of a set of vertices, $X_S \subseteq X$ is the set of vertices $X \setminus X_S$ that neighbors at least one vertex in X_S. The boundary of X_S is denoted $bd(X_S)$.*

Definition 5. *(Closure, cl): The closure of a set of vertices, $X_S \subseteq X$ is the set of vertices $cl(X_S) = X_S \cup bd(X_S)$.*

Definition 6. *A probability p is called a Markov Random Field with respect to the neighbor system on G if, $\forall x \in X,\ \forall i \in \{1, \ldots, n\}$, $p(x_i \mid x \setminus x_i) = p(x_i \mid bd(x_i))$.*

Definition 7. *A probability p on a graph G is called a Gibbs field with respect to the neighborhood system in G when it can be represented as follows:*

$$p(x) = \frac{1}{Z} e^{-H(x)}, \tag{1}$$

where $H(x) = \sum_{C \in \mathcal{C}} \Phi_C(x)$ is the energy function, $\Phi = (\Phi_{C_1}, \ldots, \Phi_{C_l})$ is the set of cliques potentials, one for each of the l maximal cliques in G. The value of $\Phi_C(x)$ depends on the local configuration on the clique C. The normalizing constant Z is the corresponding partition function, $Z = \sum_x e^{-H(x)}$.

Theorem 1. *Hammersley and Clifford theorem: Let p be a Gibbs Field defined on an independence graph G. Then, p is Markovian with respect to the graph. Conversely, the distribution p of any MRF over G that is strictly positive can be represented in factorized form.*

Let us suppose that the structure of the MRF is available in the form of an independence graph, and that approximate marginal probabilities for every maximal clique of the graph are also known. Based on this information, we would study how to discover an approximate probability distribution that captures the information in the data.

To learn an MRF, it is not enough to know the cliques potentials, it is also necessary to calculate the partition function Z. In general, finding Z is infeasible because it implies the summation on an exponential number of states. Therefore, if we intend to learn an MRF, a feasible solution to the problem of approximating Z has to be given. Fortunately, a similar problem has been treated in the field of statistical mechanics.

3.2 Energy and probability approximation in statistical systems

Statistical mechanics often treats models of interacting many-body systems. The macroscopic properties of the system are determined by the interactions among the microscopic elements. Usually, the interactions among the components of the system also determine the likelihood of the corresponding global configuration. Each state x of the system has an associated energy $H(x)$. A fundamental result of statistical mechanics is that, in thermal equilibrium, the probability of a state will be given by Boltzmann's Law:

$$p(x) = \frac{1}{Z(T)} e^{\frac{-H(x)}{T}}$$

where T is the temperature of the system, and $Z(T)$ the corresponding partition function.

Yedidia et al. (2001) suggested using the Boltzmann law postulate to define the energy corresponding to a joint probability of a non-physical system. The temperature is arbitrarily set to 1 because it simply sets a scale for the units in which one measures the energy. Similar proposals have been successfully used in optimization (Kirkpatrick et al., 1983; Mühlenbein et al., 1999).

Different techniques can be used to obtain an approximation of the energy. Kikuchi (1951) developed the Cluster Variation Method (CVM). At the base of CVM is the idea of approximating the energy as a function of a set of different marginals. The set does not necessarily form an exact factorization of the distribution, and the choice of the marginals influences the quality of the approximation.

3.3 Region based decompositions of graphs

Here, we concentrate on the question of finding an appropriate set of marginals from which to obtain an approximation of the distribution.

We define a region R of the neighborhood system $G = (V, E)$ to be a set $V' \subset V$. A graph region based decomposition is an asset of regions $\mathcal{R}$, and an associated set of *counting numbers* U which is formed by one counting number c_R for each $R \in \mathcal{R}$. c_R will always be an integer, but might be zero or negative for some R. The Kikuchi approximation is defined on a graph region based decomposition, calculating the marginal probabilities for each region. The methods used for creating the region based decomposition of the graph determine the final Kikuchi approximation.

To be valid, a decomposition must satisfy a number of constraints related to the regions and the counting numbers. Inspired in the work by Yedidia et al. (2002) , we refer to this sub-problem as *finding a valid region based decomposition of a graph*. In the CVM, $\mathcal{R}$ is formed by an initial set of regions $\mathcal{R}_0$ such that all the nodes are in at least one region of $\mathcal{R}_0$, and any other region in $\mathcal{R}$ is the intersection of one or more of the regions in $\mathcal{R}$. The set of regions $\mathcal{R}$ is closed under intersection, and can be ordered as a poset.

We say that a set of regions $\mathcal{R}$, and counting numbers c_R give a valid region based graph decomposition when for every variable X_i:

$$\sum_{\substack{R \in \mathcal{R} \\ X_i \subseteq X_R}} c_R = 1 \tag{2}$$

This condition arises from the fact that acceptable approximations of the entropy component of the free energy are those in which the appearance of each variable in the set of regions sums 1 (Pakzad and Anantharam, 2002).

Yedidia et al. (2002) introduced the concept of regions to define a region based free energy approximation on factor graphs. We have adopted their definition to explain the more general problem of finding a region based decomposition of our independence graph. The key aspect in the determination of a valid region based decomposition is the selection of the initial set of regions. The initial set of regions used by physicists are usually comprised of regions with the same size and topology.

This paper introduces a new method for selecting the first set of regions. The particular choice of the regions determines a number of convenient properties of the approximation of the probability. Given an arbitrary undirected graph G, we will form the set $\mathcal{R}_0$ by taking one region for each maximal clique in G. As a result, all the regions $R \in \mathcal{R}$ will be cliques because they are the intersection of two or more cliques. We call this type of region based decomposition of undirected graphs a *clique based decomposition.*

Example 2. *Valid clique based decomposition corresponding to the graph shown in figure 2a).*

$$\begin{aligned}
&\mathcal{R}_0 = \{\{1,2,5\},\{1,3,6\},\{1,2,3\},\{2,3,4\},\{2,4,5\},\{3,4,6\},\{4,5,6\},\{1,5,6\}\}\\
&\mathcal{R}_1 = \{\{1,2\},\{2,5\},\{1,3\},\{3,6\},\{2,3\},\{2,4\},\{3,4\},\{4,5\},\{4,6\},\{1,5\},\{1,6\},\{5,6\}\}\\
&\mathcal{R}_2 = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\}\\
&\qquad c_R = 1, \forall R \in \mathcal{R}_0; \quad c_R = -1, \forall R \in \mathcal{R}_1; \quad c_R = 1, \forall R \in \mathcal{R}_2
\end{aligned}$$

We define the Kikuchi approximation of the probability, denoted as k as:

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

where $\mathcal{R}$ comes from a region based decomposition of the graph. From now on we will constrain our analysis to Kikuchi approximations of the probability constructed from clique based graph decompositions.

Example 3. *Kikuchi approximation, constructed from the clique based graph decomposition presented in example 2.*

$$\begin{aligned}
L_0 &= p(x_1,x_2,x_5)\cdot p(x_1,x_3,x_6)\cdot p(x_1,x_2,x_3)\cdot p(x_2,x_3,x_4)\\
&\qquad \cdot p(x_2,x_4,x_5)\cdot p(x_3,x_4,x_6)\cdot p(x_4,x_5,x_6)\cdot p(x_1,x_5,x_6)\\
L_1 &= p(x_1,x_2)\cdot p(x_2,x_5)\cdot p(x_1,x_3)\cdot p(x_3,x_6)\cdot p(x_2,x_3)\cdot p(x_2,x_4)\\
&\qquad \cdot p(x_3,x_4)\cdot p(x_4,x_5)\cdot p(x_4,x_6)\cdot p(x_1,x_5)\cdot p(x_1,x_6)\cdot p(x_5,x_6)\\
L_2 &= p(x_1)\cdot p(x_2)\cdot p(x_3)\cdot p(x_4)\cdot p(x_5)\cdot p(x_6)\\
k(x) &= L_0 \cdot \frac{1}{L_1} \cdot L_2
\end{aligned}$$

Example 3 shows the Kikuchi approximation corresponding to the clique based graph decomposition presented in example 2. Note in the example that the maximal cliques can not be arranged to form an ordered factorization. The factorization shown in this example is a messy one. In general, the Kikuchi approximation will be an invalid, messy factorization.

A probability distribution p can be used in two main different forms. As an evaluator, or as a generator. When p is used as an evaluator it associates a real probability value to a given point x. If we want to use p as a generator we will expect it to generate the point x with a probability $p(x)$. In this case, the outcome of p is not a probability value but a sample of vectors with the desired frequency. To use the Kikuchi approximation as an evaluator of a given point x, the marginals defined on each region are multiplied, or divided, according to the sign of c_R.

A probability function $\tilde{p}$ based on the Kikuchi approximation can be found by normalizing k.

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

However, the determination of $\tilde{p}$ is not a realistic alternative, because it does not solve the problem of calculating a partition function $Z_k = \sum_{x'} k(x')$.

3.4 Algorithm for constructing valid region based decompositions

We present an overview of the algorithm used to construct the clique based decompositions. The algorithm receives as input the first set of regions $\mathcal{R}_0$. The final decomposition has only one set of regions $\mathcal{R}_1$ whose c_R values, $R \in \mathcal{R}_1$, are integers different from zero. Algorithm 1 presents the steps for finding a valid region based decomposition. Let $\mathcal{A}$ be an auxiliary set of regions, and $\Phi(\mathcal{A})$ be a function that finds all the intersection regions in $\mathcal{A}$. First, a candidate set of regions $\mathcal{Q}$ is constructed by progressively adding intersections of regions. When all possible intersections have been added, $\mathcal{Q}$ is ordered. The order is determined by the inclusion criterion. If $R_i \supseteq R_j$ then R_i precedes R_j in the ordering. Afterward, each region R in $\mathcal{Q}$ is inspected, and its c_R value is calculated as

$$c_R = 1 - \sum_{\substack{S \in \mathcal{R}_1 \\ S \supset R}} c_S \tag{4}$$

where c_S is the counting number of any region S in $\mathcal{R}_1$ such that S is a superset of R. If c_R is different from zero, the region is added to $\mathcal{R}_1$.

All the maximal cliques in $\mathcal{R}_0$ will be in $\mathcal{R}_1$ with $c_R = 1$. The algorithm finds a partially ordered set of all the regions. It is guaranteed (Morita, 1994) that when such a poset is closed under the intersection of regions, and the counting numbers are calculated using (4), the region based decomposition is valid.

Example 4 shows the different steps of algorithm 1 in the calculation of the counting numbers corresponding to the valid clique based decomposition shown in example 2.

Example 4.

$$\begin{aligned}
\mathcal{R}_0 &= \{\{1,2,5\},\{1,3,6\},\{1,2,3\},\{2,3,4\},\{2,4,5\},\{3,4,6\},\{4,5,6\},\{1,5,6\}\} \\
\mathcal{Q} &= \{\{1,2,5\},\{1,3,6\},\{1,2,3\},\{2,3,4\},\{2,4,5\},\{3,4,6\},\{4,5,6\},\{1,5,6\}, \\
&\quad \{1,2\},\{2,5\},\{1,3\},\{3,6\},\{2,3\},\{2,4\},\{3,4\},\{4,5\},\{4,6\},\{1,5\},\{1,6\},\{5,6\}, \\
&\quad \{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\} \\
&\quad c_R = 1, \forall R \in \mathcal{R}_0; \\
c_{\{1,2\}} &= 1 - (c_{\{1,2,5\}} + c_{\{1,2,3\}}) = -1
\end{aligned}$$

Let us suppose that all the regions of size 2 have been already added to $\mathcal{R}_1$, the counting numbers for these regions are all equal to -1, as shown in example 2. They were calculated similarly to

Algorithm 1: **Algorithm for constructing a valid region based decomposition**

```
$\mathcal{A} = \mathcal{R}_0$
$\mathcal{Q} = \mathcal{A}$
do {
    $\mathcal{A} = \Phi(\mathcal{A})$
    $\mathcal{Q} = \mathcal{Q} \cup \mathcal{A}$
} until $\mathcal{A} = \emptyset$
Find a partial ordering of $\mathcal{Q}$
$\mathcal{R}_1 = \emptyset$
$U = \emptyset$
For $R \in \mathcal{Q}$
    $c_R = 1 - \sum_{\substack{S \in \mathcal{R}_1 \\ S \supset R}} c_S$
    if $c_R \neq 0$ then
        $\mathcal{R}_1 = \mathcal{R}_1 \cup R$
        $U = U \cup c_R$
```

the one corresponding to $R = \{1,2\}$, *shown above. The other* 6 *regions have counting numbers equal to one, and are added to* $\mathcal{R}_1$. *Among the regions for which* $c_R = 1$ *is* $R = \{1\}$.

$$c_{\{1\}} = 1 - (c_{\{1,2,5\}} + c_{\{1,2,3\}} + c_{\{1,3,6\}} + c_{\{1,5,6\}} + c_{\{1,2\}} + c_{\{1,3\}} + c_{\{1,5\}} + c_{\{1,6\}}) = 1$$

4 Properties of the Kikuchi approximation

The Kikuchi approximation has a number of properties that can be used in the context of EDAs. First, we will show that in certain cases the Kikuchi approximation is a valid factorization. Then, we will demonstrate that the Kikuchi approximation satisfies the local Markov property.

4.1 Kikuchi approximations of chordal graphs

In the study of belief propagation algorithms the relationship between junction trees and general Kikuchi approximations has received particular attention. This topic is important because it helps to illustrate the conditions under which the Kikuchi approximation can be very accurate, or even exact. A detailed analysis of this issue is beyond the scope of this paper. What will be demonstrated here is that *the Kikuchi approximation constructed from a clique based decomposition of a chordal independence graph corresponding to a probability distribution* p, *gives a valid factorization of* p. But first some results related to junction trees are presented.

Since the graph is chordal, we can form a junction tree. Let us call $\{s_1, s_2, \ldots, s_l\}$ to the maximal cliques in the junction tree, and $\{r_1, r_2, \ldots, r_m\}$ to their overlappings, where l and m are respectively the number of maximal cliques and overlappings. p can be factorized as:

$$p(x) = \frac{\prod_{i=1}^{l} p(x_{s_i})}{\prod_{j=1}^{m} p(x_{r_j})^{(t_{r_j}-1)}}, \tag{5}$$

where t_{r_j} is the number of cliques that overlap in the overlapping clique r_j. A junction tree is a connected and acyclic graph, in this case it is straightforward to demonstrate (6).

$$\sum_{j=1}^{m}(t_{r_j}-1)=l-1 \tag{6}$$

We will use (6) and the running intersection property to demonstrate the following theorem.

Theorem 2. *When algorithm 1 is applied to a chordal independence graph, a clique based decomposition equivalent to the junction tree of the graph is obtained.*

Proof:

Since all the maximal cliques of the graph belong to the clique based decomposition and have counting numbers equal 1, all s_i are in the final clique based decomposition. We show now that the only other regions that belong to the decomposition are the overlappings $\{r_1, r_2, \ldots, r_m\}$ and have counting numbers $\{1-t_{r_1}, 1-t_{r_2}, \ldots, 1-t_{r_m}\}$

All the overlappings belong to the set of regions $\mathcal{Q}$ because they are the intersection of one or more maximal cliques. Intersections among other regions also belong to $\mathcal{Q}$. We assume that $\mathcal{Q}$ has already been filled and ordered. Also the maximal cliques have been added to $\mathcal{R}_1$. Recall that the maximal cliques are not contained in any other clique and their counting numbers are all equal to 1. Let $(R_1, R_2, \ldots R_q)$ be the regions in $\mathcal{Q}$ that belong to the maximal cliques. Recall that q is the number of these regions that are ordered according to the inclusion criterion. We use induction to demonstrate that from the regions $(R_1, R_2, \ldots, R_q)$ only those that belong to $\{r_1, r_2, \ldots, r_m\}$ will be added to $\mathcal{R}_1$.

R_1: R_1 cannot be an intersection of overlappings because any of the overlappings would precede it in $\mathcal{Q}$. R_1 can be only an intersection of maximal cliques that is not contained in any other intersection of maximal cliques. Therefore, $R_1 = r_i \in \{r_1, r_2, \ldots, r_m\}$. The counting number for r_i is found as:

$$c_{r_i} = 1 - \sum_{\substack{R\in\mathcal{R}_1\\ R\supset r_i}} c_R = 1 - t_{r_i}$$

where in this first case all the regions R belong to $\{s_1, s_2, \ldots, s_l\}$. As $1-t_{r_i}$ is different from zero, R_1 is added to $\mathcal{R}_1$.

R_{k+1}: Let us suppose that from the k regions $\{R_1, R_2, \ldots, R_k\}$: 1) Only those that belong to $\{r_1, r_2, \ldots, r_m\}$ have been added to $\mathcal{R}_1$ and, 2) every region r_i added to $\mathcal{Q}$ has counting number $1-t_{r_i}$. Then we demonstrate that if region $R_{k+1} = r_j \in \{r_1, r_2, \ldots, r_m\}$, then $c_{R_{k+1}} = 1 - t_{r_j}$ and R_{k+1} will be added to $\mathcal{R}_1$. If $R_{k+1} \notin \{r_1, r_2, \ldots, r_m\}$ then $c_{R_{k+1}} = 0$ and therefore it is not added to $\mathcal{R}_1$.

We calculate the counting number for R_{k+1} making a decomposition of regions in $\mathcal{R}_1$ in maximal cliques and overlappings.

$$\begin{aligned} c_{R_{k+1}} &= 1 - \sum_{\substack{R\in\mathcal{R}_1\\ R\supset R_{k+1}}} c_R \\ &= 1 - \left(\sum_{\substack{R\in\{s_1,s_2,\ldots,s_l\}\\ R\supset R_{k+1}}} c_R + \sum_{\substack{R\in\{\mathcal{R}_1\cap\{r_1,r_2,\ldots,r_m\}\}\\ R\supset R_{k+1}}} c_R \right) \end{aligned} \tag{7}$$

If $R_{k+1} = r_j \in \{r_1, r_2, \ldots, r_m\}$, and $r_j \not\subset r_i, r_i \in \mathcal{R}_1$ then the second part in (7) is zero and

$$c_{r_j} = 1 - \sum_{\substack{R \in \{s_1, s_2, \ldots, s_l\} \\ R \supset r_j}} c_R = 1 - t_{r_j}$$

This case corresponds to an overlapping that is not contained in any other overlapping. As $c_{r_j} \neq 0$, r_j will be added to $\mathcal{R}_1$.

If $R_{k+1} = r_j \in \{r_1, r_2, \ldots, r_m\}$, and r_j is the subset of at least one overlapping already in $\mathcal{R}_1$, all the maximal cliques that respectively overlap in each one of the overlappings that contain r_j form, together with their overlappings, a connected component. This fact is determined by the running intersection property, and the connected component is a junction tree. Let b be the total number of maximal cliques that overlap in overlappings where r_j is contained, then by (6),

$$\sum_{\substack{r_i \in \mathcal{R}_1 \\ r_i \supset r_j}} 1 - t_{r_i} = 1 - b$$

Exactly one of these b cliques overlaps with other $t_{r_j} - 1$ maximal cliques in r_j. Otherwise the junction tree will not be connected or acyclic. We rewrite (7) as

$$c_{r_j} = 1 - [(b + t_{r_j} - 1) + (1 - b)] = 1 - t_{r_j}$$

$c_{r_j} \neq 0$ and r_j is added to $\mathcal{R}_1$.

The remaining case is when $R_{k+1} \notin \{r_1, r_2, \ldots, r_m\}$. In this case R_{k+1} has to be contained in at least one overlapping already in $\mathcal{R}_1$ due to the running intersection property. All the maximal cliques that overlap in overlappings that contain R_{k+1} also form a connected and acyclic component. But in this case there are no maximal cliques that share R_{k+1} as overlapping, otherwise $R_{k+1} \in \{r_1, r_2, \ldots, r_m\}$ which is a contradiction. Therefore,

$$c_{R_{k+1}} = 1 - (b + 1 - b) = 0$$

As a consequence, R_{k+1} will not be added to $\mathcal{R}_1$. Since all the regions $r_i \in \{r_1, r_2, \ldots, r_m\}$ were in $\mathcal{Q}$, the final set of regions $\mathcal{R}_1$ will comprise all the maximal cliques and overlappings, and in this case the Kikuchi approximation is equal to the natural factorization shown in (5). □

4.2 Local Markov property of the Kikuchi approximation

When the independence graph is chordal the Kikuchi approximation gives a valid factorization of the underlying distribution p. If this happens $k(x) = p(x)$, but in the general case k is not a probability. However, the marginal and conditional functions of the Kikuchi approximation are defined as:

$$k(x_S) = \sum_{\substack{x' \\ x'_S = x_S}} k(x') \tag{8}$$

$$k(x_A \mid x_B) = \frac{k(x_{\{A \cup B\}})}{k(x_B)} \tag{9}$$

As k is not a probability, $k(x_S)$ and $k(x_A \mid x_B)$ are not either. Nevertheless, they can be used as approximations of $p(x_S)$ and $p(x_A \mid x_B)$ respectively.

Now, we will demonstrate that $k(x)$ satisfies a number of Markov properties. In the two following demonstrations we will use the fact that for any given variable $X_i,\ i \in \{1,\ldots,n\}$, $k(x)$ can be factorized in the product of the marginals of regions that contain X_i, and the marginal of regions that do not contain X_i.

Let $P(x,i,\mathcal{R}) = \prod_{\substack{R\in\mathcal{R}\\ X_R\supseteq X_i}} p(x_R)^{c_R}$, and $\bar{P}(x,i,\mathcal{R}) = \prod_{\substack{R\in\mathcal{R}\\ X_R\not\supseteq X_i}} p(x_R)^{c_R}$, then $k(x)$ can be expressed as in (10).

$$\begin{aligned} k(x) &= \prod_{\substack{R\in\mathcal{R}\\ X_R\supseteq X_i}} p(x_R)^{c_R} \prod_{\substack{R\in\mathcal{R}\\ X_R\not\supseteq X_i}} p(x_R)^{c_R} \\ &= P(x,i,\mathcal{R}) \cdot \bar{P}(x,i,\mathcal{R}) \end{aligned} \tag{10}$$

$P(x,i,\mathcal{R})$ and $\bar{P}(x,i,\mathcal{R})$ have been introduced for notational convenience, to have a more concise representation of $k(x)$. Also the following non-standard notation will be used to represent the summation over all the variables except X_S, when $X_S = x_s$.

$$p(x_S) = \sum_{\sim\{x_S\}} p(x') = \sum_{\substack{x'\\ x'_S=x_S}} p(x')$$

Notice the use of this notation in equation (11), which together with implications (12) and (13), are used in our demonstrations. Implication (12) derives from the fact that in the clique based decomposition any region is a clique. Therefore, any set of variables in the same region as X_i belongs to its closure.

$$\sum_{\sim cl(x_i)} \sum_{\sim\{x\setminus x_i\}} p(x') = \sum_{\sim bd(x_i)} p(x') \tag{11}$$

$$X_R \supseteq X_i \Rightarrow X_R \subseteq cl(X_i) \tag{12}$$

$$X_R \not\supseteq X_i \Rightarrow X_R \subseteq X \setminus X_i \tag{13}$$

First, we give a simplified expression for $k(x_i \mid bd(x_i))$, then we demonstrate that $k(x_i \mid x \setminus x_i) = k(x_i \mid bd(x_i))$.

Theorem 3.

$$k(x_i \mid bd(x_i)) = \frac{P(x,i,\mathcal{R})}{\sum_{\sim\{x\setminus x_i\}} P(x,i,\mathcal{R})} \tag{14}$$

Proof:

$$\begin{aligned}
&k(x_i \mid bd(x_i)) \\
&= \frac{\sum_{\sim cl(x_i)} P(x', i, \mathcal{R})\bar{P}(x', i, \mathcal{R})}{\sum_{\sim bd(x_i)} P(x', i, \mathcal{R})\bar{P}(x', i, \mathcal{R})} \\
&= \frac{P(x, i, \mathcal{R}) \sum_{\sim cl(x_i)} \bar{P}(x', i, \mathcal{R})}{\sum_{\sim cl(x_i)} \sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R})\bar{P}(x', i, \mathcal{R})} \\
&= \frac{P(x, i, \mathcal{R}) \sum_{\sim cl(x_i)} \bar{P}(x', i, \mathcal{R})}{\sum_{\sim cl(x_i)} \left(\bar{P}(x', i, \mathcal{R}) \sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R}) \right)} \\
&= \frac{P(x, i, \mathcal{R})}{\sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R})} \frac{\sum_{\sim cl(x_i)} \bar{P}(x', i, \mathcal{R})}{\sum_{\sim cl(x_i)} \bar{P}(x', i, \mathcal{R})} \\
&= \frac{P(x, i, \mathcal{R})}{\sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R})} \\
&\square
\end{aligned}$$

Theorem 4.

$$k(x_i \mid x \setminus x_i) = k(x_i \mid bd(x_i))$$

Proof:

$$\begin{aligned}
&k(x_i \mid x \setminus x_i) \\
&= \frac{k(x)}{k(x \setminus x_i)} \\
&= \frac{P(x, i, \mathcal{R}) \cdot \bar{P}(x, i, \mathcal{R})}{\sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R}) \cdot \bar{P}(x', i, \mathcal{R})} \\
&= \frac{P(x, i, \mathcal{R}) \cdot \bar{P}(x, i, \mathcal{R})}{\bar{P}(x, i, \mathcal{R}) \sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R})} \\
&= \frac{P(x, i, \mathcal{R})}{\sum_{\sim \{x \setminus x_i\}} P(x', i, \mathcal{R})} \\
&= k(x_i \mid bd(x_i)) \\
&\square
\end{aligned}$$

Also the normalized version of the Kikuchi approximation satisfies theorem 3.

$$\begin{aligned}
\tilde{p}(x_i \mid bd(x_i)) &= \frac{\tilde{p}(cl(x_i))}{\tilde{p}(bd(x_i))} \\
&= \frac{\sum_{\sim cl(x_i)} \tilde{p}(x')}{\sum_{\sim bd(x_i)} \tilde{p}(x')} \\
&= \frac{\sum_{\sim cl(x_i)} k(x') \setminus Z_k}{\sum_{\sim bd(x_i)} k(x') \setminus Z_k} \\
&= k(x_i \mid bd(x_i))
\end{aligned}$$

Theorems 4 and 3 are both important for our purpose of sampling from the Kikuchi approximation. Theorem 4 demonstrates that in the Kikuchi approximation each variable is conditionally independent of the rest given its closure. Therefore, for doing sampling we only need to use the conditional probability of each variable given its closure. Theorem 3 gives a convenient expression for calculating the conditional probability of X_i given its closure. This probability can be calculated from the marginal probabilities defined on those cliques that contain X_i. Theorem 3 also shows an interesting property of the Kikuchi approximation. For every variable X_i, functions $k(X_i \mid bd(X_i))$ are probability distributions due to the normalization in the denominator.

5 Sampling of the Kikuchi approximation using the Gibbs Sampler

Section 3.3 explained how the Kikuchi approximation can be used as an evaluator. To use it as a generator, we will take advantage of the local Markov property it satisfies. As k is not a probability distribution, we assume that points will be generated from its normalized expression $\tilde{p}(x)$. To sample points from $\tilde{p}(x)$, we will use the Gibbs sampler or Gibbs Sampling algorithm (Geman and Geman, 1984). It is assumed that whilst $\tilde{p}(x)$ is too complex to draw samples from directly, its conditional distributions $\tilde{p}(x_i \mid x \setminus x_i) = k(x_i \mid x \setminus x_i)$ are tractable to work with.

To draw samples from $\tilde{p}(x)$, start with an initial state $x^0 = (x_1^0, \ldots, x_n^0)$. At each time t of the GS, a location i to be updated is chosen from x^t, and the new vector x^{t+1} is selected using the transition rules in (15).

$$x^{t+1} = \begin{cases} x_j^{t+1} = x_j^t & \text{for } j \neq i \\ x_j^{t+1} \approx k(x_i^t \mid bd(x_i^t)) & \text{for } j = i \end{cases} \tag{15}$$

We define VS, Cy, and In 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 sampling starts from can be randomly selected ($In = 0$), or sampled from an approximate factorization found using a chordal subgraph of the independence graph ($In = 1$).

GS guarantees that after a sufficiently long time, say for t steps, each state of the chain can be taken as a sample of the target probability. In our case the marginal probabilities are calculated using the Laplace estimator, therefore conditional distributions are all positive, and GS will determine an ergodic Markov chain that will converge to the target probability $\tilde{p}(x)$.

6 Learning the Kikuchi approximation from the data

In the previous sections we have assumed that the maximal cliques of the independence graph, and their marginal probabilities are known. To use the Kikuchi approximation in the context of an EDA we need a method that learns the structure and parameters of the model. Algorithm 2 is an extension of the algorithm introduced in (Santana, 2003) for learning ordered factorizations from the data. Steps from 1 to 3 are the same, in steps 4 and 5 the Kikuchi approximation is constructed.

Algorithm 2: **Algorithm for learning the Kikuchi approximation from the data**

```
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 Construct a clique based decomposition of the graph.
5 Find the marginal probabilities for the regions of the decomposition.
```

6.1 Learning an independence graph

The independence graph can be learned from the data by means of score+search methods or using independence tests. In the first case, it is important to take into account the fact that the final graph may have cycles, this is not the case in traditional applications of score+search methods. We have used an algorithm that constructs the graph by means of independence tests. To determine if an edge belongs to the graph, it is enough to make an independence test on each pair of variables given the rest. Nevertheless, from an algorithmic point of view it is important to reduce the cost of the independence tests. We have adopted the methodology previously followed by Spirtes et al. (1993) . The idea is to start from a complete undirected graph, and then try to remove edges by testing for conditional independence between the linked nodes, but using conditioning sets as small as possible.

The current implementation can perform independence tests of order zero, one, and two. However, in all the experiments presented in this paper we only used independence tests up to order one. As the order of tests increases their reliability decreases, and therefore the use of higher order tests is not practical, unless a big population size is employed. 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 $w(i,j)$.

6.2 Refinement of the graph

When the independence graph is very dense, we can expect that 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. Our refinement algorithm avoids introducing a bias in the way the edges are removed. However, it has one main drawback: it could be the case that there exist more than $r-1$ variables depending on a single one, but the maximum clique where this variable is included is smaller than r. In this case, the procedure that eliminates the edges would remove dependencies from the graph without a real need to do so. An alternative to circumvent this problem is analyzed in the following section.

6.3 Maximal cliques of the graph

To find all the cliques of the graphs the Bron and Kerbosch algorithm (Bron and Kerbosch, 1973) is used. This algorithm uses a branch and bound technique to cut off branches that can lead to cliques. 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 G' of G is calculated as $W(G') = \sum_{i \sim j \in G'} w(i,j)$. These weights associated with each maximal clique are used to construct a junction graph that serves to generate non-random initial points for the GS. The algorithm to construct the junction graph is explained in (Santana, 2003).

An alternative to the refinement of the graph is to adapt the algorithm for finding cliques to test whether cliques have size greater than r, and to remove edges with lowest weights. This alternative may need more than one pass of the cliques finding algorithm. Also, the growth of the number of smallest cliques must be taken care of. In the experiments presented in this paper only the refinement of the graph was used.

7 The Markov Network EDA

The EDA based on the Kikuchi approximation is being referred to as Markov Network EDA (MN-EDA). It is related to MN-FDA because both algorithms share part of the procedure used for learning the probabilistic model. Nonetheless, while MN-FDA can only handle ordered factorizations, MN-EDA is able to use the messy ones. The goal of using the Kikuchi approximation in the context of EDAs is to have a probabilistic model able to capture as many dependencies as possible, but without adding new edges to the independence graph, as other procedures (e.g. triangulation algorithms) do. The MN-EDA's pseudo-code is shown in algorithm 3.

Algorithm 3: **Markov Network EDA**

```
Set $t \Leftarrow 0$. Generate $N$ solutions randomly.
do {
    Select a set $S$ of $l \leq N$ solutions according to a selection method.
    Learn a Kikuchi approximation of $S$ using algorithm 2.
    Generate $N$ new solutions sampling from the Kikuchi approximation.
    $t \Leftarrow t + 1$
} until Termination criteria are met
```

In the current implementation, the level of significance α was set to 0.75. This choice was motivated by the need to capture as many dependencies as possible. A low significance level will allow more dependencies to be detected in the last generations, when many variables tend to be independent. Even if some of the dependencies found might be false, this is better than missing some of the real dependencies. If many dependencies are learned, the refinement algorithm will contribute to eliminate those with lowest weights, balancing the effect of the low significance level. The maximum number of allowed neighbors for the refinement algorithm was set to 9. In all the experiments presented in this paper, GS was used with the random visitation scheme.

The name MN-EDA^f is given to the MN-EDA that uses a fixed model of interactions. This algorithm can be given the clique based decomposition, or can calculate it from a given independence graph. Once the clique based decomposition has been constructed, it will be used in all the generations of the algorithm. In every generation, MN-EDA^f only makes a parametric approximation of the marginal probabilities of the regions.

7.1 Computational cost of the MN-EDA

The number of operations needed to make the independence tests is upper bounded by $O(Nn^3)$. The worst complexity of the refinement algorithm is bounded by $O(n^2 log(n))$. It has been calculated considering the case when after the independence tests, the graph remains complete. The time complexity of the Bron and Kerbosch algorithm is not calculated in their original work (Bron and Kerbosch, 1973). However from comparisons with other algorithms for which bounds have been calculated (Bomze et al., 1999), the worst complexity of the algorithm can be estimated as $O(\mu^2)$, where μ is the number of maximal cliques. When there are at most k edges for each variable, and $k << n$, a bound for the number of cliques can be given by kn, and the complexity of the Bron and Kerbosch algorithm roughly estimated as $O(\mu^2) \approx O(n^2)$. The complexity of learning the parameters depends on the size of the population N, the number of cliques μ, and their size. The order of this step is $O(N\mu) \approx O(Nn^2)$.

The cost of the method for finding a clique based decomposition clearly depends on the number of cliques, their size, and more importantly, the number of overlappings among the variables. The number of comparisons needed to find the first set of overlappings is $\frac{(\mu)(\mu-1)}{2}$, this is also a bound for the number of new regions in the first set. The process is repeated taking this maximum number of regions as a bound. A bound for the total number of regions is $N_R = r \cdot \frac{(\mu)(\mu-1)}{2}$, where r is the size of the maximum clique in the graph. The cost of the sampling algorithm can be estimated as:

$$N \cdot C_y \cdot \sum_{i=1}^{n} \sum_{R \in \mathcal{R}} (2^r - 1) \approx Nn2^r,$$

where C_y is the number of cycles of the GS. The cost is exponential on the size of the maximum clique, and linear in N, and n. Considering C_y and r constant, the total complexity of MN-EDA is $O(GNn^3)$, where the population size N and the number of generations G change according to the difficulty of the problem. In the next section, the question of population size scaling is investigated.

8 Experiments

The experiments presented here pursue two main goals. To reveal the way the different components of MN-EDA determine its performance, and to evaluate the behavior of MN-EDA in the optimization of theoretical and practical functions. To achieve these goals the experiments have been divided into three parts. First, the role played by the learning and sampling components of MN-EDA are illustrated. Then a number of experiments are given where MN-EDA is compared with other EDAs. Finally, the results on the scalability of the algorithm are discussed.

The design of the experiments has privileged the investigation of those algorithms' factors related to the probabilistic model used by MN-EDA. We focus on the strategies and parameters used in learning the model, and on the sampling step of the EDA. We have not investigated other factors that could improve the behavior of the algorithm, such as the type and parameters of the selection method, the number of elitist solutions, non-random initialization of the first population, the incorporation of local optimization methods, or the use of more sophisticated replacing strategies.

Unless otherwise stated, 100 runs of every experiment have been made. From these experiments, we have calculated the successful rate (S), which is the number of the successful runs out of 100. The average number of generations ($\bar{g}$), and the average number of function evaluations ($\bar{e}$) to reach the optimum were calculated from the runs where it was found. The results of the comparisons among EDAs have all been obtained from our own simulations. To this end, we have incorporated into our computing environment the code developed for different Bayesian network based EDAs. Unless otherwise stated, all the algorithms used the following settings.

1. Random generation of the initial population.
2. Truncation selection with $T = 0.15$.
3. The best individual in each generation is passed to the next one.
4. Termination criteria are:
   - The optimum is found.
   - The maximum number of generations has been reached.
   - All the individuals in the population are identical.

8.1 Functions and problems used to evaluate the algorithms

Let $u(x) = \sum_{i=1}^{n} x_i$, $f(x)$ is a unitation function if $\forall x, y \in \{0,1\}^n, u(x) = u(y) \Rightarrow f(x) = f(y)$. A unitation function can be defined in terms of its unitation value $u(x)$, or in a simpler way u. Unitation functions serve for the definition of deceptive functions, so called because they show the deceptive nature of GAs behavior.

Functions F_1, F_2^k are deceptive functions formed by the additive sum of deceptive sub-functions with definition sets that do not overlap.

$$F_1 = f_{3deceptive}(x) = \sum_{i=1}^{\frac{n}{3}} f_{Gdec}^3(x_{3i-2} + x_{3i-1} + x_{3i}) \tag{16}$$

$$f_{Gdec}^3(u) = \begin{cases} 0.9 & for \quad u = 0 \\ 0.8 & for \quad u = 1 \\ 0.0 & for \quad u = 2 \\ 1.0 & for \quad u = 3 \end{cases}$$

F_2^k represents the class of parametric deceptive function, where k is a parameter of the function that determines the order of the sub-functions,

$$F_2^k = f_{deceptivek}(x) = \sum_{i=1}^{\frac{n}{k}} f_{dec}^k(x_{k \cdot i-k+1} + x_{k \cdot i-2} + \cdots + x_{k \cdot i}) \tag{17}$$

$$f_{dec}^k(u) = \begin{cases} k-1 & for \quad u = 0 \\ k-2 & \quad\quad u = 1 \\ & \cdots \\ k-i-1 & \quad\quad u = i \\ & \cdots \\ k \cdot n & for \quad u = k \end{cases}$$

Function F_3 is also a deceptive function with overlapping definition sets, $m = n-1$.

x	00	01	10	11
$IsoC_1$	m	0	0	$m-1$
$IsoC_2$	0	0	0	m

(18)

$$F_3 \quad = \quad F_{IsoPeak}(x) = IsoC_2(x_1, x_2) + \sum_{i=2}^{m} IsoC_1(x_i, x_{i+1})$$

When analyzing the different types of interactions that can arise among the variables, it is important to consider interactions that do not depend on the linear codification of solutions. To this end function F_4 (19) was considered, where x_{up}, x_{left}, etc., are defined as the appropriate neighbors, wrapping around. In this case, we take $n = m^2$.

u	0	1	2	3	4	5
$IsoT_1$	m	0	0	0	0	$m-1$
$IsoT_2$	0	0	0	0	0	m^2

(19)

$$\begin{aligned} F_4 \quad = \quad & f_{Isotorus} = IsoT_1(x_{n-m+1} + x_m + x_1 + x_2 + x_{1+m}) + \\ & \sum\nolimits_{i=2}^{n} IsoT_2(x_{up} + x_{left} + x_i + x_{right} + x_{down}) \end{aligned}$$

$Cuban5$ (Mühlenbein et al., 1999) is a non-separable additive function. The second best value of this function is very close to the global optimum.

$$Cuban5(x) = F^5_{cuban1}(s_0) + \sum_{j=0}^{m} (F^5_{cuban2}(s_{2j+1}) + F^5_{cuban1}(s_{2j+2})), \tag{20}$$

where $s_i = x_{4i}x_{4i+1}x_{4i+2}x_{4i+3}x_{4i+4}$ and $n = 4(2m+1)+1$

$$F^3_{cuban1}(x) = \begin{cases} 0.595 & for \quad x = 000 \\ 0.200 & for \quad x = 001 \\ 0.595 & for \quad x = 010 \\ 0.100 & for \quad x = 011 \\ 1.000 & for \quad x = 100 \\ 0.050 & for \quad x = 101 \\ 0.090 & for \quad x = 110 \\ 0.150 & for \quad x = 111 \end{cases}$$

$$F^5_{cuban1}(x) = \begin{cases} 4 * F^3_{cuban1}(x_1, x_2, x_3) & if \ x_2 = x_4 \ and \ x_3 = x_5 \\ & \\ 0 & otherwise \end{cases}$$

$$F^5_{cuban2}(x) = \begin{cases} u(x) & for \quad x_5 = 0 \\ 0 & for \quad x_1 = 0, x_5 = 1 \\ u(x) - 2 & for \quad x_1 = 1, x_5 = 1 \end{cases}$$

The Hierarchically-if-and-only-if(HIFF) function (Watson et al., 1998) is defined by the recursive function f, $HIFF(x) = f(x_{\{1,\ldots,n\}})$.

$$f(x_{\{1,\dots,s\}}) = \begin{cases} 1, & \text{if } (|s| = 1) \\ |s| + f(x_{\{1,\dots,\frac{s}{2}\}}) + f(x_{\{\frac{s}{2}+1,\dots,s\}}), & \text{if } (|s| > 1) \text{ and} \\ & (\sum_{i=1}^{|s|} x_i = 0, \text{or} \sum_{i=1}^{|s|} x_i = |s|) \\ f(x_{\{1,\dots,\frac{|s|}{2}\}}) + f(x_{\{\frac{|s|}{2}+1,\dots,|s|\}}), & \text{otherwise,} \end{cases} \tag{21}$$

where s is a set of indices of contiguous variables in x, $|s|$ is the cardinality of s, and $x_{\{1,\dots,\frac{s}{2}\}}$ and $x_{\{\frac{s}{2}+1,\dots,s\}}$ are respectively the sets of variables in left and right halves of x. In the first call to the function, $|s| = n = 2^p$, with p an integer indicating the number of hierarchical levels.

The generalized Ising model is described by the energy functional (Hamiltonian) (22) 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 -1. One 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.

$$H = -\sum_{i<j \in L} J_{ij}\sigma_i\sigma_j - \sum_{i \in L} h_i\sigma_i \tag{22}$$

8.2 Study of the Kikuchi approximation

We will begin with one experiment that builds some intuition regarding the difference between the probabilistic models used by MN-FDA and MN-EDA. The objective of our experiment is to compare the approximation quality of the distribution of the selected set given by an ordered factorization, and a Kikuchi approximation. The optimization problem is the problem of finding the ground state of the Ising model on a grid of dimensions 4x4 with toroidal neighborhood.

The generalized Ising model problem is perhaps one of the clearest examples of the difference between traditional and Kikuchi approaches to probability approximation. In (Mühlenbein et al., 1999) it is stated without proof that exact factorizations of additively decomposed functions defined on 2-D grids require the computation of marginal distributions of size $O(\sqrt{n})$, where n is the number of elements in the grid. If we assume this statement to be true, the dimension of the probability model will be exponential in n. In any case, the size of the maximum clique in triangulated grids is clearly greater than 2.

The junction graph and the Kikuchi approximation use cliques from the original graph. The maximal cliques in the grid correspond to edges and thus have size 2. These approximations are feasible as well as efficient ways to store a factorization of the distribution. While the junction graph uses only some edges of the graph, the Kikuchi approximation uses all the edges of the graph.

In our experiment, the quality of the approximation is assessed only for the first selected set of the EDA. An initial population of 1000 individuals is generated, truncation selection is done, and the independence graph is learned from the data. Table 1 shows the set of cliques (edges) learned from the selected set. All the 19 edges learned correspond to interactions that arise in the toroidal grid-like structure of the problem. On the other hand, 13 of the 32 interactions represented in the structure of the grid have not been captured in the independence graph. This can be explained because the existence of a link in the structure does not necessary mean the existence of dependencies, these dependencies are also determined by the strength of the interactions J_{ij}.

E	P	W	p_{JT}	p_{K1}	p_{K100}	E	P	W	p_{JT}	p_{K1}	p_{K100}
$(6,10)$	1	19.36	0.028	0.039	0.056	$(14,2)$	11	8.55	0.035	0.066	0.059
$(6,5)$	2	11.71	0.019	0.091	0.034	$(14,15)$	12	18.47	0.034	0.076	0.017
$(9,10)$	3	11.22	0.057	0.091	0.056	$(3,2)$	13	8.34	0.055	0.029	0.050
$(8,9)$	4	17.28	0.019	0.059	0.073	$(7,11)$	14	12.77	0.085	0.027	0.049
$(8,12)$	5	15.86	0.041	0.009	0.018	$(1,5)$	$-$	9.65	0.339	0.053	0.073
$(9,13)$	6	10.12	0.068	0.091	0.033	$(3,0)$	$-$	7.94	0.282	0.084	0.037
$(1,13)$	7	15.97	0.029	0.030	0.053	$(12,13)$	$-$	7.32	0.270	0.030	0.051
$(1,2)$	8	13.63	0.035	0.025	0.053	$(6,2)$	$-$	6.72	0.283	0.078	0.093
$(1,0)$	9	10.55	0.039	0.084	0.053	$(4,8)$	$-$	6.31	0.315	0.062	0.017
$(4,5)$	10	9.64	0.013	0.054	0.073	$Tot.$			2.05	1.09	0.95

Table 1: Errors in the approximation of the marginals of the selected set given by valid and Kikuchi approximations.

From the undirected graph, the two possible probabilistic models are constructed. Although in the general setting the labeled junction graph can represent cycles, when the maximum clique size in the junction graph is 2, the junction graph has no cycles. This happens because at least one variable in every clique of the ordered factorization has to be new in the ordering. Hence, each new clique in the ordering can be only connected to exactly one previous clique in the ordering. The ordered factorization of the learned graph is composed by the cliques numbered in columns 2 and 8. These edges form a tree, more exactly, a forest. In this case, the factorization is valid. For the Kikuchi approximation, all the edges are used in the model.

New populations are generated from each model. In the case of the Kikuchi approximation, we set the GS to begin the sampling with a point sampled from the ordered factorization ($In = 1$). Let p^s_{ij} and p^{new}_{ij} respectively be the bivariate marginal of (X_i, X_j) calculated from the selected set, and from the new population. The error in the approximation is calculated as $\sum_{X_i=x_i, X_j=x_j}(p^{new}_{ij} - p^s_{ij})$.

In the table, p_{JT} represents the error given by the valid approximation for different marginals. The errors are small for the marginals corresponding to the numbered edges, but remarkably worse for the remaining edges. This happens because the junction tree cannot cover these interactions and use them in the sampling process. After only one cycle of the GS, the Kikuchi approximation gets results (p_{K1}) better than the junction tree approximation. For the Kikuchi approximation there are no important differences between the approximation errors of the different cliques. In the table, p_{K100} shows the error when the number of sampling steps is increased to 100. It can be seen that the overall error diminishes. Nevertheless, the error in the approximation of some edges can increase, probably due to the sampling errors of GS.

Summarizing, the results of the experiments illustrate that by covering a higher number of dependencies than the junction tree, the Kikuchi approximation can be more accurate.

8.2.1 Influence of the GS in the convergence results

The objective of the following experiment is to determine if the way GS is initialized, and the number of cycles of the GS, have a significant effect on the behavior of MN-EDA. The experiment consists of running MN-EDA for functions F_1-F_4. The goal is to evaluate the effect that tuning the investigated parameters has in the convergence rate, and the number of generations of MN-EDA. The results of these experiments are shown in table 2. In all the experiments $n = 36$. The upper part of the table shows the results of the MN-EDA that uses the GS with random initialization. The bottom part shows the MN-EDA that uses the GS initialized with a point sampled from an ordered factorization. This factorization has been constructed from a subgraph of the independence graph.

Init	Cycles	F_1		F_2^3		F_2^4		F_3		F_4	
		S	$\bar{g}$	S	$\bar{g}$	S	$\bar{g}$	S	$\bar{g}$	S	$\bar{g}$
Random	2	1	13.00	4	15.00	4	10.00	51	12.43	97	8.42
	4	56	14.80	67	13.39	61	13.98	100	5.98	98	6.03
	6	80	13.76	87	11.77	98	10.70	99	5.73	96	6.04
	8	96	12.95	95	10.79	100	9.38	98	5.85	90	5.04
	12	98	11.97	100	6.91	99	8.43	94	5.66	88	4.68
	16	95	11.44	99	5.69	97	8.14	87	6.19	95	5.14
	20	91	10.78	100	5.87	95	8.10	77	6.35	90	4.25
Fact.	0	78	7.78	100	4.70	92	5.58	60	5.03	80	5.90
	1	94	8.69	100	4.77	99	5.64	70	4.78	90	4.86
	2	89	8.56	100	4.78	100	5.53	86	5.37	95	4.43
	3	92	9.44	100	4.87	100	5.67	89	6.56	98	4.59
	4	87	8.67	100	4.82	100	5.96	95	7.32	95	4.33
	5	87	9.04	100	4.88	99	5.99	97	6.55	96	4.59

Table 2: Successful rate and average number of generations of the MN-EDA for different GS initializations, and different number of cycles.

For the MN-EDA that uses the GS with random initialization, the successful rate increases with the increment in the number of cycles. After two cycles of GS, every variable has been updated on average a couple of times. This means that if the initial configurations of the vectors have been randomly set, the distribution of the sampled points will be far from the target Kikuchi approximation. After more cycles of the GS the distribution recovered in the new sampled population is a more accurate Kikuchi approximation. This fact contributes to a more efficient search, which is expressed in a higher success rate. As the number of cycles is increased, the average number of generations to find the optimum diminishes. However, care must be taken because too many sampling cycles produces what is called the overfitting problem.

Overfitting the data by the probabilistic model refers to an overly accurate approximation of data, which does not reflect more general features of the search space. When it occurs in the first generations of EDAs, it can lead to a premature convergence of the algorithms. Bayesian network based EDAs deal with this problem by constraining the number of parents in the Bayesian networks, and in some cases adjusting the penalty on the complexity of the network.

There exist differences in the rate of convergence for the functions. Functions F_3 and F_4 need less cycles than the other functions to be optimized with a successful rate over 90. For function F_3, results deteriorate when the number of cycles of MN-EDA are increased. This is a function that can be easily optimized with an FDA that uses a simple chain shaped graphical model. The more complex Kikuchi approximation, together with many GS cycles, provokes overfitting of the data for this function.

To use MN-EDA with random initialization is not a practical alternative because many cycles are needed, and sampling is a time consuming task. The solution we have found is to begin the simulation of the GS from a vector sampled from an ordered factorization. The ordered factorization can be based on a junction graph. In this case, if no GS step is applied after initializing the vector from the factorization, MN-EDA will behave as the MN-FDA. In all the following experiments MN-EDA uses only one cycle of GS. By using only one cycle of GS on a solution sampled from a junction graph, we intend to avoid overfitting, looking for a balance between the exploration and exploitation capabilities of the model.

Summarizing: GS plays an important role for MN-EDA. This sampling method guarantees that new points will have a distribution similar to the distribution of the selected set. The number of cycles, and the initial point of the simulation are both critical elements of the algorithm.

8.3 Comparison with other EDAs

We compare the behavior of MN-EDA with the following FDAs: EBNA$_B$, EBNA$_{local}$, EBNA$_{K2}$ (Etxeberria and Larrañaga, 1999; Larrañaga and Lozano, 2002), the Bayesian Optimization Algorithm (BOA) (Pelikan et al., 1999), and the Learning Factorized Distribution Algorithm (LFDA) (Mühlenbein and Mahnig, 2001). All are Bayesian network based FDAs that use different metrics to construct the Bayesian structure (Larrañaga and Lozano, 2002). In the cases of the EBNA algorithms, and BOA, it is important to note that the selection methods and/or replacing strategies commonly used by these algorithms are not the ones employed in our experiments. We have adapted the programming code of these algorithms to work with truncation selection with $T = 0.15$. This modification helped us to focus on the impact of the differences among their corresponding probability models in the behavior of the algorithms, disregarding the influence of other factors. Our experimentation design includes results obtained with UMDA, and the Mixture of Trees FDA (MT-FDA) (Santana et al., 2001), an FDA that uses a mixture of tree models to estimate and sample the probability.

8.3.1 Results for deceptive functions

In table 2 it can be seen that in many cases the results of MN-EDA improve when the number of cycles is increased. Now, we analyze the behavior of the MN-EDA with fixed settings compared to other EDAs. In figure 3 we show the successful rate of MN-EDA, MN-FDA, EBNA$_B$, EBNA$_{local}$, UMDA and LFDA in the optimization of four of the five functions shown in table 2. Results for EBNA$_{K2}$ and BOA were not included in the figure since they were not better than those obtained with the other EDAs. Function F_2^3 has not been included because it is very easy to optimize for all the algorithms. The successful rates corresponding to MN-EDA and MN-FDA have been taken from table 2.

It can be appreciated in the figure that UMDA is only able to optimize function F_4, for the rest of functions its successful rate is zero. This fact gives us an idea of the importance of capturing the relevant interactions for the optimization of these functions. MN-EDA clearly outperforms the Bayesian network based FDAs in functions F_1 and F_2. It ranks third for function F_3, and second for function F_4. This is a good performance, particularly if we take into consideration that by increasing the number of cycles of GS, the results of MN-EDA could be improved for functions F_3 and F_4, as it has been shown in table 2.

8.3.2 Results for the problem of finding a ground state of the Ising model

In the following experiments we assess the behavior of MN-EDA for the problem of finding a ground state of the Ising model, and the influence of incorporating prior

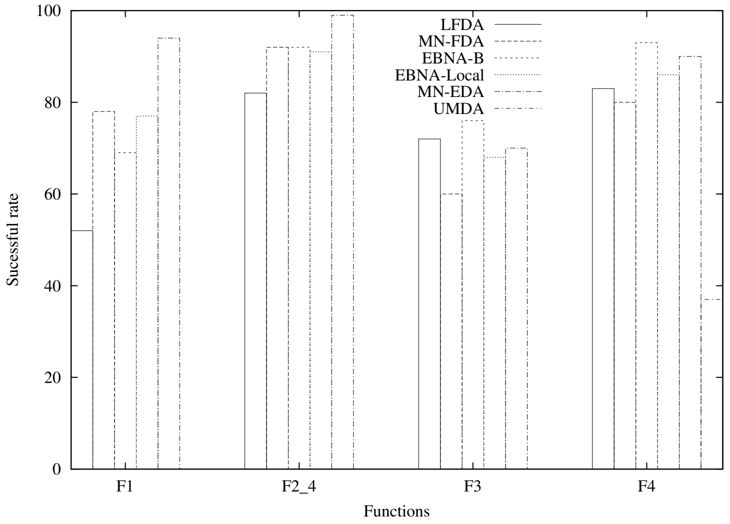

Figure 3: Successful rate of different EDAs in the optimization of deceptive functions.

information about the interactions of the variables to the algorithm for learning the Kikuchi approximation.

I	MN-EDA		MN-EDA^f		MN-FDA		EBNA$_{\mathrm{K2}}$		MT-FDA	
	S	$\bar{e}$	S	$\bar{e}$	S	$\bar{e}$	S	$\bar{e}$	S	$\bar{e}$
I_1^{16}	100	362.4	100	127.4	100	219.5	96	149.5	100	176.9
I_2^{16}	100	910.0	98	362.7	100	885.0	90	924.6	97	653.2
I_3^{16}	90	587.3	96	206.2	93	774.8	91	525.0	98	380.5
I_4^{16}	100	378.2	100	130.4	100	246.5	93	139.0	100	176.9
I_1^{36}	93	4496.7	97	1970.7	90	4939.4	92	6089.0	91	4173.9
I_2^{36}	90	3967.7	97	1733.4	90	5230.0	94	7142.8	91	3658.5
I_3^{36}	97	2799.5	93	830.3	92	2924.1	91	2881.7	93	2588.1
I_4^{36}	91	3959.6	94	1737.7	90	6074.2	91	7202.6	93	4490.3
I_1^{64}	98	8956.8	100	3838.3	94	11376.8	93	13793.7	91	9908.1
I_2^{64}	91	11251.7	95	11649.5	91	21071.0	93	25702.3	90	19235.8
I_3^{64}	91	10012.8	95	4607.6	91	16188.9	94	18834.6	90	12574.3
I_4^{64}	94	10701.9	90	4347.8	90	15095.1	99	30687.7	94	16213.8

Table 3: Successful rate and average number of function evaluations of EDAs for different Ising instances.

Four random instances of the Ising model were generated with $n \in \{16, 36, 64\}$. To generate a random instance where $J_{ij} \in \{-1, 1\}$, each coupling was set to -1 with probability 0.5, otherwise the constant was set to $+1$. The results were verified using the Spin Glass Ground State server, provided by the group of Prof. Michael Juenger[1]. We found the population size needed by five different EDAs to achieve a successful rate of 90 percent in 100 experiments. We compared the results of MN-EDA, MN-FDA,

[1] www.informatik.uni-koeln.de/ls_juenger/projects/sgs.html

MN-EDA^f, MT-FDA with 2 trees, and EBNA$_{K2}$. Experiments for LFDA and BOA were not conducted. EBNA$_{K2}$ achieved the best results among all the EBNA algorithms, and therefore the results for the other EBNA algorithms were not included either. For the Ising model problem, MN-EDA^f learns the clique based decomposition straight from the grid. The clique based graph decomposition will include all the edges of the graph and their intersections. The Kikuchi approximation calculated from this graph decomposition is equivalent to the Bethe approximation commonly used in statistical physics (Morita, 1994; Yedidia et al., 2001; Yedidia et al., 2002).

Table 3 presents the successful rate and average function evaluations of each algorithm for the 12 instances. The results are summarized in table 4, where we have counted the number of times each algorithm has ranked in the five possible positions. Ranking is based on the average number of evaluations. The best algorithm is by far MN-EDA^f, it ranks the first for all the experiments except one, where it ranks the second. This fact illustrates the capabilities of the Kikuchi approximation to incorporate information about the structure of the problem, even if the interactions among the variables form many cycles, as is the case of the Ising model.

The performance of the rest of EDAs is related to the size of the problems. For $n = 16$, MT-FDA and EBNA$_{K2}$ are the best algorithms. For $n = 36$, MT-FDA keeps the lead ahead of MN-EDA. For $n = 64$, MN-EDA outperforms the rest of the algorithms. The relatively good results achieved by MT-FDA can be explained by the presence of symmetry in the Ising problem. By clustering the population of solutions, and learning one specialized model for each cluster, a mixture of trees can deal with some types of symmetric problems. But when the number of variables is increased, more components would be needed to keep the same successful rate. To deal with the Ising problem in a satisfactory way, EDAs have to be able to maintain a high diversity in the population. The competence of MN-EDA to deal with this problem could be improved by incorporating techniques used for this purpose in GAs, for example niching methods.

	I^{16}					I^{36}					I^{64}					Tot.				
Alg. \ Ranks	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
MN-EDA^f	4	0	0	0	0	4	0	0	0	0	3	1	0	0	0	11	1	0	0	0
MN-EDA	0	0	0	2	2	0	1	3	0	0	1	3	0	0	0	1	4	3	2	2
MN-FDA	0	0	1	2	1	0	0	0	3	1	0	0	1	3	0	0	0	2	8	2
MT-FDA	0	2	2	0	0	0	3	1	0	0	0	0	3	1	0	0	5	6	1	0
EBNA$_{K2}$	0	2	1	0	1	0	0	0	1	3	0	0	0	0	4	0	2	1	1	8

Table 4: Summary of the results achieved by EDAs in the optimization of different Ising instances.

8.3.3 Functions with building block interdependency and frustration

In the following experiment we test MN-EDA in the optimization of two difficult functions. The HIFF (21) and $Cuban5$ (20) functions. Hierarchically decomposable functions, like HIFF, serve to model problems with building block interdependency (Watson et al., 1998). To optimize these functions, an EDA has to be able to correctly combine subsolutions corresponding to different optimal solutions. $Cuban5$ was introduced and studied in (Mühlenbein et al., 1999), where it was shown to be a very difficult non-separable additive function.

We calculate first an upper bound of the sufficient population size needed by MN-

EDA	$HIFF$					$Cuban5$				
	n	N	S	$\bar{g}$	$\bar{e}$	n	N	S	$\bar{g}$	$\bar{e}$
MN-EDA	32	700	30	4.90	3426.1	37	1850	30	6.40	11834.6
	64	1300	30	9.43	12254.9	69	9600	30	9.23	88631.8
	128	4300	30	16.30	69931.4	101	23600	10	11.90	280829.0
LFDA	32	700	30	4.86	3402.8	37	1850	27	5.85	10830.9
	64	1300	26	8.5	11735.3	69	9600	27	8.70	83547.9
	128	4300	29	12.27	52774.9	101	23600	10	10.90	257230.0
BOA	32	700	30	4.1	2865.9	37	1850	20	6.85	12678.8
	64	1300	26	8.5	11041.9	69	9600	23	9.65	92651.2
	128	4300	20	15.0	64485.0	101	23600	10	12.4	292627.0

Table 5: Results of MN-EDA, LFDA, and BOA in the optimization of functions HIFF and Cuban5.

EDA to optimize these functions in 30 consecutive experiments. In the case of the $Cuban5$, $n = 101$, and due to the large population needed to optimize this function, only 10 successful experiments are required for determining the sufficient population size. With the population sizes found, we run BOA and LFDA and calculate their successful rates and average number of evaluations. Results are shown in table 5. In only two of the six experiments LFDA and BOA could reach the same successful rate as MN-EDA, although the average number of functions evaluations of the MN-EDA is higher than that of the LFDA in most of the cases. For function HIFF, $n = 32$, MN-EDA is outperformed by both Bayesian network based FDAs. In the case of function $Cuban5$, $n = 101$, MN-EDA is the second behind LFDA. The results show that as well as for difficult functions with many variables, MN-EDA is an alternative to Bayesian network based FDAs.

8.4 Population size experiments

In the following experiments we investigate MN-EDA scalability. For all the functions analyzed, the critical population size for which the algorithm converges to the optimum in 90 of 100 executions is found. First, we compare the scalability of MN-EDA and other EDAs for function $f_{3deceptive}$. We use the following algorithms: EBNA$_B$, EBNA$_{local}$, and EBNA$_{K2}$. Additionally, we investigate MN-FDA scalability. We set a maximum population size of 6000. If an algorithm reaches this population size without converging, we stop the simulation. Figures 4 a) and b) respectively show the scaling of the population size and the average number of function evaluations for $f_{3deceptive}$.

It can be seen that the worst behavior is achieved by MN-FDA. This algorithm cannot optimize the function $f_{3deceptive}$, $n = 82$, with less than 6000 individuals. On the other extreme is MN-EDA. To optimize this function for $n = 120$, the algorithm needs a population size of $N = 5650$, and an average of evaluations $\bar{e} = 74151.6$. All the Bayesian network based FDAs have very similar scalability, they perform better than MN-FDA, and worse than MN-EDA.

We now investigate the effect of increasing the order of the additive function in the scalability of MN-EDA. We use the function $f_{deceptivek}$, and the same setting as in the previous experiment. Figures 5 a) and b) show respectively the population size, and number of evaluations for $k \in \{3, 4, 5\}$. To evaluate the increment in the complexity of the algorithm, we have found the coefficients of the linear polynomials $\beta log(n) + \gamma$ that

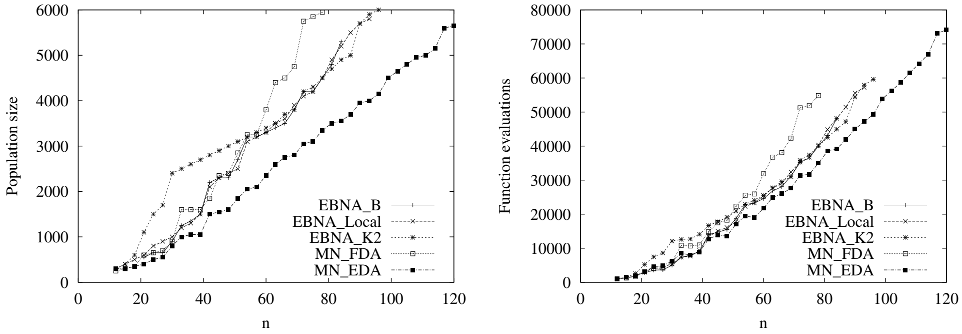

Figure 4: Scalability of MN-EDA, MN-FDA and Bayesian network based FDAs for function $f_{3deceptive}$

best fit the curves $(log(n), log(\bar{e}))$. The approximations are the following:

$$
\begin{aligned}
log(\bar{e}_{fdeceptive3}) &= 1.921 log(n) + 0.875 \\
log(\bar{e}_{fdeceptive4}) &= 2.060 log(n) + 0.916 \\
log(\bar{e}_{fdeceptive5}) &= 2.440 log(n) + 0.433
\end{aligned}
$$

The change in the values of β indicate the order of the increment in the number of function evaluations when the order of the functions are increased. As expected, the population size and the number of evaluations grow exponentially with the order of the function.

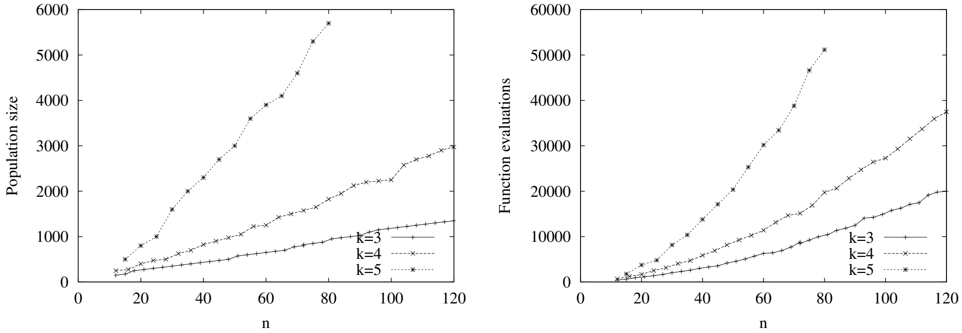

Figure 5: Scalability of MN-EDA for function $f_{deceptivek}$, $k \in \{3, 4, 5\}$

9 Preliminary conclusions and further work

The main contribution of this paper is in presenting MN-EDA. The algorithm uses the Kikuchi approximation to estimate the probability, and GS to sample it. Compared to previous EDAs, these are entirely different approaches to approximate and sample the probability distributions. We have provided a systematic empirical evaluation of MN-EDA, comparing it with many of the most referenced EDAs, and for different difficult functions. The performance displayed by MN-EDA shows that, in the context of EDAs, probabilistic models based on undirected graphs, and learned using tests of independence, it can be an alternative to Bayesian networks learned using score and search methods. Additionally, we have theoretically derived a number of properties of the Kikuchi approximation.

We have constrained our analysis of factorizations to the case of undirected models. However, in the space of directed models, there exists a class of models that can represent cycles in the interactions among the variables. They are called consistent dependency networks (Heckerman et al., 2000). Consistent dependency networks and MRFs are interchangeable representations. These directed networks have not been investigated in the context of EDAs. Recently, Mühlenbein and Mahnig (2002) have pointed out the convenience of using other procedures for the estimation of the Boltzmann distribution. Iterative proportional fitting, and advanced mean fields, have been among the methods proposed as promising for the estimation of probability distributions in EDAs (Mühlenbein and Mahnig, 2002).

Our work is also connected with recent research on approximate algorithms for belief propagation (Yedidia et al., 2001; Yedidia et al., 2002; Wainwright et al., 2001). While most of the work on belief propagation is concerned with computing marginal probabilities, we focus on sampling from the approximate distributions. Results from the study of belief approximation algorithms can be useful to understand for which types of probability distributions good estimators based on the Kikuchi approximation can be found. Many improvements can be made to MN-EDA. Here we will enumerate some of the areas worth researching further:

1. Preliminary results show that the use of niching methods, such as restricted tournament replacement, can accelerate the convergence of the algorithm. The suitability of incorporating these methods to the algorithm is left to further research

2. In many cases, during the evolution of MN-EDA, the independence graph found is chordal. In these cases the clique based decomposition will correspond to a junction tree, and less costly probabilistic logic sampling can be used instead of GS. An MN-EDA able to switch the type of sampling method used, according to the characteristics of the graph, will certainly be more efficient.

3. The expected waiting time of the GS for moves from states of high probability to states of lower probability can be very long. One way of speeding the mixing rate of the Markov chain is to use non local update Markov chain Monte Carlo methods that change the value of a block of variables instead of updating only one site of the vector. In (Santana and Mühlenbein, 2002) we present results about the use of Blocked Gibbs Sampling (BGS) in optimization. The use of BGS can be incorporated into MN-EDA to accelerate the convergence of the sampling algorithm.

10 Acknowledgments

The author thanks to Heinz Mühlenbein, Robin Höns, Li-Vang Lozada-Chang, two anonymous referees, and the editors of this special issue for their valuable comments on the reading that contributed to improve previous versions of the paper.

References

Bomze, I., Budinich, M., Pardalos, P., and Pelillo, M. (1999). The maximum clique problem. In Du, D.-Z. and Pardalos, P. M., editors, *Handbook of Combinatorial Opti-*

mization (supp. Vol. A), volume 4, pages 1–74. Kluwer Academic Publishers, Boston, MA.

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

Etxeberria, R. and Larrañaga, P. (1999). Global optimization using Bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 151–173, Habana, Cuba.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images. *IEEE transactions on pattern analysis and Machine Intelligence*, (6):721–741.

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

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadie, C. M. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75.

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

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

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

Larrañaga, P. and Lozano, J. A. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Optimization*. Kluwer Academic Publishers, Boston/Dordrecht/London.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford:Clarendon Press.

Morita, T. (1994). Formal structure of the cluster variation method. *Progressive Theoretical Physics Supplements*, 115:27–39.

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

Mühlenbein, H. and Mahnig, T. (2001). Evolutionary synthesis of Bayesian networks for optimization. *Advances in Evolutionary Synthesis of Neural Systems, MIT Press*, pages 429–455.

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

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

Mühlenbein, H. and Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, Berlin. Springer Verlag. LNCS 1141.

Ochoa, A., Soto, M. R., Santana, R., Madera, J. C., and Jorge, N. (1999). The Factorized Distribution Algorithm and the junction tree: A learning perspective. In Ochoa, A., Soto, M. R., and Santana, R., editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 368–377, Habana, Cuba.

Pakzad, P. and Anantharam, V. (2002). Belief propagation and statistical physics. In *Electronic Proceedings of 2002 Conference on Information Sciences and Systems*, Princeton University. Paper No.225, CD-ROM, 3 pages.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian Optimization Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume I, pages 525–532, Orlando, FL. Morgan Kaufmann Publishers, San Francisco, CA.

Santana, R. (2003). 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. Springer-Verlag.

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

Santana, R., Ochoa, A., and Soto, M. R. (2001). The Mixture of Trees Factorized Distribution Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pages 543–550, San Francisco, CA. Morgan Kaufmann Publishers.

Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction and Search*, volume 81 of *Lecture Notes in Statistics*. Springer-Verlag, New York.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2001). Tree-based reparameterization framework for analysis of belief propagation and related algorithms. Technical Report LIDS Technical Report P-2510, Laboratory for Information and Decision Systems, MIT.

Watson, R. A., Hornby, G. S., and Pollack, J. B. (1998). Modeling building-block interdependency. In *Parallel Problem Solving from Nature - PPSN V International Conference*, pages 97–106, Amsterdam, The Netherlands. Springer Verlag. LNCS 1498.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories.

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