Improved EDNA (Estimation of Dependency Networks Algorithm) Using Combining Function with Bivariate Probability Distributions

José A. Gámez
jgamez@dsi.uclm.es

Juan L. Mateo
juanlmc@dsi.uclm.es

José M. Puerta
jpuerta@dsi.uclm.es

Computing Systems Department
Intelligent Systems and Data Mining Group – $i^3\mathcal{A}$
University of Castilla-La Mancha, Albacete, 02071, Spain

ABSTRACT

One of the key points in Estimation of Distribution Algorithms (EDAs) is the learning of the probabilistic graphical model used to guide the search: the richer the model the more complex the learning task. Dependency networks-based EDAs have been recently introduced. On the contrary of Bayesian networks, dependency networks allow the presence of directed cycles in their structure. In a previous work the authors proposed EDNA, an EDA algorithm in which a multivariate dependency network is used but approximating its structure learning by considering only bivariate statistics. EDNA was compared with other models from the literature with the same computational complexity (e.g., univariate and bivariate models). In this work we propose a modified version of EDNA in which not only the structural learning phase is limited to bivariate statistics, but also the simulation and the parameter learning task. Now, we extend the comparison employing multivariate models based on Bayesian networks (EBNA and hBOA). Our experiments show that the modified EDNA is more accurate than the original one, being its accuracy comparable to EBNA and hBOA, but with the advantage of being faster specially in the more complex cases.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Global Optimization*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Design.

Keywords

Estimation of Distribution Algorithms, Dependency Networks, Probability Combining Function, Combinatorial Optimization, Scalability

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. INTRODUCTION

Estimation of Distribution Algorithms (EDAs) [20] were proposed as an extension of genetic algorithms (GA) [14]. The key difference is that instead of using crossover and mutation operators, EDAs carry out the evolution by learning a probability distribution (PD) from the population which encodes the relationships between the individual components and sampling new individuals from that learned distribution. In the literature can be found different models which are used for learning such PD. They are grouped by its complexity or the order of dependencies allowed: univariate (e.g. UMDA [18] and cGA [10]); bivariate (e.g. MIMIC [2] and COMIT [1]); or multivariate.

Among multivariate models there are also several approaches. EcGA [8] is based on clustering components (genes or variables) in independent groups and learning a joint PD for each group. FDA [19] sets the dependencies at the beginning and in each iteration it only learns the conditional probabilities distributions (CPD). Other models use Bayesian networks (BNs) [24] in order to represent the dependencies in the population, so they use typical BNs learning algorithms [3, 11] to identify those dependencies. Two representative examples are EBNA [16] and hBOA [25]. Basically, the difference between these two models is that hBOA introduce the use of niching through the use of the Restricted Tournament Replacer (RTR) [9].

As in FDA the dependencies are established beforehand and in a specific way for each problem it loses generality with respect to BNs-based EDAs. On the other hand, EcGA assume that all variables in the same cluster are dependent among themselves what can lead to a very complex joint PD, meanwhile if in fact there are some conditional independencies between those variables, modeling them with a BN will be more efficient in terms of space needed to represent the PD and we can reduce overfitting if these PD are smaller.

Given that the use of BN has succeeded in this field, is fair thinking about using other kind of probabilistic graphical models like dependency networks (DN) [12]. That was the reason why EDNA [5] was developed claiming that, taking benefit from the properties of DN, the structure of this multivariate model can be approximated by using only statistics of order two.

This paper describes a modification to EDNA which improves notably its accuracy. This modification is based on reducing overfitting in parameter estimation due to the large conditional probability tables needed to learn in the original EDNA. Now, the model approximation is extended to the probabilistic graphical model parameter component, and instead of estimating n-dimensional conditional probability tables, we combine all conditional probability table for every single dependent variable by means of the *mean* gate.

This work starts by describing previously presented EDNA with the definition of dependency networks in Section 2. Section 3 is devoted to explain the concept of combining functions (gates) and our choice which is the mean function. The experiments carried out are presented and discussed in Section 4. Finally, in Section 5 the conclusions of this work are exposed.

2. EDNA

In this Section we are going to present the probabilistic graphical model we use as base for our proposal EDNA, but before it is necessary to introduce some material about DN.

2.1 Dependency Networks

Dependency networks (DNs) were proposed by Heckerman et al. [12] as an alternative to BNs. They can be defined as a tuple $(\mathbb{G}, \mathbf{P})$ over a domain $\mathbf{X}$ where $\mathbb{G}$ is a directed graph (not necessarily acyclic) and $\mathbf{P}$ is a set of CPDs, one for each variable in $\mathbf{X}$. Every $P \in \mathbf{P}$ must be such that

$$P(X_i|\mathbf{Pa}_i) = P(X_i|\mathbf{X} \setminus X_i).$$

This means that the set of parents $Pa(X_i)$ for every variable X_i is its Markov blanket $MB(X_i)$.

This definition requires *consistency* in the sense that the joint PD for $\mathbf{X}$ can be exactly recovered from $\mathbf{P}$. This is a very restrictive condition when learning from data, so in [12] the authors defined *general dependency networks* in order to relax the factorization: $P(\mathbf{X}) \approx \prod P(X_i|Pa_i)$.

It can be observed that a DN can be learned from data by independently learning the parent set for each variable, which quickly lead to the design of parallel learning algorithms. The fact of allowing directed cycles, although enlarges the representation issue has the disadvantage of avoiding the use of traditional BNs exact inference algorithms. In this case, Heckerman et al. [12] propose to use approximate inference carried out by using Gibbs sampling [6], instead of probabilistic logic sampling which is typically used in EDAs.

DNs have some advantages and disadvantages with respect BNs. For instance a DN is not usefull for encoding causal relationships and is dificult to build with a knowledge-based approach. Nonetheless in a scenario in which this model is built with automatic learning algorithms DNs present a great advantage because is easy and straightforward to make computationally efficient learning algorithms. In [12], authors propose some tasks in which DNs can be very suitable like *collaborative filtering* or *probabilistic inference*. We can find in the literature some examples of the use of DNs in different fields [21, 28, 4].

2.2 Multivariate relationship by Pairwise Computations

From the previous definition and with the idea of using a low complex algorithm when inducing the structure of the probabilistic graphical model, authors introduced EDNA ([5]). EDNA takes benefit from the lack of restrictions about acyclicity and carries out the structural learning for discovering the dependencies between variables[1] by using the following mutual information-based approximation:

$$Merit(X_i; X_j) = I(X_i; X_j) - \frac{\sum_{X_p \in \mathbf{Pa}_i} I(X_p; X_j)}{|\mathbf{Pa}_i| + 1},$$

which evaluates the goodness of adding X_j as a new parent of X_i taking into account the current set of parents of X_i ($\mathbf{Pa}_i$), where $I(\cdot;\cdot)$ is mutual information function (see [5] for details).

This function is not symmetrical, that is to say, at the end of the learning process X_j can be parent of X_i but not the opposite. Nonetheless is a heuristic way to approximate a multivariate computation by bivariate ones, reducing in this way the computational complexity specially when there are many dependencies.

Once we got the graphical structure of the DN, we learn the n-dimensional CPDs, getting in this way a fully specified multivariate probabilistic graphical model. However, later we have detected that there is a problem with this way of proceed: even if we assume that we have correctly recovered the dependencies in the population, the set of parents for every variable will be greater (or at least equal) than the equivalent BN model. That implies that because of the (usually) small considered training set (i.e. the best individuals in the population) the CPD for every variable will be more difficult to be properly estimated and can be in fact overestimated. This fact leads to a deterioration in the performance.

3. COMBINING FUNCTIONS

In order to solve the problem commented above and to improve EDNA we propose to extend the use of bivariate statistics from only structural learning to the whole process, including parameter learning and simulation. Thus, instead of learning a full CPD for every variable, we propose to estimate an independent CPD for each variable and each parent. Later, when the probability value of a variable given all their parents is required, we compute this value from the set of estimated bivariate CPDs (notice that we compute the value only for required configurations, not the whole CPD).

This approach is largely used in practice when constructing BNs [23], mostly because efficiency reasons, number of parameters to be elicited from experts, and/or in order to introduce a better way of designing the interaction between the variables involved. We can found examples in the literature applied to medical domains [22] or in the field of Relational Bayesian Networks [15]. The combination of the effect of each single parent to obtain the whole effect is carried out by means of a *gate* or *combining function*:

DEFINITION 1. *A combining function is a function that maps a finite set of probabilities distributions into a probability distribution.*

Perhaps the most known example of combining function is *noisy-OR*. This function is used to describe the interaction between an observable effect E and its n possible causes

[1]with variable we refer to every component/gene of the individual

$\{C_1, C_2, \cdots, C_n\}$. It is used for instance to model the relationship between an illness and the conditions that can cause that illness. It is supposed that each cause alone is sufficient to make true the effect in absence of other causes and its ability to do it is assumed independent of the presence of other causes.

If we consider the simplest case in which both causes and effects are binary, it is possible to define the entire CPD $P(E|\{C_1, C_2, \cdots, C_n\})$ with only n parameters $\{p_1, p_2, \cdots, p_n\}$, where p_i represents the probability E being true when cause C_i is also true and all the other causes are false. That is:

$$p_i = P(E|\overline{C_1}, \overline{C_2}, \cdots, C_i, \cdots, \overline{C_{n-1}}, \overline{C_n}). \quad (1)$$

Then the probability of E given that a subset of causes $\mathbf{C}_s$ are simultaneously true is given by the following expression:

$$P(E|\mathbf{C}_s) = 1 - \prod_{i:C_i \in \mathbf{C}_s} (1 - p_i).$$

This combining function has a well defined semantic and a solid theoretical root, as well as other combining functions, nonetheless given our purpose we have two concerns about it. First one is that it is only worthwhile when the variables have the logical interaction explained above, but we cannot expect that because this relationship will depend on the optimization problem to be solved and we want a general model for different kind of problems.

The second one is that, in spite of we can avoid computing a full CPD for all variables, causes and effects, we still need to use such a joint configurations during parameter learning, e.g., $P(C = 1|X_i = 1)$ must be estimated as $P(E = 1|C_1 = 0, \ldots, C_{i-1} = 0, C_i = 1, C_{i+1} = 0, \ldots, C_n = 0)$. Thus, bearing in mind that EDNA learns the structural part of the model with statistics with only one conditioning variable, we also aim to estimate CPDs with only one conditioning variable.

Therefore, if we have a variable X_i and its parent set $\mathbf{Pa}_i = \{Pa_{i_1}, Pa_{i_2}, \cdots, Pa_{i_k}\}$, we propose to estimate the CPD of every variable in the problem given its parents using just the set of bivariate CPDs $P_{i_j} = P(X_i|Pa_{i_j})$. Besides, instead of using the semantic of the noisy-OR combining function a more general function such as the *mean*. So we will compute the CPD for every variable with the following formula:

$$\widehat{P}(X_i|\mathbf{Pa}_i) = \frac{1}{k} \sum_{j=1}^{k} P_{i_j}. \quad (2)$$

Notice that in this way (1) the estimation will be more accurate from small training sets, and (2) we considerably reduce the storage space.

Because of the presence of cycles, new individuals are generated by Gibbs sampling instead of probabilistic logic sampling (PLS). Although PLS is the standard simulation process in the most EDAs, Gibbs sampling has also been used in other non BNs-based EDAs [26]. Anyway, as the sampling process is based on configurations of values, there is no need of recovering the whole CPDs (given all the parents) but only to compute the probability of the required configuration by using eq. 2.

4. EXPERIMENTS

This section collects all the information about experiments carried out in order to validate our proposal. First we indicate the set of test functions used, which correspond with four functions with a high number of dependencies, and three of them with a high grade of deceptiveness (all but HIFF). Next we explain the procedure and framework that we employ to perform these experiments. Finally we present the results and we analyse them.

4.1 Test Functions

We have selected these four functions: TRAP [7], MMDP [19], HIFF [29], and HTRAP [25]. Due to space restrictions and given that these functions are well known we skip to give their definition, only we show the configuration of parameter we use in these experiments.

For function TRAP we have two parameters: order k and number of building blocks b. We have chosen 6 configurations labelled as $k \times b$ with $k = 5, 6$ and $b = 5, 7, 9$.

For function MMDP the only parameter is number of building blocks and we have $b = 3, 5, 7, 9$

For function HIFF the parameter is the level l. We have selected 4 configurations $l = 4, 5, 6, 7$.

For function HTRAP we have two parameters as well: order k and level l. In our experiments we have picked 4 configurations by combining $k = 3$ and $k = 4$ with $l = 3$ and $l = 4$, labelled as $k \times l$.

4.2 Description of Experiments

We want to compare our modified EDNA model, we called it EDNA-mean, with the previously proposed EDNA, one bivariate model from the literature (same computational complexity) as COMIT, and two other multivariate models (same representation ability), EBNA and hBOA. These two models employ typically a learning process based on the *score+search* approach. The search method uses to be the hill climbing search algorithm, and the score function can be BDe [13] or BIC [27]. In our experiments we use the BIC score, however we have tested previously EBNA and hBOA with both functions and we obtain the same performance. For EBNA and hBOA we use the structure learned in the previous iteration as starting point for the local search process, in the first iteration we use an empty graph. The local operator considered are addition, deletion and reversal. This procedure has been tested previously with good results [16].

For each test function we have selected several configurations in order to check the models under different dimensions in the same problem. We have tested the influence of the population size, so we have run all the experiments with a population size of 2048, 1024 and 512 individuals. The stopping criterion is set to $population\_size * 100$ evaluations at maximum.

All the models have been tested in the same conditions. They have to learn from the half of the population with the best individuals. The old population is replaced by the best individuals taken from the old population and the last sampled individuals, truncated selection, as usual, except, of course, in the case of hBOA where is used the Restricted Tournament Replacer (RTR). For every configuration 30 runs have been made and in each single run all the models start from the same initial population in order to avoid the results get biased by that fact.

For DNs model we use a simple implementation of the Gibbs sampler and we set the number of burn-in samples to two times the number of variables in the problem and a latency of two.

The algorithms tested and the problems used have been implemented in library of optimization LiO [17]. The hardware platform is the same for all runs to make time measures reliable and consists of a server with Pentium IV 3Ghz processor with 2Gb of RAM memory.

4.3 Results

Once we have run all the experiments we can gather the results. We want to analyse the performance of our proposal in two ways. First we want to compare these models about convergence. In this topic we take into account how many runs out of 30 the model can reach the optimum. We will compare this amount between the models for the same configuration.

Second, we want to check model speed and we will compare the average time spent by each model to perform the 30 executions. The run time reported is relative to the run time of our model EDNA-mean.

4.3.1 Convergence

In Table 1 are shown the results for TRAP function. In bold face we indicate those cases in which the given model can solve at least 66% of the runs.

It can be seen easily that EDNA-mean is the best one in all configurations, no matter the size of the problem or the population size. Neither EDNA nor EBNA can get the optimum in any case.

As it expected, the smaller the population size is the worse results are obtain, but this is the same for every model.

Table 1: Number of runs in which optimum is reached for TRAP function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	5x5	5x7	5x9	6x5	6x7	6x9
COMIT	0	0	0	0	0	0
EDNA	0	0	0	0	0	0
EDNA-mean	**28**	**29**	**26**	**29**	**29**	**30**
EBNA	0	0	0	0	0	0
hBOA	**30**	**30**	**27**	15	1	0

(a)

COMIT	0	0	0	0	0	0
EDNA	0	0	0	0	0	0
EDNA-mean	**29**	**30**	**29**	**24**	**25**	**21**
EBNA	0	0	0	0	0	0
hBOA	**29**	18	8	0	0	0

(b)

COMIT	0	0	0	0	0	0
EDNA	0	0	0	0	0	0
EDNA-mean	**29**	18	14	**28**	15	1
EBNA	0	0	0	0	0	0
hBOA	**23**	7	0	0	0	0

(c)

In Table 2 we show the results for MMDP problem. In that case is hBOA which is the best model without a doubt. Nonetheless, EDNA-mean is still better that EDNA and EBNA.

Table 3 shows the results for HIFF function. Here hBOA is still the best model, EBNA is now better than EDNA-mean, however the later is always better than EDNA but in one case.

Table 2: Number of runs in which optimum is reached for MMDP function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	3	5	7	9
COMIT	2	0	0	0
EDNA	3	0	0	0
EDNA-mean	**24**	7	1	0
EBNA	5	0	0	0
hBOA	**30**	**30**	**28**	**24**

(a)

COMIT	0	0	0	0
EDNA	0	0	0	0
EDNA-mean	9	4	2	0
EBNA	0	0	0	0
hBOA	**30**	**30**	**26**	16

(b)

COMIT	1	0	0	0
EDNA	0	0	0	0
EDNA-mean	9	7	0	0
EBNA	2	0	0	0
hBOA	**30**	**28**	**21**	4

(c)

Table 3: Number of runs in which optimum is reached for HIFF function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	4	5	6	7
COMIT	**30**	**30**	**30**	7
EDNA	**30**	16	1	0
EDNA-mean	**30**	**24**	10	0
EBNA	**30**	**30**	**30**	7
hBOA	**30**	**30**	**30**	**30**

(a)

COMIT	**30**	**30**	**24**	0
EDNA	**30**	7	0	0
EDNA-mean	**30**	**22**	3	0
EBNA	**30**	**30**	**23**	0
hBOA	**30**	**30**	**30**	**29**

(b)

COMIT	**30**	**30**	4	0
EDNA	**30**	12	0	0
EDNA-mean	**30**	19	2	0
EBNA	**30**	**30**	3	0
hBOA	**30**	**30**	**29**	2

(c)

Finally, in Table 4 we can see the results for HTRAP function. This function is probably the more difficult to solve for the models taken into account. In this case again hBOA has the best performance and, like for MMDP function, EDNA-mean is the second one, ahead of EBNA and EDNA. Is important to point out that hBOA behaves better with a smaller order (i.e the size of the trap) but EDNA-mean behaves better with smaller level.

In short, we can say, according to our experiments, that EDNA-mean outperforms EDNA without a doubt in terms of convergence, what, in fact, is our purpose. EDNA-mean also is better in general than EBNA and COMIT, but in

Table 4: Number of runs in which optimum is reached for HTRAP function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	3x3	3x4	4x3	4x4
COMIT	0	0	0	0
EDNA	0	0	0	0
EDNA-mean	**29**	5	**30**	0
EBNA	**20**	0	0	0
hBOA	**30**	**30**	19	0

(a)

COMIT	0	0	0	0
EDNA	0	0	0	0
EDNA-mean	**25**	0	10	0
EBNA	3	0	0	0
hBOA	**30**	**20**	3	0

(b)

COMIT	0	0	0	0
EDNA	0	0	0	0
EDNA-mean	**23**	0	0	0
EBNA	0	0	0	0
hBOA	**30**	0	0	0

(c)

HIFF problem, what does not happend with hBOA. Nonetheless EDNA-mean is the best one for TRAP function, specially with higher problem dimension.

4.3.2 Model Speed

This section is devoted to analyse the computational cost of the models, that is we want to check how fast are them and if there is some relation with the dimension of the problem. We focus on run time in order to asset how efficient are the models we have in our comparison. We report, in the form of table, the run time for every model relative to the run time of EDNA-mean, so a figure greater than 1 means that this model is slower and a figure lower than 1 means that this model is faster than EDNA-mean.

In Table 5 we report the results for TRAP function. It can be seen that EDNA-mean is always faster than hBOA, EBNA and EDNA, but the later is a bit faster in one case. It is noticeable that EDNA-mean can be more that 40 times faster than hBOA.

In Table 6 are shown the results for MMDP function. Here we can take almost the same conclusion than with the TRAP function, EDNA-mean is faster than EBNA and hBOA. Meanwhile EDNA and EDNA-mean have a quite stable relation in their run times, EBNA and hBOA are slower and slower as the problem dimension increase so that means that our model is more scalable than the other two, however hBOA is faster in the smaller problem dimension. Again COMIT is faster than EDNA-mean.

In Table 7 we have the results for HIFF function. This table shows different information. Now EDNA-mean is slower that the other models in most of the cases. Nonetheless we can see as EBNA and hBOA are much slower than EDNA-mean with the lagerst problem dimensions.

For function HTRAP we can see its results in Table 8. Again EBNA and hBOA are slower and with worse scalability, COMIT is in average faster, and EDNA is not faster than EDNA-mean.

Table 5: Running time for TRAP function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	5x5	5x7	5x9	6x5	6x7	6x9
COMIT	1.16	1.41	0.89	1.59	1.28	1.58
EDNA	4.71	5.66	3.62	6.43	5.22	6.3
EBNA	7.65	10.65	7.67	10.62	10.12	14.07
hBOA	2.13	4.49	5.76	15.54	26.46	43.94

(a)

COMIT	1.53	1.7	1.17	0.76	0.73	0.54
EDNA	5.96	6.66	4.63	2.99	2.89	2.15
EBNA	9.29	12.39	9.68	5	5.68	4.91
hBOA	3.79	15.34	17.19	10.75	15.48	16.81

(b)

COMIT	1.07	0.47	0.34	1.05	0.38	0.25
EDNA	4.05	1.75	1.29	3.94	1.44	0.95
EBNA	6.48	3.39	2.86	6.8	3.01	2.31
hBOA	5.17	5.56	5.19	12.98	6.6	6.25

(c)

Table 6: Running time for MMDP function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	3	5	7	9
COMIT	0.68	0.34	0.33	0.35
EDNA	2.56	1.17	1.09	1.08
EBNA	9.59	5.89	6.27	6.84
hBOA	0.69	1.38	3.64	7.25

(a)

COMIT	0.38	0.34	0.34	0.34
EDNA	1.43	1.2	1.14	1.11
EBNA	4.38	4.87	5.57	6.51
hBOA	0.42	1.66	4.51	8.55

(b)

COMIT	0.35	0.32	0.32	0.33
EDNA	1.41	1.18	1.13	1.12
EBNA	3.41	4.31	5.22	6.29
hBOA	0.52	2.23	6.09	9.11

(c)

This analysis about running time can be a bit tricky because not all models yield the same results, i.e. normally some of the models can find the optimum in many of the runs but the others in a few or none of them. However we can compare running time for EDNA-mean with other models in the case where all of them can achieve a reasonable number times the optimum fitness. For instance, for HIFF function with 2048 population and in the smaller configuration we can make a more fair comparison and in this case all the models a faster than EDNA-mean, our model has not good results with this function. In the other hand we can compare EDNA-mean with hBOA in TRAP function with 2048 population in the three first configurations and here our model if faster and more scalable.

Evaluation Cost.

Apart of the running time is common the analysis of the number of evaluations, which is a very good way to assess the speed, efficiency and scalability of a model when the

Table 7: Running time for HIFF function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	4	5	6	7
COMIT	0.08	0.04	0.06	0.21
EDNA	0.36	0.95	1.03	0.96
EBNA	0.62	0.56	1.17	4.72
hBOA	0.68	0.61	1.24	3.62

(a)

COMIT	0.08	0.05	0.11	0.26
EDNA	0.33	1.32	1	0.97
EBNA	0.59	0.64	1.9	6.62
hBOA	0.58	0.62	1.51	6.6

(b)

COMIT	0.08	0.05	0.24	0.26
EDNA	0.34	0.91	0.98	0.98
EBNA	0.66	0.74	3.58	8.78
hBOA	0.63	0.69	2.49	14.03

(c)

Table 8: Running time for HTRAP function with a population of 2048 (a), 1024 (b) and 512 (c) individuals.

	3x3	3x4	4x3	4x4
COMIT	1.35	0.28	1.31	0.24
EDNA	5.34	1.08	5.18	0.98
EBNA	4.77	3.69	14.11	0
hBOA	2.09	2.72	20.83	17.88

(a)

COMIT	0.74	0.25	0.32	0.23
EDNA	2.87	0.95	1.25	0.98
EBNA	4.93	3.45	3.42	9.8
hBOA	1.35	4.67	6.71	20.96

(b)

COMIT	0.55	0.26	0.25	0.24
EDNA	2.05	0.96	0.94	1.01
EBNA	3.77	4.06	2.79	23.85
hBOA	1.42	6.6	5.11	37.19

(c)

cost of every evaluation is constant or at least is the same for all the models. However, in this comparison is evident that this is not true, the cost per evaluation dependends on the complexity of the model and the problem dimension. Normally when the problem dimension increases there is a higher number of dependencies that the model have to manage and thus a multivariate model should need more time per evaluation, although if it can deal properly with these dependencies should need less evaluations than a simpler model in total. That is the reason why we report next the time that each model spend for one evaluation. As this rate is almost constant among the different population sizes we only show the figures for 2048 individuals experiments.

In Table 9 are shown the results for TRAP problem.COMIT has in all cases the least computational cost per evaluation, what is expected, meanwhile EDNA and EDNA-mean have a similar cost. Also is noticeable that hBOA has the biggest increase with the problem dimension, so it exhibits the worst scalability apart that it has the greater cost always.

Table 9: Time per evaluation rate for TRAP function.

	5x5	5x7	5x9	6x5	5x7	5x9
COMIT	0.01	0.01	0.02	0.01	0.02	0.03
EDNA	0.03	0.05	0.08	0.04	0.07	0.11
EDNA-mean	0.04	0.06	0.09	0.05	0.08	0.11
EBNA	0.05	0.10	0.17	0.07	0.14	0.24
hBOA	0.25	0.47	0.60	0.18	0.38	0.74

With function MMDP, as we can see in Table 10, COMIT is again cheaper per evaluation and EDNA-mean is a bit better than EDNA. hBOA is the worst and with bad scalability like before but EBNA shows worse scalability according with problem dimension.

Table 10: Time per evaluation rate for MMDP function.

	3	5	7	9
COMIT	0.01	0.02	0.03	0.04
EDNA	0.03	0.06	0.09	0.13
EDNA-mean	0.02	0.05	0.08	0.12
EBNA	0.11	0.28	0.51	0.82
hBOA	0.09	0.32	0.69	1.17

From Table 11 we can take the same conclusions from HIFF problem like with MMDP.

Table 11: Time per evaluation rate for HIFF function.

	4	5	6	7
COMIT	0.01	0.02	0.06	0.14
EDNA	0.03	0.05	0.15	0.55
EDNA-mean	0.02	0.05	0.15	0.57
EBNA	0.05	0.23	1.13	3.22
hBOA	0.05	0.25	1.21	7.36

Finally, in Table 12 we have the results for HTRAP function, in which models show the same behabiour. In the last two tables we can see that the problem dimension affects more the cost per evaluation than with the first two problems.

Table 12: Time per evaluation rate for HTRAP function.

	3x3	3x4	4x3	4x4
COMIT	0.01	0.06	0.04	0.55
EDNA	0.04	0.21	0.14	2.30
EDNA-mean	0.04	0.22	0.15	2.34
EBNA	0.09	0.73	0.38	14.48
hBOA	0.20	2.14	0.99	41.86

Time Distribution.

In this last point we want to analyse how the models distribute the running time between structural and parametrical learning and sampling. In Table 13 we show the percentage of the total time devoted to each one of these tasks. We

show the range of these values, maximum and minimum percentage, for all the executions. As is expected, DNs models spend most of the time in the sampling stage. To this task these two models dedicate approximately 4 times more time than to structural learning, however the rest of the models spend much less time sampling new individuals. If we compare only the distribution between EDNA and EDNA-mean we can realize that the parametrical learning in EDNA-mean is less expensive, so learning bivariate probability distributions is better computationally.

Table 13: Percentage of runtime divided into structural and parametrical learning and sampling.

	structural		parametrical		sampling	
	max	min	max	min	max	min
COMIT	92.34	42.88	12	0.83	52.1	6.83
EDNA	18.99	7.87	8.4	1.23	84.1	77.77
EDNA-mean	22.47	10.36	1.8	0.22	88.45	77.12
EBNA	98.56	70.48	3.83	0.06	25.69	1.38
hBOA	98.95	78.27	3.6	0.06	18.13	0.99

5. CONCLUSIONS

In this paper we have presented an approach whose aim is to improve a model for EDAs based on DNs. This model tries to learn a multivariate probabilistic representation from the population but using only bivariate statistics in a very simple way thanks to the facilities provided by DNs.

This approach is based on the concept of *combining function* in the way that we want to be able to obtain the conditional probability distribution for every variable in the model by only computing and storing a CPD for the given variable given each one of its parent, i.e. a CPD with only one conditioning variable.

The reason of doing that is two fold. First, we think that estimating simpler CPD and combining them can help us to overcome the potential overfitting problem of the full CPD with all parents, and the second reason is to spread the idea of EDNA, learning the structural part by using only 2-order statistics, to the parametrical part of the learning.

Besides, we want to compare our proposal with other multivariate models from the literature, which have the same representation ability, and a bivariate model which has the same computational complexity. We have chosen EBNA and hBOA as multivariate models, because of their generality and good results, and COMIT as the representative for bivariate models. We have selected too a set of test functions with different characteristics but all of them with a great complexity in order to get a good assessment about the goodness of the models.

From the results presented we can clearly conclude that our proposal improve EDNA in all the problems tested about the ability to solve them and it does not have more computational cost. Concerning EBNA and hBOA, our model is faster than them in most of the cases tested and behaves better when the problem dimension increases. About COMIT we have seen that is always faster and shows better results than EDNA-mean for HIFF problem

EDNA-mean is the best one for the TRAP function without a doubt, in which only hBOA shows a performance worthy to be compared to EDNA-mean. For the other functions hBOA is better than EDNA-mean, in some case with a great difference, however EBNA only win our model in HIFF problem but it loses in the rest of them. Taking into account that HIFF problem is the only non deceptive problem we can say that EDNA-mean model shows very good results for deceptive functions, is better than EBNA and COMIT and can be considered as good as hBOA in average, although is much better in TRAP function.

From the distribution of time that each model dedicates to each task we see that most of the time that EDNA and EDNA-mean spend is for sampling, so improving the sampling algorithm, in terms of computational complexity, for these models can get us a global improvement.

We have to take into account that in our version of EDNA we are using restricted information, bivariate estimation, against a much more complex estimation as the EDAs based on BNs are using. As final remark we think that we have achieved successfully our initial objective of presenting an improvement of the EDNA model.

As future works, we plan to test our proposal in a more systematic way, and to improve the EDNA with other combining functions and also using other kind of replacer as the used by hBOA. Also, we plant to scale up the algorithm by reusing the later model learned in the previous stages as hBOA and EBNA do. Also, as it has been pointed, can be interesting to try other way to make the sampling, with other parameters for Gibbs sampling or even try other sampling algorithms.

6. ACKNOWLEDGMENTS

This work has been partially supported by Spanish Ministerio de Educación y Ciencia (project TIN2007-67418-C03-01); Junta de Comunidades de Castilla-La Mancha (projects PBI-05-022 and PBI-08-048) and FEDER funds.

7. REFERENCES

[1] S. Baluja and S. Davies. Combining Multiple Optimization Runs with Optimal Dependency Trees. Technical Report CMU-CS-97-157, Carnegie Mellon University, 1997.

[2] J. S. D. Bonet, C. L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, volume 9, pages 424–430. MIT Press, Cambridge, 1997.

[3] N. Friedman and M. Goldszmidt. Learning bayesian networks with local structure. In *12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 252–262, 1996.

[4] J. A. Gámez, J. L. Mateo, and J. M. Puerta. Dependency networks based classifiers: learning models by using independence test. In *Third European Workshop on Probabilistica Graphical Models (PGM06)*, pages 115–122, 2006.

[5] J. A. Gámez, J. L. Mateo, and J. M. Puerta. EDNA: Estimation of Dependency Networks Algorithm. In *International Work-Conference on the Interplay between Natural and Artificial Computation (IWINAC'07)*, volume 1, pages 427–436, 2007.

[6] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:147–156, 1984.

[7] D. E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. In *Parallel Problem Solving from Nature (PPSN)*, volume 2, 1992.

[8] G. Harik. Linkage learning in via probabilistic modelling in the EcGA. Technical Report 99010, Illinois Genetic Algorithms Laboratory, 1999.

[9] G. Harik. Finding multimodal solutions using restricted tournament selection. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, 2005.

[10] G. Harik, F. G. Lobo, and D. E. Golberg. The compat genetic algorithm. In *Proceedings of the 4th International Conference on Evolutionary Computation*, pages 1–5, 1997.

[11] D. Heckerman. A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft Research, Mar. 1995.

[12] D. Heckerman, D. M. Chickering, and C. Meek. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[13] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *10th Conf. Uncertainty in Artificial Intelligence*, pages 293–301. Morgan Kaufmann Publishers, 1994.

[14] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[15] M. Jaeger. Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32:179 – 220, 2001.

[16] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Combinatonal Optimization by Learning and Simulation of Bayesian Networks. In *UAI '00: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, pages 343–352, 2000.

[17] J. L. Mateo and L. de la Ossa. LiO: an easy and flexible library of metaheuristics. Technical Report DIAB-06-04-1, Departamento de Sistemas InformÃąticos, Escuela PolitÃl'cnica Superior de Albacete, Universidad de Castilla-La Mancha, 2006.

[18] H. Mühlenbein. The equation for response to selection an its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.

[19] H. Mühlenbein, T. Mahnig, and A. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215–247, 1999.

[20] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary patameters. In *4th Internatial Conference on Parallel Problem Solving from Nature*, pages 178–187, 1996.

[21] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

[22] A. Onisko, M. J. Druzdzel, and H. Wasyluk. Learning bayesian network parameters from small data sets: Application of noisy-or gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001.

[23] J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[24] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[25] M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 511–518, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

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

[27] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[28] Y. Tian, Q. Yang, T. Huang, C. X. Ling, and W. Gao. Learning Contextual Dependency Network Models for Link-Based Classification. *IEEE Transactions on Knowledge and Data Engineering*, 18:1482–1496, Nov 2006.

[29] R. A. Watson, G. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *Proceedings of the 5th International Conference on Parallel Problem Solving From Nature*, pages 480–490, 1998.