Model-based template-recombination in Markov network estimation of distribution algorithms for problems with discrete representation

Roberto Santana
Intelligent Systems Group
University of the Basque Country
San Sebastian, Spain.
Email: roberto.santana@ehu.es

Alexander Mendiburu
Intelligent Systems Group
University of the Basque Country
San Sebastian, Spain.
Email: alexander.mendiburu@ehu.es

ja.lozano@ehu.es
Intelligent Systems Group
University of the Basque Country
San Sebastian, Spain.
Email: alexander.mendiburu@ehu.es

***Abstract*—While estimation of distribution algorithms (EDAs) based on Markov networks usually incorporate efficient methods to learn undirected probabilistic graphical models (PGMs) from data, the methods they use for sampling the PGMs are computationally costly. In addition, methods for generating solutions in Markov network based EDAs frequently discard information contained in the model to gain in efficiency. In this paper we propose a new method for generating solutions that uses the Markov network structure as a template for crossover. The new algorithm is evaluated on discrete deceptive functions of various degrees of difficulty and Ising instances.**

***Keywords*-estimation of distribution algorithms; Markov networks; recombination; combinatorial problems; probabilistic modeling**

I. Introduction

Estimation of distribution algorithms (EDAs) [4], [9] are evolutionary algorithms (EAs) that apply probabilistic modeling of the selected solutions and use the models to generate new solutions. They are able to capture, explicitly represent, and exploit the interactions between the problem variables. EDAs can be classified according to the type of probabilistic models they use [4]. One particular class of EDAs uses Markov networks and other undirected graphical models [12], [18], [19]. It has been argued [11] that although Markov networks may sometimes cover the same distribution that a Bayesian network using fewer edges in the dependency model, their sampling becomes more complicated than the sampling of Bayesian networks. This is indeed the case for the vast majority of Markov-network based EDAs that use Gibbs sampling (GS) or similar Markov chain Monte Carlo (MCMC) sampling strategies. These sampling methods can transfer the information contained in the Markov models to the generated solutions but are computationally costly.

Many researchers have investigated the question of linkage learning in EDAs (see [1] for a survey). EDAs can be included in the class of EAs that apply virtual linkage, i.e., algorithms that use graph, groupings, matrices, pointers, or other data structures that control the subsequent pairing or clustering organization of decision variables [1]. By means of the strategies used for generating new solutions, EDAs explicitly use the models to create new solutions.

Some authors have proposed the combination of different methods of generating new solutions in EDAs, and the use of information about the dependencies of the variables for the implementation of crossover operators. Zhou et al [23] combine model-based and genetics-based offspring generation for multi-objective optimization. Hauschild et al [3] propose a network-crossover operator that uses information about the interactions of the variables. However, the information about the interactions is known a priori and not extracted from the data.

In this paper, we propose an efficient algorithm for generating solutions in Markov-based EDAs that uses the Markov network structure as a template for crossover. The model-based template crossover introduced in this paper also uses the probabilistic model to store information about the dependencies between the variables. However, instead of applying statistical techniques to sample from it, the crossover operator itself uses the structure in a way similar to the way multiple point recombination is applied in GAs. The benefit of this approach is that disruption can be minimized and the pace of recombination can be slowed.

To validate the introduced crossover method we compare it with different variants of methods for generating solutions in Markov network EDAs [16]. Some of these variants incorporate message passing algorithms for improving the search [6], [7], [16]. Extensive experiments on discrete fitness functions with different levels of difficulty have been conducted. Our results show that the introduced procedure can be an effective and efficient alternative to the single use of PLS and GS for Markov based EDAs.

The paper is organized as follows. In the next section, we review the Markov network factorized distribution algorithm with G-tests (MN-FDA^G) and the Markovianity based optimization algorithm (MOA) [20], [22], briefly discussing their learning and sampling steps. Details on the algorithms' implementation, the optimization functions and the numerical results of our experiments are given in Section IV. The conclusions of our paper and lines for future work are presented in Section V.

978-1-4799-3230-6/13/$31.00 ©2013 IEEE

II. Markov networks as probabilistic models in EDAs

While research on discrete EDAs have focused on algorithms that use Bayesian networks [10], other EDAs use different variants of undirected graphical models to represent the probability distributions of the solutions [21]. Among these algorithms are the Markov network factorized distribution algorithm (MN-FDA) [12] and the Markovianity based optimization algorithm (MOA) [20], [22].

Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote a vector of discrete random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $\{1, \ldots, n\}$, and X_S (respectively x_S) a subset of the variables of $\mathbf{X}$ (respectively $\mathbf{x}$) determined by the indices in S. We will work with positive distributions denoted by p. $p(x_S)$ will denote the marginal probability for $\mathbf{X}_S = \mathbf{x}_S$. We use $p(x_i \mid x_j)$ to denote the conditional probability distribution of $X_i = x_i$ given $X_j = x_j$.

MN-FDA was introduced as an extension to EDAs based on decomposable models like the Factorized Distribution Algorithm (FDA) [8] and it can capture some of the dependencies that acyclic models like junction trees are not able to represent [12]. However, since MN-FDA models dependencies as a junction graph (JG) of constrained tree-width, its capacity to represent dependencies is limited. Nevertheless, using a JG representation allows MN-FDA to sample new solutions by means of probabilistic logic sampling (PLS), which is a very efficient sampling procedure. The general steps of MN-FDA are shown in Algorithm 1.

Algorithm 1: **MN-FDA**

1 Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly.
2 **do** {
3 Evaluate the points using the fitness function.
4 Select a set D_t^S of $k \leq N$ points according to a selection method.
5 Learn an undirected graphical model from D_t^S.
6 Generate the new population sampling from the model.
7 $t \Leftarrow t + 1$
8 } **until** Termination criteria are met

To learn the junction graph, MN-FDA applies first statistical tests to determine the variables that are pairwise dependent. We use the G-test of independence [5] to detect dependencies. To compute the test, the relationship between the G-test and the mutual information is used, $G(X_i, X_j) = 2NMI(X_i, X_j)$, where MI is the mutual information and N is the number of samples. The MN-FDA that uses this test is called MN-FDA^G [16]. After the undirected model $\mathcal{G}$ has been learned, it is refined to make it sparser. Then, the set L of all the maximal cliques of $\mathcal{G}$ are found. A labeled JG is learned from L and finally the marginal probabilities for the cliques in the JG are computed from the data. Details on the algorithms used for refining the graph, finding the maximal cliques, and constructing the labeled JG are given in [13]

MOA [20], [22] is a more powerful algorithm in terms of the number of probabilistic dependencies that it is able to capture. It was conceived combining features from the MN-FDA and the Distribution Estimation Using Markov network (DEUM) [22]. MOA is based on a more general undirected model than MN-FDA, although some constraints can be enforced on the number of neighbors that each node can have. It also employs more efficient procedures for learning the structure of the model. Nevertheless, it requires to use GS to sample this model. MOA's workflow is described in Algorithm 2.

Algorithm 2: **MOA**

1 Set $t \Leftarrow 0$. Generate M points randomly
2 **do** {
3 Evaluate the points using the fitness function.
4 Select a set D_t^S of $N \leq M$ points according to a selection method.
5 Estimate the structure of a Markov network from D_t^S.
6 Estimate the local Markov conditional probabilities, $p(x_i|N_i)$, for each variable X_i as defined by the undirected structure.
7 Generate M new points sampling from the Markov network.
8 $t \Leftarrow t + 1$
9 } **until** Termination criteria are met

Model learning in MOA is achieved by firstly computing the mutual information between pairs of variables and creating later edges between pairs for which the mutual information is higher than a given threshold TR. Let $avg(MI)$ be the average of the elements of the mutual information matrix, and sig the significance parameter, the threshold is computed as $TR = avg(MI) * sig$. Following [22], we set $sig = 1.5$. The number of neighbors is also limited to the N_{neigh} neighbors that have the highest mutual information.

To generate new solutions, MOA applies r steps of GS, where $r = n \times ln(n) \times IT$, and IT, the *iteration coefficient*, is set to 4. For binary variables, the probability of $x_i = 1$ given the value of its neighbors N_i is $p(x_i = 1|N_i) = \frac{e^{p(x_i=1,N_i)/T}}{e^{p(x_i=1,N_i)/T} + e^{p(x_i=\bullet,N_i)/T}}$, where T is the *temperature coefficient*. T serves to balance the exploration and exploitation capabilities of MOA. More details on the learning and sampling procedures used by MOA can be obtained from [22].

III. Using Markov networks as template for crossover

One straightforward way to use the Markov network structure for generating solutions is by means of a crossover

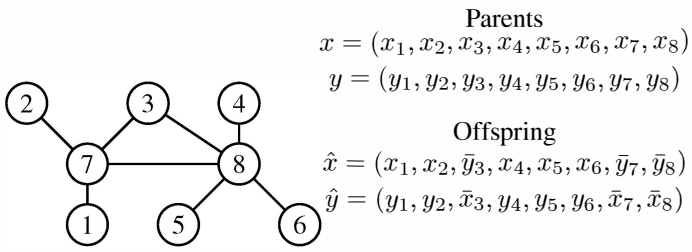

Figure 1. Markov network representing the dependencies between 8 variables and a possible crossover between two solutions. Crossover is based on the largest clique of the Markov network, comprising variables X_3, X_7, X_8.

operator that respects the problem dependencies captured within the cliques. Respecting the dependencies is important because very often they correspond to interactions between the variables of the problem.

Algorithm 3: **Markov-network template crossover**

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

We use the Markov network as a template for exchanging partial configurations between solutions. Partial configurations corresponding to cliques of the Markov network are exchanged between pairs of solutions. The pseudocode for this approach is shown in Algorithm 3. An example of the application of the Markov-network template crossover is shown in Figure 1. In this example, crossover is based on only one of the cliques of the Markov network.

Previous results have shown that cliques can capture relevant, and usually interpretable [14], partial information about the problem. By exchanging these subsolutions we combine the global information contained in the probabilistic model with local information contained in the solutions. The distinguished feature of this crossover method is that the subsolutions to be exchanged will correspond to maximal cliques of the graph.

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

The choice of MN-FDA^G or MOA influences the type of undirected model that is learned from data but also the strategy used for generating new solutions. The graphical model used by MN-FDA is simpler, usually with less and smaller cliques than MOA.

IV. Experiments

In this section we compare the Markov-network template crossover to different variants of methods for generating solutions in Markov network EDAs. First, we provide the implementation details and introduce the functions and experimental benchmarks. Then the numerical results are presented and discussed.

A. Function and experimental benchmark

To evaluate our algorithms we use two classes of functions. Separable additively decomposable functions (ADFs) defined on discrete non-binary variables and non-separable decomposable binary functions. Separable ADFs can be solved by EDAs based on marginal product factorizations [2], [15]. However, the difficulty of these functions increases with the cardinality of the variables. Non-separable functions are generally hard to solve due to the arousal of many dependencies between the variables.

1) Separable ADFs defined on discrete non-binary variables: We use a deceptive function defined on non-binary variables.

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

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

The general deceptive function $f^c_{deceptivek}(\mathbf{x})$ of order k [17], [15] is formed as an additive function composed by the function $F_{dec}(x_1, \ldots, x_k, k, c)$ evaluated on substrings of size k and cardinality c, i.e. $x_i \in \{0, \ldots, c-1\}$. This function is a generalization of the binary $f_{deceptivek}(\mathbf{x})$ to variables with non-binary values.

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

$$H = -\sum_{i<j\in L} J_{ij}\sigma_i\sigma_j - \sum_{i\in L} h_i\sigma_i \qquad (1)$$

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

B. Methods for generating solutions

We evaluate the Markov-network template crossover using MN-FDA^G and MOA. We compare to other three strategies used for generating solutions [16]. The four strategies were:

s1. Insert-MAP: Generate the new population using the EDA's original sampling method and inserting the maximum a posteriori (MAP) solution computed from the Markov network.
s2. Template-MAP: Generate solutions using the EDA's original sampling and combine them with the MAP solution.
s3. Insert-MAP + Template-MAP: Generate new population using Template-MAP and add the MAP solution.
s4. Insert-MAP + Markov-network template crossover: Generate new population using Markov-network template crossover and add the MAP.

In previous work [6], [7], [16], we have described how the insertion of the MAP solution improves the results of the search for many functions. The MAP solution can be found using different methods. Therefore, we also evaluate the influence that using different inference methods has in the behavior of the EDA when combined with Markov-network template crossover. Four inference methods are compared:

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

1) Experimental benchmark: We compare the algorithms in terms of their critical population size to reach the optimum and the average number of evaluations needed. The critical population size is computed as the minimum population size needed to reach the optimum in l successive experiments and the average number of evaluations is computed from a maximum of r independent computations of the critical population size (less than r if it was not possible to find a critical population size in all the runs).

For function $f^c_{deceptivek}(\mathbf{x})$, we set $l = 30$ and $r = 30$. For the Ising model, $l = 5$ and $r = 20$. The population size is started at $N = 32$ and doubled if the algorithm does not converge in the l experiments until a maximum of $N = 131072$. The maximum number of generations was $g = 40$ in all the experiments. Truncation selection, $T = 0.5$, is applied together with best elitism in which the complete selected population is passed to the next generation.

C. Numerical results

1) Deceptive discrete functions: We investigate the performance of MN-FDA^G and MOA using function $f^c_{deceptivek}(\mathbf{x})$, $k \in \{3, 4, 5\}$, $c \in \{2, 3, 4, 5\}$. We are interested in evaluating the behavior of the EDAs when the size of the definition sets and the number of values of each variable are increased.

Table I shows the number of successful runs of different EDAs variants when the cardinality of the variables is increased from $c = 2$ to $c = 5$. Each cell of the table groups the number of successful runs for 3 different cardinality values $k \in \{3, 4, 5\}$. Therefore the maximum number of successful runs is 90.

It can be seen in Table I that for MN-FDA^G the Markov-network template crossover ($s4$) does not outperform the results achieved by the original sampling method enhanced with Insert-MAP ($s1$). This is particularly evident when $c = 3$ and $c = 4$, for which the successful rate of the other strategies (except $s2$) is higher. However, for MOA, $s4$ is equal or better than all other algorithms. Improvements are also achieved in the number of function evaluations. This fact can be appreciated in Figure 2 that shows the results of the different strategies for generating new solutions for $f^c_{deceptivek}(\mathbf{x})$, $n = 32$, $k = 3$, $c = 5$. The Markov-network template crossover ($s4$) needs the lowest number of function evaluations among strategies used for MOA and it is only outperformed by MN-FDA^G when strategy $s1$ is used.

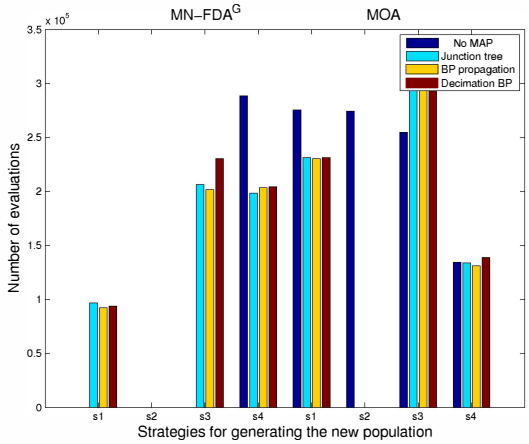

Figure 2. Comparison of Markov-network template crossover with other strategies for generating new solutions and using different variants of inference methods for $f^c_{deceptivek}(\mathbf{x})$, $n = 32$, $k = 3$, $c = 5$.

2) Ising instances: The Ising problem allows to evaluate the EDAs in a problem with multiple, loopy dependencies, where the definition functions are different among them. Each Ising instance corresponds in fact to a different fitness function. The results of the algorithms are summarized in Table II where we show the number of times the algorithms were able to solve the problem in 5 consecutive runs. Each cell of the table adds the number of successful runs for the 20 experiments conducted for each of the 4 instances. Solving the problems in every run for every instance means that the cell entry is $4 \times 20 = 80$.

An analysis of Table II reveals that, for MN-FDA^G, when the BP and Dec-BP inference procedures are used, the Markov-network template crossover achieves results similar to $s1$ for $n = \{36, 64\}$. However, when the dimension of the problem increases ($n = 100$), its results remarkably deteriorates. Also for MOA, when the BP and Dec-BP inference

Table I
COMPARISON OF MARKOV-NETWORK TEMPLATE CROSSOVER WITH OTHER STRATEGIES FOR GENERATING NEW SOLUTIONS AND USING DIFFERENT VARIANTS OF INFERENCE METHODS FOR $f^c_{deceptivek}(\mathbf{x}), n = 30,\ k \in \{3,4,5\}$. NUMBER OF RUNS THAT ACHIEVED THE OPTIMUM.

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

Table II
COMPARISON OF MARKOV-NETWORK TEMPLATE CROSSOVER WITH OTHER STRATEGIES FOR GENERATING NEW SOLUTIONS AND USING DIFFERENT VARIANTS OF INFERENCE METHODS FOR THE ISING PROBLEM. NUMBER OF RUNS THAT ACHIEVED THE OPTIMUM.

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

procedures are used, the Markov-network template crossover achieves results similar to $s1$ for $n = \{36, 64\}$. However, for $n = 100$, the Markov-network template crossover is the only strategy for generating new solutions that finds the optimum. In general, the results of the experiments indicate that the benefit of applying the introduced model-based crossover extends only to MOA. One possible reason is that the number of cliques and their sizes are higher for MOA than for MN-FDA^G and under these conditions one speculate that the crossover tends to better recombine the information of the solutions.

V. CONCLUSIONS

In this paper we have introduced a new method for generating solutions in Markov network based EDAs. The methods intends to combine the structural information contained in the Markov networks with the local exchange of information that the application of a crossover operator allows. We have evaluated different variants of the new algorithm that incorporate the MAP solution learned using different inference methods. As a function benchmark, we have used hard deceptive functions defined on binary and non-binary discrete representation. Ising instances of up to 100 variables have been also included in the benchmark.

Our preliminary results show that the Markov-network template crossover can improve the efficiency of Markov network based EDAs, such as MOA, able to represent very intricate dependencies between the variables. However, the model-based crossover does not improve the results of PLS, the traditional strategy for generating solutions, in MN-FDA^G. For the experiments included in this paper, we did not carefully tune the parameters of the crossover operator. Further research is required to investigate the dynamics of the Markov-network template crossover and propose a way to set its parameters, in particular the crossover rate and the amount of information to be exchanged between the parents.

ACKNOWLEDGEMENTS

This work has been partially supported by Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2010-14931, and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

REFERENCES

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

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

[3] M. W. Hauschild and M. Pelikan. Network crossover performance on NK landscapes and deceptive problems. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation (GECCO-2010)*, pages 713–720. ACM, 2010.

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

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

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

[7] A. Mendiburu, R. Santana, and J. A. Lozano. Fast fitness improvements in estimation of distribution algorithms using belief propagation. In R. Santana and S. Shakya, editors, *Markov Networks in Evolutionary Computation*, pages 141–155. Springer, 2012.

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

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

[10] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*, volume 170 of *Studies in Fuzziness and Soft Computing*. Springer, 2005.

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

[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.

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

[14] R. Santana, R. Armañanzas, C. Bielza, and P. Larrañaga. Network measures for information extraction in evolutionary algorithms. *International Journal of Computational Intelligence Systems*, 6(6):1163–1188, 2013.

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

[16] R. Santana, A. Mendiburu, and J. A. Lozano. Message passing methods for estimation of distribution algorithms based on Markov networks. In *Proceedings of the 4th Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO-2013)*, Lectures Notes in Computer Science, pages 419–430, Chennai, India, 2013. Springer. In press.

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

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

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

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

[21] S. Shakya and R. Santana. A review of estimation of distribution algorithms and Markov networks. In S. Shakya and R. Santana, editors, *Markov Networks in Evolutionary Computation*, pages 21–37. Springer, 2012.

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

[23] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 892–899. IEEE, 2006.