Message Passing Methods for Estimation of Distribution Algorithms Based on Markov Networks

Roberto Santana, Alexander Mendiburu, and Jose A. Lozano

Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country (UPV/EHU)
Paseo Manuel de Lardizabal 1, 20080, San Sebastian, Guipuzcoa, Spain
{roberto.santana,alexander.mendiburu,ja.lozano}@ehu.es

Abstract. Sampling methods are a fundamental component of estimation of distribution algorithms (EDAs). In this paper we propose new methods for generating solutions in EDAs based on Markov networks. These methods are based on the combination of message passing algorithms with decimation techniques for computing the maximum a posteriori solution of a probabilistic graphical model. The performance of the EDAs on a family of non-binary deceptive functions shows that the introduced approach improves results achieved with the sampling methods traditionally used by EDAs based on Markov networks.

Keywords: estimation of distribution algorithms, message passing, Markov networks, probabilistic modeling, belief propagation, abductive inference.

1 Introduction

Genetic Algorithms that apply simple genetic operators are usually unable to identify the problem variables that interact and therefore they are not very efficient at the time of generating solutions. Extensive research has been devoted to design heuristic recombination operators based on a priori information about the problem domain and adaptive genetic operators that attempt to identify and preserve the linkage between the variables. However, these are not general and robust solutions to these problems.

Estimation of distribution algorithms (EDAs) [4,11] are evolutionary algorithms (EAs) that deal in a different way with the "building-block identification" and "building-block mixing or exchange" questions [16]. EDAs use machine learning techniques to learn a probabilistic model from the selected solutions and to sample new solutions from this model. There is overwhelming evidence that probabilistic modeling of selected solutions is a very competitive approach to the "linkage problem" as known in EAs [3,5,20].

The choice of the probabilistic model in EDAs influences the type of problems that can be successfully addressed and the machine learning algorithms used to

B.K. Panigrahi et al. (Eds.): SEMCCO 2013, Part II, LNCS 8298, pp. 419–430, 2013.

learn the models and to sample from them. One class of EDAs able to represent higher order interactions between the variables are EDAs that use Markov networks [13,18,20]. Sampling methods commonly applied by this type of EDAs can transfer the information contained in the Markov models to the generated solutions but are computationally costly [21]. In this paper we propose the combination of message passing algorithms [12] with decimation strategies [1] for generating solutions in EDAs based on Markov networks.

The paper is organized as follows. In the next section, we review two typical examples of EDAs based on Markov networks. Section 3 presents factor graphs, message passing algorithms, and decimation methods for computing the (MAP) of a factor graph. Section 4 introduces three variants of algorithms for generating solutions in EDAs. The experimental results on the validation of our proposal are given in Section 5. The conclusions of our paper are presented in Section 6.

2 Markov Network Factorized Distribution Algorithm and the Markovianity Based Optimization Algorithm

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

The Markov network factorized distribution algorithm (MN-FDA) [13] and the Markovianity based optimization algorithm (MOA) [19,21] are two different approaches to the use of undirected graphical models to represent probabilistic dependencies in EDAs. MN-FDA can capture some of the dependencies that acyclic models like trees are not able to represent, however the number of these dependencies that it is able to represent is rather limited. The reason for this limitation is that MN-FDA represents dependencies using a junction graph of bounded complexity [15]. A good characteristic of MN-FDA is that it samples new solutions using probabilistic logic sampling (PLS), which is a very efficient sampling procedure. MOA is a more powerful algorithm in terms of the number of probabilistic dependencies that it is able to capture. Nevertheless, it requires to use the Gibbs sampling (GS) algorithm to sample this model. GS is a computationally costly algorithm.

2.1 MN-FDA

MN-FDA defines a factorization in which the factors correspond to a set of maximal cliques organized as a labeled junction graph (JG). The cliques in the factorization can be ordered in such a way that at least one of the variables in every clique is not contained in the previous nodes in the ordering. MN-FDA

allows the existence of cycles between the factors, therefore expanding the class of distributions represented by junction trees [13].

Algorithm 1. MN-FDA

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

The general steps of MN-FDA are shown in Algorithm 1. The steps that describe the way the probabilistic model is learned are shown in Algorithm 2. In this paper, we use the G-test of independence [6] to detect dependencies. G-tests are a subclass of likelihood ratio tests, a general category of tests that have many uses for testing the fit of data to mathematical models. To compute the test, we use the relationship between the G-test and the mutual information:

$$G(X_i, X_j) = 2NMI(X_i, X_j) \tag{1}$$

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

Since the use of the G-statistics implies a different way to determine the relationships between variables and different characteristics of the learning algorithm, we call the MN-FDA that uses this test as MN-FDA^G.

Algorithm 2. Model learning in MN-FDA

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

Details on the algorithms used for refining the graph, finding the maximal cliques, and constructing the labeled JG are given in [14]

2.2 MOA

MOA [19,21] exploits the Markovianity or local Markov property of Markov networks. Its workflow is described in Algorithm 3.

Algorithm 3. Markovianity based optimization algorithm

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

To learn the Markov network structure, first the mutual information for each pair of variables in computed. Then, an edge between two variables is created if the mutual information between them is higher than a given threshold. Here we compute the threshold, TR, as $TR = avg(MI) * sig$, where $avg(MI)$ is the average of the elements of the mutual information matrix and sig is the significance parameter. Following [21], we set $sig = 1.5$. If the number of neighbors of a variable is higher than an allowed maximum number, N_{neigh}, then only the N_{neigh} neighbors that have the highest mutual information are kept.

MOA uses a Gibbs sampler with r steps, where $r = n \times ln(n) \times IT$, and IT, the *iteration coefficient*, is set to 4. In the particular case of binary representation, the probability of $x_i = 1$ given the value of its neighbors N_i is $p(x_i = 1|N_i) = \frac{e^{p(x_i=1,N_i)/T}}{e^{p(x_i=1,N_i)/T}+e^{p(x_i=0,N_i)/T}}$, where T is the *temperature coefficient* that controls the convergence of the Gibbs probability distribution. More details on the learning and sampling procedures used by MOA can be obtained from [21].

3 Message Passing and Decimation Methods Methods on Factor Graphs

A *factor graph* [2] is a bipartite graph that can serve to represent the factorized structure of a distribution. It has two types of nodes: variable nodes, and factor nodes. Variable nodes are indexed with letters starting with i, and factor nodes with letters starting with a. The existence of an edge connecting variable node i to factor node a means that x_i is an argument of function f_a in the referred factorization.

Belief propagation (BP) [12] is an inference method used to calculate the marginal probabilities of a probabilistic model. It is usually applied after some evidence about the observed states of the variables has been incorporated to the model. A key characteristic of the BP algorithm is that the inference is done using message-passing between nodes. Each node sends and receives messages until a stable situation is reached. Messages, locally calculated by each node, comprise statistical information concerning neighbor nodes.

When the algorithm converges (i.e. messages do not change), marginal functions (sum-product) or max-marginals (max-product) are obtained as the normalized product of all messages received by X_i. For the the max-product approach, each variable in the optimal solution is assigned the value given by the configuration with the highest probability at each max-marginal.

Decimation strategies [1] are used to solve constraint satisfaction problems. They work by repeatedly fixing variable values and simplifying the constraints without reconsidering earlier decisions. They can be seen as a divide and conquer approach in which the problem is split and simplified at each step.

In the context of EDAs, we will apply a decimation strategy to find an approximate solution to the maximum a posteriori probability (MAP) configuration of a factor graph representing a probability distribution. At each step of the simple BP-decimation algorithm the idea is to identify the variable X_i that is almost certainty to have a given value in the problem's most probable configuration. First, max-BP is run. Then, the variable X_i with the largest likelihood on one of its configurations is identified and fixed to its most probable value. In the next step, the variable node X_i is removed from the factor graph, and all factors that contain X_i are reduced by eliminating the tuples which force X_i to take a value different from the one fixed. The cycle is repeated until all variables are fixed. The BP-decimation scheme used in this paper is based on the *decmap* program included in the libDAI implementation [9].

4 New Methods for Generating Solutions in EDAs Based on Markov Networks

We propose two different ways for using the information contained in the Markov networks for generating new solutions: 1) Adding the most probable configuration and 2) Using the most probable configuration as a template for crossover.

In Algorithm 4, the MAP is generated using BP-decimation and it is passed to the new generation together with the elitist solutions E, and the $N - |E| - 1$ solutions generated from the probabilistic model using GS or PLS. This is essentially the same idea used in previous EDA work on the use of the most probable configurations [7,8]. However, there are also some relevant differences:

- The factor graph is constructed from a Markov network that will represent two different types of structural information. Reduced set of cliques, for MN-FDA^G, and neighborhood based cliques, for MOA.
- Three different methods are used to compute the most probable configuration: junction-tree-based method for computing exact MAP, loopy belief propagation (LBP) for computing approximate MAP, and decimation-based LBP for computing approximate MAP [9].
- An archive is used as a memory of the EDA to avoid adding MAPs already found in previous generations. The archive stores at most one new MAP in each generation.

Algorithm 4. Insert-MAP

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

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

In Algorithm 5, the MAP is used as a template for recombining the information of the solutions in the current population with the global information contained in the Markov model. Recombination guarantees that partial configurations that coincide with the MAP will remain in the next population. In Algorithm 5, inference is applied only once in each generation and PLS or Gibbs sampling are not applied. It has been acknowledged [22] that operators that generate offspring through combination of global statistical information and the local information of solutions found so far (e.g., guided mutation) can improve search efficiency.

Algorithm 5. Template-MAP

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

5 Experiments

To evaluate the algorithms introduced in this paper we use separable additively decomposable functions (ADFs) defined on discrete non-binary variables.

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

$$F_{dec}(x_1, \ldots, x_k, k, c) = \begin{cases} k\cdot(c-1), & for \sum_{i=1}^{k} x_i = k\cdot(c-1) \\ k\cdot(c-1) - \sum_{i=1}^{k} x_i - 1 & otherwise \end{cases} \tag{3}$$

The general deceptive function $f^c_{deceptivek}(\mathbf{x})$ of order k [17] 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 family of functions is a generalization of the binary $f_{deceptivek}(\mathbf{x})$ to variables with non-binary values. Its difficulty increases with the cardinality of the variables (c) and the size of the definition sets (k).

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 $l = 30$ successive experiments and the average number of evaluations is computed from a maximum of $r = 30$ 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). The population size is started at $N = 32$ and doubled if the algorithm does not converge in the l experiments until a maximum $N = 131072$. The maximum number of generations was $g = 40$. Truncation selection, $T = 0.5$, is applied together with best elitism in which the complete selected population is passed to the next generation.

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

1. EDAs (MN-FDA^G and MOA)
2. Strategy for generating new solutions: S1) Insert-MAP. S2) Template-MAP. S3) Insert-MAP + Template-MAP

We also evaluate the influence that using different inference methods has in the behavior of the EDA. 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

The choice of MN-FDA^G or MOA influences the type of undirected model that is learned from data but also the strategy used for generating new solutions. When Insert-MAP is combined with NoMAP it means that no MAP is computed and therefore the original method used by MN-FDA^G or MOA to generate new solutions is applied. In total, we compare $2 \times 3 \times 4 - 2 = 22$ variants of the algorithms.

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

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

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

Table 1. Results of the EDAs variants for $f^c_{deceptivek}(\mathbf{x})$, $k \in \{3,4,5\}$, $c \in \{2,3,4,5\}$. Number of runs that achieved the optimum.

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

- As the size of the definition sets is increased the results of all algorithms deteriorate.
- Insert-MAP is a marginally better strategy than Insert-MAP+MAP-Template for MN-FDA^G, and a better strategy for MOA, as it can be observed for $n = 32, k = 4$.
- For Insert-MAP, there are not important differences between exact and approximate inference algorithms.
- MAP-Template is the worst generation strategy for all the problems.

We also investigate the number of function evaluations required by the algorithms to find the optimum for each combination of k and c. Figures 1, 2, and 3 show this information as boxplots. On each box, the central mark is the median number of evaluations from those runs (out of 30) in which the optimum was found. The edges of the box are the 25-th and 75-th percentiles. The whiskers extend to the most extreme datapoints the plotting algorithm considers to be not outliers, and the outliers are plotted individually as crosses.

A first finding from the analysis of the figures is that, for a fixed k, as the cardinality of the variables is increased, all algorithms dramatically decrease their performance. This can be seen in the fact that some of the algorithms are not able to find the optimum even once, and in the dramatic increase in the number of evaluations. For a fixed c (e.g., $c = 2$), the number of evaluations also increases with k. The combination of a high cardinality and a high definition set is particularly harmful to all the algorithms. For $c = 5$, only functions with $k = 3$ are solved.

Figures 1, 2, and 3 reveal that Insert-MAP is a more efficient algorithm than No-MAP and Insert-MAP+MAP-Template. The advantage of using Insert-MAP over No-MAP can be clearly observed for $c = 5$ for which MN-FDA^G with No-MAP never finds the optimum. The figures help to realize that Insert-MAP outperforms Insert-MAP+MAP-Template significantly. For all combinations of c and k the average number of evaluations needed by Insert-MAP is smaller than that needed by Insert-MAP+MAP-Template. Another interesting observation

can be made from the analysis of the outliers in Figure 2, $(k = 4, c = 2)$. Although the average behavior of MOA is worse than MN-FDA^G for $s1$, there are runs were the number of evaluations needed by MOA to find the optimum is smaller than the number of evaluations required by the best run of MN-FDA^G. This fact indicates that MN-FDA^G is more stable but MOA can eventually require a smaller number of evaluations in some runs.

Once possible explanation of the poor performance achieved by Template-MAP is that it produces a lack of diversity in the population that leads to search stagnation. Insert-MAP+MAP-Template attempts to counteract this effect but the result does not outperform Insert-MAP. Another issue that should be taken into account in the analysis of the experiments is that the critical population size has been computed by requiring to reach the optimum in $l = 30$ successive experiments. This is a very strong condition and may explain that for some functions the success rate is close but not equal to 30.

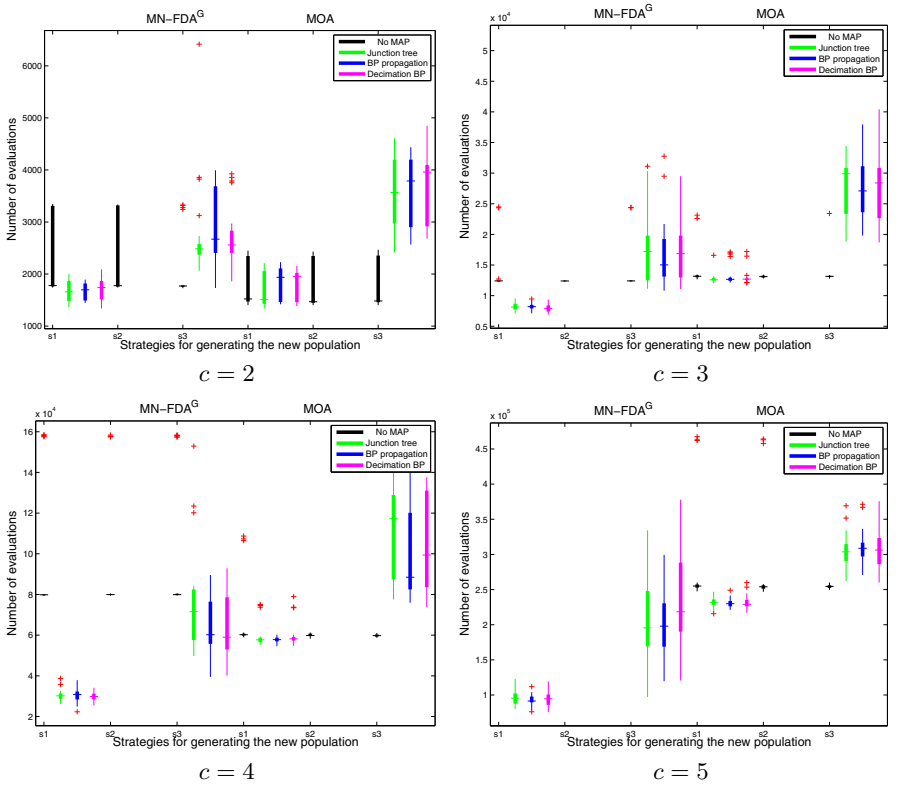
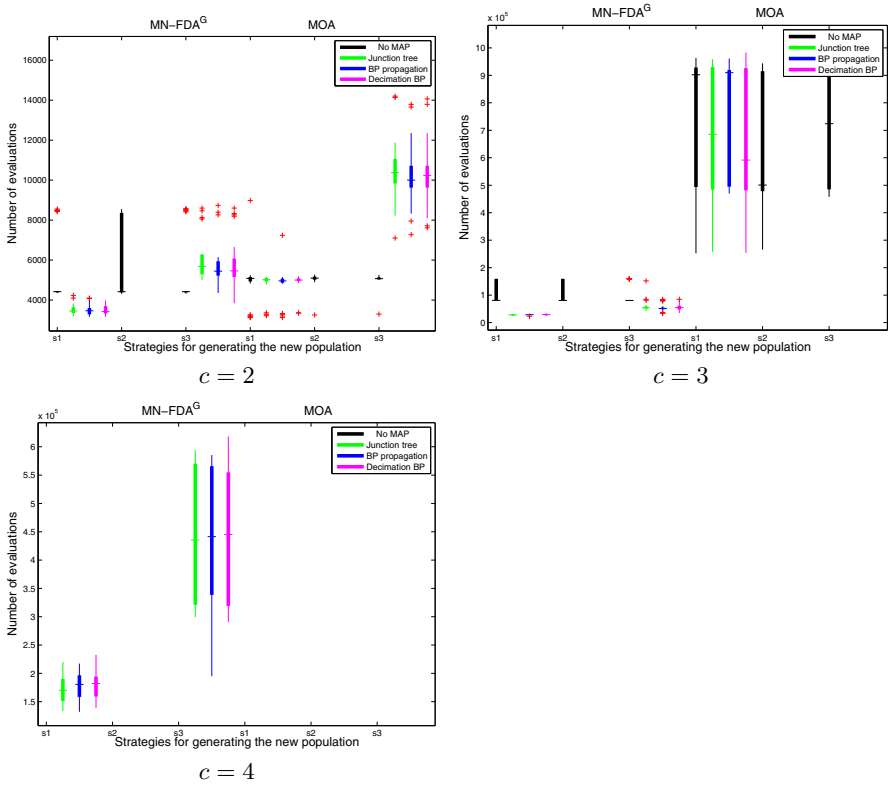


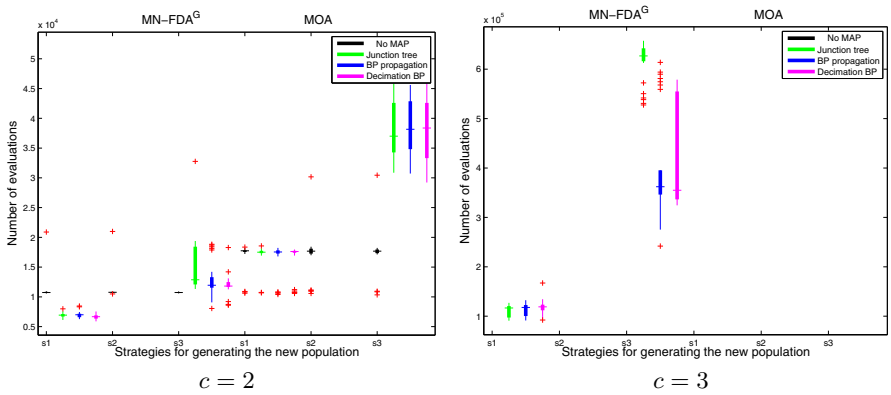
Fig. 1. Results of the different strategies for generating new solutions and inference methods for $f^c_{deceptivek}(\mathbf{x})$, $n = 30$, $k = 3$, $c \in \{2, 3, 4, 5\}$



$c = 2$ $c = 3$

$c = 4$

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



$c = 2$ $c = 3$

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

6 Conclusions

In this paper we have introduced new strategies for generating solutions in EDAs based on Markov networks and presented ways in which they can be combined. We have compared exact and approximate inference strategies in the context of EDAs based on Markov networks and we have shown that the choice of the inference procedure can help to dramatically reduce the number of evaluations.

It has been acknowledged [10] that "While the computation of the conditional probabilities and even learning of the Markov network structure can be efficiently done, efficient sampling of Markov networks is still an unsolved problem." The same argument has been exposed or implicitly assumed by other authors from the field. Although we do not claim that the use of inference techniques contribute to a more accurate sampling of the Markov network probability distribution, our experiments show that focusing on the MAP of the Markov network can actually improve the optimization results. The lesson to be taken is that the sampling problem could be somewhat relaxed by aiming at directly generating the solutions with highest probabilities instead of sampling the full distribution using Gibbs sampling. This is what strategies for generating new solutions based on approximate inference methods for MAP are able to do.

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. Kroc, L., Sabharwal, A., Selman, B.: Message-passing and local heuristics as decimation strategies for satisfiability. In: Proceedings of the 2009 ACM Symposium on Applied Computing, pp. 1408–1414. ACM (2009)
2. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
3. Larrañaga, P., Karshenas, H., Bielza, C., Santana, R.: A review on probabilistic graphical models in evolutionary computation. Journal of Heuristics 18(5), 795–819 (2012)
4. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
5. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer (2006)
6. McDonald, J.: Handbook of biological statistics, vol. 2. Sparky House Publishing, Baltimore (2009)
7. Mendiburu, A., Santana, R., Lozano, J.A.: 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)

8. Mendiburu, A., Santana, R., Lozano, J.A.: Fast fitness improvements in estimation of distribution algorithms using belief propagation. In: Santana, R., Shakya, S. (eds.) Markov Networks in Evolutionary Computation, pp. 141–155. Springer (2012)
9. Mooij, J.: libDAI: A free and open source C++ library for discrete approximate inference in graphical models. The Journal of Machine Learning Research 11, 2169–2173 (2010)
10. Mühlenbein, H.: Convergence theorems of estimation of distribution algorithms. In: Shakya, S., Santana, R. (eds.) Markov Networks in Evolutionary Computation, pp. 91–108. Springer (2012)
11. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
12. Pearl, J.: Causality: Models, Reasoning and Inference. Cambridge University Press (2000)
13. Santana, R.: A markov network based factorized distribution algorithm for optimization. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 337–348. Springer, Heidelberg (2003)
14. Santana, R.: Estimation of distribution algorithms with Kikuchi approximations. Evolutionary Computation 13(1), 67–97 (2005)
15. Santana, R.: MN-EDA and the use of clique-based factorisations in EDAs. In: Shakya, S., Santana, R. (eds.) Markov Networks in Evolutionary Computation, pp. 73–87. Springer (2012)
16. Santana, R., Larrañaga, P., Lozano, J.A.: Research topics on discrete estimation of distribution algorithms. Memetic Computing 1(1), 35–54 (2009)
17. Santana, R., Ochoa, A., Soto, M.R.: 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. Shakya, S., McCall, J.: 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. Shakya, S., Santana, R.: An EDA based on local Markov property and Gibbs sampling. In: Keijzer, M. (ed.) Proceedings of the 2008 Genetic and Evolutionary Computation Conference (GECCO), pp. 475–476. ACM, New York (2008)
20. Shakya, S., Santana, R. (eds.): Markov Networks in Evolutionary Computation. Springer (2012)
21. Shakya, S., Santana, R., Lozano, J.A.: A Markovianity based optimisation algorithm. Genetic Programming and Evolvable Machines 13(2), 159–195 (2012)
22. Zhang, Q., Sun, J., Tsang, E.P.K.: Evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Transactions on Evolutionary Computation 9(2), 192–200 (2005)