Mixtures of Kikuchi Approximations

Roberto Santana, Pedro Larrañaga, and Jose A. Lozano

Intelligent System Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country, 20080, San Sebastián, Spain
{rsantana, pedro.larranaga, ja.lozano}@ehu.es

Abstract. Mixtures of distributions concern modeling a probability distribution by a weighted sum of other distributions. Kikuchi approximations of probability distributions follow an approach to approximate the free energy of statistical systems. In this paper, we introduce the mixture of Kikuchi approximations as a probability model. We present an algorithm for learning Kikuchi approximations from data based on the expectation-maximization (EM) paradigm. The proposal is tested in the approximation of probability distributions that arise in evolutionary computation.

Keywords: Mixture of distributions, Kikuchi approximations, estimation of distribution algorithms, EM.

1 Introduction

Probabilistic modeling by finite mixture of distributions [8] concerns modeling a statistical distribution by a mixture (or weighted sum) of other distributions. Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote a vector of discrete random variables, and $\mathbf{x} = (x_1, \ldots, x_n)$ denote an assignment to the variables. A mixture $q^m(\mathbf{x})$ of distributions $p_j(\mathbf{x})$ is defined to be a distribution of the form:

$$q^m(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j p_j(\mathbf{x}) \tag{1}$$

with $\lambda_j > 0$, $j = 1, \ldots, m$, $\sum_{j=1}^{m} \lambda_j = 1$.

The $p_j(\mathbf{x})$ are called mixture components, and the λ_j are called mixture coefficients. m is the number of components of the mixture. A mixture of distributions can be viewed as containing an unobserved choice variable Z which takes value $j \in \{1, \ldots, m\}$ with probability $p(Z = j) = \lambda_j$. In some cases the choice variable Z is known.

Probabilistic modeling based on mixtures of distributions has been used in many domains. Two of the most frequent applications are data clustering and approximation of probability distributions [8]. Mixtures are specially suited for modeling problems that exhibit complex interactions between their variables.

Much research has gone into elucidating the properties of mixture distributions as well as into designing efficient algorithms to learn them. One important

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): ECML 2006, LNAI 4212, pp. 365–376, 2006.

issue is whether the capacity of mixtures for probabilistic modeling can be enhanced by considering as components of the mixture probability distributions able to represent a higher number of dependencies of the data. Mixtures of mean field distributions [1] are one example of the way this type of models can improve the results achieved with simple factorial models.

Recently, the research on probability distribution approximation has widened its scope by the emergence of approximation methods inspired on region-based decompositions of the free energy [16]. These methods have been applied in the context of probabilistic modeling for classification [5] and evolutionary computation [13], where algorithms for learning Kikuchi approximations from data have been introduced.

One of these methods is the Kikuchi approximation of a probability that uses clique-based decomposition of independence graphs [13,14]. Kikuchi approximations can represent complex interactions between the variables of a problem. This provides our motivation to define a class of mixture of Kikuchi approximations and an algorithm to learn this approximation from data. The main goal in tackling this problem is to combine the capacity of mixtures to exploit asymmetric independence assertions with the power of Kikuchi approximations to represent complex interactions. To evaluate the efficiency of this model, we apply it in the context of function optimization by means of estimation of distribution algorithms (EDAs) [7,10] and for unsupervised learning. EDAs are optimization algorithms that explicitly model probabilistic dependencies between variables of the problem domain to make an efficient search for optimal solutions.

The remainder part of the paper is ordered as follows. In the next section, the Kikuchi approximation defined on clique-based decompositions is presented. Section 3 introduces the mixture of Kikuchi approximations and an algorithm for learning these approximation from data. Section 4 briefly explains EDAs and introduces an EDA that uses mixtures of Kikuchi approximations. Section 5 presents a number of experiments that analyze the dynamics of the introduced learning algorithm, and the effect of using the mixture of Kikuchi approximations as the probability model in EDAs. The conclusions of our paper are presented in Section 6.

2 Kikuchi Approximation: Recapitulation

Kikuchi approximations of the free energy [6] are region-based decompositions of the free energy that satisfy certain constraints. The Kikuchi approximation of a probability distribution from a clique-based decomposition of an independence graph [13] is a particular type of factorization in probability marginals. The marginals in the factorization are completely determined by the independence graph. Given this graph, the clique-based decomposition is formed by the maximal cliques of the graphs and their intersections. All these cliques are called regions. More formally: let S denote a set of indices in $N = \{1, \dots, n\}$, and $\mathbf{X}_S$ (respectively $\mathbf{x}_S$) a subset of the variables of $\mathbf{X}$ (respectively a subset of values of $\mathbf{x}$) determined by the indices in S. We will work with positive probability distributions denoted by $p(\mathbf{x})$. Similarly, $p(\mathbf{x}_S)$ will denote the marginal probability for $\mathbf{X}_S$. We use $p(x_i \mid x_j)$ to denote the conditional probability distribution of X_i given $X_j = x_j$.

Given a probability distribution $p(\mathbf{x})$, its independence graph $G = (V, E)$ associates one vertex with every variable of $\mathbf{X}$, and two vertices are connected if the corresponding variables are conditionally dependent given the rest of the variables. We define a region R of the independence graph $G = (V, E)$ of a probability distribution $p(\mathbf{x})$ as a subset of V. A graph region-based decomposition $(\mathcal{R}, U)$, is a set of regions $\mathcal{R}$ that covers all the V, and an associated set of *overcounting numbers* U which is formed by assigning one overcounting number c_R for each $R \in \mathcal{R}$. c_R will always be an integer, and might be zero or negative for some R.

To find a region-based decomposition, the cluster variation method (CVM) can be used [6,16]. In CVM, $\mathcal{R}$ is formed recursively 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 the intersection operation, and can be ordered as a partially ordered set.

In a clique-based decomposition the CVM is applied making a particular choice of the initial regions. The set $\mathcal{R}_0$ is formed 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 define the Kikuchi approximation of the probability distribution $p(\mathbf{x})$ associated with a clique-based decomposition, $k(\mathbf{x})$ as:

$$k(\mathbf{x}) = \prod_{R \in \mathcal{R}} p(\mathbf{x}_R)^{c_R} \tag{2}$$

where $\mathcal{R}$ comes from a clique-based decomposition and the overcounting numbers c_R are calculated using the following recursive formula:

$$c_R = 1 - \sum_{\substack{S \in \mathcal{R} \\ R \subset S}} c_S \tag{3}$$

where c_S is the overcounting number of any region S in $\mathcal{R}$ such that S is a superset of R. c_R values corresponding to the initial maximal cliques are equal to 1. If c_R is different from zero, the region is included in the clique-based decomposition.

From now on, when we refer to a Kikuchi approximation, we imply a Kikuchi approximation obtained from a clique-based decomposition. The Kikuchi approximation has a number of convenient properties for approximating distributions. If the independence graph is chordal, the Kikuchi approximation calculated from a clique-based decomposition corresponds to an exact factorization of the probability distribution calculated from a junction tree of the independence graph. The Kikuchi approximation also satisfies a number of Markov and decomposability properties [14].

3 Mixture of Kikuchi Approximations

In this section, we define the mixture of Kikuchi approximations and present an algorithm for learning this type of model from data.

Definition 1. *A mixture of Kikuchi approximations is defined to be an approximation of the form:*

$$k^m(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j k_j(\mathbf{x}) \tag{4}$$

with $\lambda_j > 0$*,* $j = 1, \ldots, m$*,* $\sum_{j=1}^{m} \lambda_j = 1$*.*

The components of $k^m(\mathbf{x})$ are Kikuchi approximations. Since Kikuchi approximations are not probability distributions in general, the mixture of Kikuchi approximations is not either. However, notice that whenever the clique-based decompositions correspond to chordal graphs, each component of the mixture will be a junction tree, and $k^m(\mathbf{x})$ will be a probability distribution. In fact, a mixture of Kikuchi approximations opens the possibility of combining components that are probability distributions with other that are not.

Approaches used to learn mixtures of distributions include [8]: graphical methods, minimum-distance methods, maximum likelihood, methods of moments, and Bayesian approaches. If the choice variable is not observed, one of the alternatives that can be used for learning the structure and parameters of the components is the EM algorithm [4], that looks for a mixture that maximizes the likelihood of the data. The iterative EM algorithm is a general, usually reliable numerical method for obtaining maximum likelihood (or Bayesian maximum a posteriori) estimates of parameters in incomplete-data contexts.

To learn a mixture of Kikuchi approximations, we propose to use a version of the EM algorithm. The general scheme is similar to the procedure used to learn mixture of trees [9]. However, fundamental differences arise in the expectation and maximization steps. The learning problem can be established as: Given a set of observations $D = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$, we are required to find the mixture of Kikuchi approximations $k^m(\mathbf{x})$ that satisfies

$$k^m(\mathbf{x}) = \arg \max_{k'^m(\mathbf{x})} \sum_{i=1}^{N} \log k'^m(\mathbf{x}^i) \tag{5}$$

Within the framework of the EM algorithm, expresion (5) is commonly referred as the incomplete log-likelihood of the data given a probability distribution. Since Kikuchi and mixture of Kikuchi approximations are not probability distributions in general, in these cases this expression will not correspond to the incomplete log-likelihood. Nevertheless, we will use the right term in equation (5) as a measure of the accuracy of the approximation given by the mixture of Kikuchi approximations.

In the EM algorithm, the complete likelihood is defined as the log-likelihood of both, the observed and the unobserved data, given the current model estimate $k^m(\mathbf{x})$. A version of the complete log-likelihood for the mixture of Kikuchi approximations is shown in equation (6).

$$\begin{aligned}\mathcal{L}(\mathbf{x}^1,\ldots,\mathbf{x}^N,z^1,\ldots,z^N|k^m(\mathbf{x})) &= \sum_{i=1}^{N} log \prod_{j=1}^{m} \left(\lambda_j k_j(\mathbf{x}^i)\right)^{\delta_{j,z^i}} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{m} \delta_{j,z^i}(\log\lambda_j + \log k_j(\mathbf{x}^i)) \qquad (6)\end{aligned}$$

where δ_{j,z^i} is equal to one if z^i is equal to the j^{th} value of the choice variable, and zero otherwise.

By maximimizing (6), the mixture Kikuchi-EM learning algorithm pursues to indirectly find a solution to the maximization problem defined by equation (5). The idea underlying the Kikuchi-EM algorithm is to compute and optimize the expected value of $\mathcal{L}(\mathbf{x}^1,\ldots,\mathbf{x}^N,z^1,\ldots,z^N|k^m(\mathbf{x}))$. However, the way the expectation and maximization steps are implemented is very particular. In the expectation step of the mixture Kikuchi-EM, we use Kikuchi approximations values for estimating the posterior probability of the hidden variable for each of the observations. In our case, this means estimating the probability of each component of the mixture generating data point $\mathbf{x}^i$.

$$p(Z^i = j|\mathbf{x}^i, k^m(\mathbf{x})) = \gamma_j(\mathbf{x}^i) = \frac{\lambda_j k_j(\mathbf{x}^i)}{\sum_{j'} \lambda_{j'} k^{j'}(\mathbf{x}^i)} \qquad (7)$$

One uses these posterior probabilities to compute the expectation of $\mathcal{L}$, which is a linear function of the $\gamma_j(\mathbf{x})$ values. Let us introduce the following quantities:

$$\Gamma_j = \sum_{i=1}^{N} \gamma_j(\mathbf{x}^i),\ j = 1,\ldots,m \qquad (8)$$

$$q^j(\mathbf{x}^i) = \frac{\gamma_j(\mathbf{x}^i)}{\Gamma_j} \qquad (9)$$

The sums $\Gamma_j \in [0,N]$ can be interpreted as the total number of data points that are generated by the j-th component. By normalizing the posteriors $\gamma_j(\mathbf{x}^i)$ with Γ_j we obtain a probability distribution $q^j(\mathbf{x}^i)$ over the data set. Notice that even if $k_j(\mathbf{x})$ is not a probability distribution, $q^j(\mathbf{x})$ is a probability distribution because it is the result of normalization.

The maximization step of the Kikuchi-EM algorithm looks for estimating the parameters of the model so as to maximize $E[\mathcal{L}(\mathbf{x}^1,\ldots,\mathbf{x}^N|k^m(\mathbf{x}))]$. Consequently, it is necessary to obtain the model that best fits the data in each component. In the case of the mixtures of trees, this problem can be solved using an algorithm that is guaranteed to give the best structure [9]. For Kikuchi approximations, we use a learning algorithm introduced in [13]. This algorithm serves for searching in the space of the Kikuchi approximations, but the optimum is not guaranteed to be found.

The algorithm learns an independence graph from the data and finds its clique-based decomposition. To learn the independence graph, independence tests are used. We use the Chi-square independence test. If for two variables X_i and X_j, we reject the null hypothesis of independence with a specified level of significance α, they are joined by an edge. The pseudocode of the Kikuchi-EM learning algorithm is shown in Algorithm 1. As a method for constructing the initial mixture of Kikuchi approximations, we propose a heuristic approach in which the learning algorithm proposed in [13] is used to learn each initial Kikuchi approximation component from the data using a different value of parameter α for each component. The termination criterion used is that the difference between the likelihood achieved at iterations $t + 1$ and t is below a given threshold.

Algorithm 1. Mixture of Kikuchi EM

```
t ← 1; Set an initial mixture of Kikuchi approximations
do {
    for $j \Leftarrow 1$ to $m$
        for $i \Leftarrow 1$ to $N$
            Compute $\gamma_j(\mathbf{x}^i)$, $\Gamma_j$, $q^j(\mathbf{x}^i)$ using equations (7), (8) and (9) respectively
    for $j \Leftarrow 1$ to $m$
        Compute the Kikuchi approximation $k_j(\mathbf{x})$ of $q^j(\mathbf{x})$
        $\lambda_j = \frac{\Gamma_j}{N}$
    $t \leftarrow t + 1$
} until Termination criteria met
```

4 Application Domain: The Mixture of Kikuchi Approximations EDA

The mixture of Kikuchi approximations can be used in classification and in the approximation of distributions. In this section, we will describe an application of mixtures of Kikuchi approximations to function optimization by means of EDAs [7].

The goal of EDAs is function optimization. One essential assumption of these algorithms is that it is possible to build a probabilistic model of the search space that can be used to guide the search for the optimum. The probabilistic model can be built using available information about the function or learned from samples.

EDAs work with a set (or population) of points. Initially, a random sample of points is generated. These points are evaluated using the function, and a subset of points is selected based on this evaluation. Usually, points with higher evaluation has a higher probability of being selected. A probabilistic model of the selected solutions is built, and the model is sampled to obtain a new set of points. The process iterates until the optimum has been found or another termination criterion is fulfilled. A key characteristic and crucial step of EDAs is the construction of the probabilistic model. These models may differ in the order and number of the probabilistic dependencies that they represent.

Applications of mixtures of distributions in EDAs include the use of mixtures of Gaussian models for the solution of multiobjective continuous problems [2], the application of mixtures of Bayesian models [11] for clustering in continuous optimization, and the use of mixtures of trees [15] in discrete optimization. In Algorithm 2, the pseudo-code of the mixture of Kikuchi approximations EDA (MKA-EDA) is presented.

Algorithm 2. Mixture of Kikuchi approximations EDA

```
Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly
do {
    Evaluate the points using the fitness function
    Select a set $S$ of $M \leq N$ points according to a selection method
    Calculate a mixture of Kikuchi approximations $Q(\mathbf{x})$ using a learning
    algorithm
    Generate new points sampling from $Q(\mathbf{x})$
    $t \Leftarrow t + 1$
} until Termination criteria are met
```

Algorithm 2 uses Kikuchi-EM algorithm to learn the model. To generate points from the Kikuchi approximations, a Gibbs sampling algorithm introduced in [13] is used. The selection method is truncation selection of parameter T, in which the $M = NT$ points with best function evaluation are selected.

The computational cost of MK-EDA depends on the cost of the algorithms used to learn and sample the Kikuchi approximation plus the cost of the EM algorithm. The complexity of learning the parameters depends n, N, the number of cliques μ, and their size. The order of this steps is $O(N\mu) \approx O(Nn2)$. The total complexity of the learning algorithm is roughly estimated as $O(Nn3)$.

5 Experiments

We start this section by presenting the optimization problem and instances used in our experiments. The following experiments study the dynamics of the Kikuchi-EM learning algorithm and compare it with other probabilistic models. Finally, the section shows the results of the MKA-EDA.

The satisfiability (SAT) problem consists in finding an assignment of values to a set of n boolean variables such that they satisfy a given set of clauses $c_1, c_2, \ldots, c_r$, where c_i is a disjunction of literals, and a literal is a variable or its negation. The restriction of SAT to instances where all clauses have length 3 is denoted 3-SAT. This problem is NP-complete [3]. In our representation, a variable X_i is associated to each boolean variable. As the objective function, we use the sum of clauses satisfied by the solution.

The selected problems benchmark is composed by three sets of difficult instances. These instances are contained in the files *uf20-91*, *aim-50-3-4-yes1-j* and *aim-100-3-4-yes1-j*[1]. The *uf20-91* file contains 1000 instances, all have 20 variables and 91 clauses, all are satisfiable. File *aim-50-3-4-yes1-j* and *aim-100-3-4-yes1-j* contain four instance each. The number of the variables of instances in these two files are, respectively, 50 and 100. The number of clauses are, respectively, 170 and 340.

In a preprocessing step, the *uf20-91* instances where classified in five groups according to the difficulty they pose for a very simple EDA. The criterion for classification was the EDA success rate in 100 runs. The most difficult class comprises to instances for which the simple EDA converged in less than 20 runs. This class includes 38 instances and was used to evaluate the performance of MKA-EDA.

5.1 Dynamics of the Kikuchi-EM Learning Algorithm

In the first experiment, we evaluate the behavior of the Kikuchi-EM learning algorithm in the approximation of the empirical probability distribution of data obtained from the optimization of the *aim-50-3-4-yes1-1* instance. A population size of 500 points and truncation selection with parameter $T = 0.1$ are used. First, all the solutions are evaluated, the first selected set of solutions is used for the experiment. The goal of the experiment is to evaluate the approximation achieved at each step of the Kikuchi-EM learning algorithm.

To evaluate the quality of the approximation we use a quality measure similar to the Kullback-Leibler divergence between the target empirical distribution and the learned mixture of Kikuchi approximations $D(p||k^m) = \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x})}{k^m(\mathbf{x})}$. Kullback-Leibler divergence is only defined between probability distributions. We employ the same measure but warning that as mixture of Kikuchi approximations are not in general probability distributions, the measure used does not fulfill a number of properties satisfied by the Kullback-Leibler divergence. Additionally, we measure the Kullback-Leibler divergence between the empirical distribution and the mixture of Kikuchi approximations normalized in the set of data $\bar{k}^m(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j q^j(\mathbf{x})$. Normalization of the Kikuchi approximation guarantees to obtain a probability distribution on the set of data. Therefore, the Kullback-Leibler divergence will be always non-negative in this case. Additionally, normalization is a required step of the EM method used to learn the Kikuchi approximations. In each step of the learning algorithm, the divergences are calculated from the current approximation learned by the model.

[1] All files, together with a description about the instances, are available from http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html

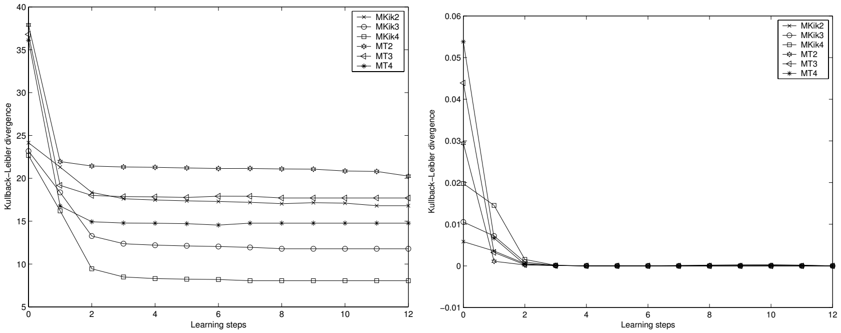

Fig. 1. Amount of decrease in the Kullback-Leibler divergence between the mixture probability model and the target probability during the learning process. Left: Kullback-Leibler divergence has been calculated from the mixture. Right: Kullback-Leibler divergence has been calculated from the normalized mixture.

Figure 1 shows the results of this experiment for mixtures of Kikuchi approximations (MKik) with different number of components. In the figure, we have included the results obtained using the mixture of trees (MT) EM learning algorithm. For every mixture, the results are the average from 1000 runs. As learning may take different number of steps for each selected population, the average of the divergence at step i is calculated from all the experiments that reach step i. It can be seen in Figure 1 that during the first steps of the learning algorithm the divergence to the empirical distribution decreases. The best results are achieved for the mixtures of Kikuchi approximations of 4 components (MKik4). For both types of mixtures, the divergence to the target probability decreases with the increase in the number of components. It can be appreciated that MT4 outperforms the behavior of MKik2.

We use the information collected from these experiments to calculate the gain in the approximation measured as the difference between the Kullback-Leibler obtained in the last and second iterations of the learning algorithms. Results are shown in Figure 2. The value of divergence at the second iteration is taken as a reference because the algorithm for learning mixtures of trees starts for a complete random initialization, while the heuristic method used to initialize the Kikuchi-EM learning algorithm takes profit of the information contained in the data. It can be appreciated in the figure that also the gain achieved by the Kikuchi-EM learning algorithm is higher than for the mixtures of trees.

Table 1. Percentage of success of MN-EDA and MKA-EDA for a set of difficult *uf20-91* instances

Alg.	MN-EDA	MKA-EDA$_2$	MKA-EDA$_3$	MKA-EDA$_4$	MKA-EDA$_5$
	72.78	84.11	78.63	75.84	72.96

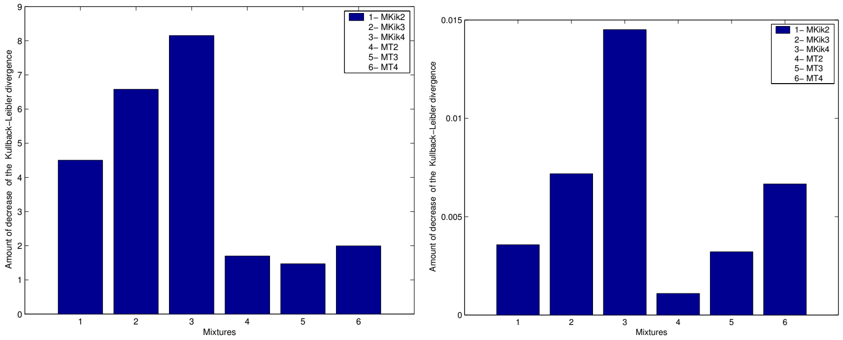

Fig. 2. Amount of decrease in the Kullback-Leibler divergence between the mixture probability model and the target probability after the learning procedure is complete. Left: Kullback-Leibler divergence has been calculated from the mixture. Right: Kullback-Leibler divergence has been calculated from the normalized mixture.

5.2 Results of MKA-EDA

In the following experiments we investigate the ability of MKA-EDA to find the solution of the SAT problem. In an initial experiment, we compare the performance of MKA-EDA with an EDA based on Kikuchi aproximations (MN-EDA)[13]. MN-EDA uses a probabilistic model based on a single Kikuchi approximation. We calculate the average success of MN-EDA and MKA-EDA with different number of components for a set of difficult *uf20-91* instances. MN-EDA and MKA-EDA use a population size $N = 500$, a maximum of 25 generations, and the parameter of truncation was $T = 0.15$.

Table 1 shows the results of experiments. Notice the improvement in the results achieved by MKA-EDA compared to those obtained by MN-EDA. When the number of components of the mixture of Kikuchi approximations is increased results deteriorate. This fact may be due to the overfitting problem. However, it is even better to use MKA-EDA with a mixture of five components than the MN-EDA.

In the next experiment, we evaluate the behavior of MKA-EDA for instances of higher size. We compare its performance to FDA, MN-FDA and MN-EDA which are EDA with an increasing complexity in their probability models [12,13]. We employ a very simple local optimization method to accelerate the convergence of all the EDAs. We apply this algorithm to every solution during the evaluation step. Except in one case, all the experiments were done for a population size $N = 1000$, parameter of truncation $T = 0.15$, and a maximum number of generations 50. The only exception is MKA-EDA$_4$. For this algorithm, $N = 5000$ and the maximum number of generations was 200. The goal of including MKA-EDA with these parameters was to evaluate the improvement in the convergence results when the population size is increased.

Table 2. Comparison among MN-EDAs for the aim instances of the SAT problem

		FDA		MN-FDA		MN-EDA		MKA-EDA$_2$		MKA-EDA$_4$	
n	mc	mb	v	mb	v	mb	v	mb	v	mb	v
50	170	169	10	170	2	170	1	170	2	170	10
50	170	170	6	170	7	170	9	170	9	170	10
50	170	170	7	170	8	170	9	170	10	170	10
50	170	170	9	170	9	170	10	170	10	170	10
100	340	337	1	336	6	337	2	337	1	336	4
100	340	337	1	337	2	337	4	337	3	340	1
100	340	336	7	336	7	336	9	336	6	336	10
100	340	340	2	340	3	340	2	340	2	340	6
Tot.			24		29		31		31		47

Table 2 shows the results achieved with the algorithms for instances *aim-50-3-4-yes1-j* and *aim-100-3-4-yes1-j*. In the table, mc is the number of clauses in each instance (optimum of the function), mb is the best value reached by the corresponding algorithm, and v is the number of times that the best found value was reached in 10 runs. The first rows correspond to the four instances of *aim-50-3-4-yes1-j*, and the next four to the four instances of *aim-100-3-4-yes1-j*. The last row shows the number of time the optimum was found in the 80 experiments.

It can be seen in Table 2 that MKA-EDA$_2$ achieves better or equal results than the rest of algorithms. For these intances however, the difference between results achieved by MN-EDA and MKA-EDA$_2$ is not statistically significant. The results of MKA-EDA can be improved by augmenting the number of components in the mixture, the population size, and the number of generations of the algorithm. However, determining the exact number of components for mixture of Kikuchi approximations is an open problem. Overfitting can arise and it is a general problem for Kikuchi and other mixture of distributions. On the other hand, EDAs based on trees and mixture of trees have good behavior for problems with low dependencies between the variables. Kikuchi and mixture of Kikuchi approximations outperform them for problems with more complex interactions.

6 Conclusions

In this paper, we have introduced a new class of probability models based on Kikuchi approximations of probability distributions. An algorithm has been introduced lo learn mixtures of Kikuchi approximations from data. The approximations learned by the algorithm can be more accurate, in terms of the Kullback-Leibler divergence, than other approximations based on mixtures of less complex components. The mixture of Kikuchi approximations combines the capacity of mixtures to exploit asymmetric independence assertions with the power of Kikuchi approximations to handle complex interactions. We recommend its application to problems with very complex interactions that can not be represented with simpler model.

References

1. C. M. Bishop, N. Lawrence, T. Jaakkola, and M. I. Jordan. Approximating posterior distributions in belief networks using mixtures. In *Proc. Conf. Advances in Neural Information Processing Systems 10, NIPS*, pages 416–422. MIT Press, 1998.
2. P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.
3. S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 1971.
4. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B(39):1–38, 1977.
5. A. Jakulin, I. Rish, and I. Bratko. Kikuchi-Bayes: Factorized models for approximate classification in closed form. Technical Report RC23314 (WO408-175), IBM, August 2004.
6. R. Kikuchi. A theory of cooperative phenomena. *Physical Review*, 81(6):988–1003, 1951.
7. P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.
8. G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley & Sons, 2000.
9. M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.
10. 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*, pages 178–187, Berlin, 1996. Springer Verlag. LNCS 1141.
11. J. Peña, J. A. Lozano, and P. Larrañaga. Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. *Evolutionary Computation*, 13(1):43–66, 2005.
12. 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-Verlag.
13. R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.
14. R. Santana, P. Larrañaga, and J. A. Lozano. Properties of Kikuchi approximations constructed from clique based decompositions. Technical Report EHU-KZAA-IK-2/05, Department of Computer Science and Artificial Intelligence, University of the Basque Country, April 2005. Available from http://www.sc.ehu.es/ccwbayes/technical.htm.
15. R. Santana, A. Ochoa, and M. R. Soto. The mixture of trees factorized distribution algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pages 543–550, San Francisco, CA, 2001. Morgan Kaufmann Publishers.
16. J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2004-040, Mitsubishi Electric Research Laboratories, May 2004.