# Graphical Markov Model Learning with a Double-Stressed Genetic Algorithm (GML-2SGA)

Elva Díaz[1] , Eunice Ponce de León[1] , Felipe Padilla[1]

[1] Departamento de Sistemas Electrónicos
Centro de Ciencias Básicas
Universidad Autónoma de Aguascalientes
elva.diaz@itesm.mx
{eponce, fpadilla}@correo.uaa.mx

**Abstract.** The graphical model learning is considered as a multiobjective optimization problem and a genetic algorithm added with two stressors over the exploration and exploitation capabilities (GML-2SGA) is presented here and applied to obtain a Pareto front. To stress the exploitation of the GA a Metropolis step is added to the genetic operators, and to stress the exploitation, independent samples are taken from different combinations of the GA parameter values. An experimental design is executed using benchmarks of complexities sparse, medium and dense, generated with a Markov Model Random Sampler (MMRS). It is compared to GMS-MGA, and the performance is assessed using the mean number of cliques of the true model identified by the algorithms. As results, the algorithm presented here identifies more cliques of the true models than the one used to compare, in all complexity types, and the complexity of the models made a difference in the performance of the algorithms.

## 1 Introduction

A new paradigm known as Estimation of Distribution Algorithms (EDAs) in Evolutionary Computation [12] [17] makes use of graphical model estimation and selection (learning) to guide the search in the space of solutions in optimization problems. This has made model estimation and selection an important tool in Evolutionary Computation. However, graphical model selection is not an ease task. The space of models grows exponentially with the number of variables, and there is no polynomial time certificate for a model if one were available. So, one way to tackle the problem is to construct a heuristic.

The genetic algorithm has been used to learn models in Poli and Roverato [19] and Roverato and Paterlini [20]. In those papers the problem of learning a model was considered as an optimization one, using the Akaike index as a simple criterion to optimize. In Díaz and Ponce de Leon [6] the problem of learning models was considered a multiobjective optimization one. The two objectives are (1) the fitting of the model to the data measured by the Kullback-Leibler deviance, and (2) the simplicity of the model, measured by the number of edges of the graph. The two

*A. Gelbukh, S. Suárez, H. Calvo (Eds.)*
*Advances in Computer Science and Engineering*
*Research in Computing Science 29, 2007, pp. 3-14*

*Received 16/09/07*
*Accepted 19/10/07*
*Final version 24/10/07*

objective functions are conflicting because the model that best fit the data is the one represented by a complete graph, whose fitting is zero. As the number of edges decrease the fitting grows. In Diaz and Ponce de Leon [6], a relative preference vector is used to convert the multiple objectives into a single one. In Ponce de Leon and Diaz [18] a Pareto front optimality criterion is used helped by a genetic algorithm (GMS-MGA).

Evolutionary Computation algorithms in search need to balance the extension of exploration of the space through recombination and mutation, with the extension of exploitation through the selection operator. If the solutions obtained are exploited too much, premature convergence is expected, but if too much stress is given on the search, the information obtained so far is not properly used, the execution time is enormous and the search exhibits a similar behavior to that of a random search. On the other hand, it is known that the convergence of a MOEA algorithm to the Pareto Front can not be assured by the performance of the genetic operators alone [5] [14]. This last author suggests introducing a Metropolis step to the genetic algorithm to assure the convergence. This step is really a local search stressor and the question is how the balance between exploration and exploitation is affected. To solve this problem, independent samples for a multistart method are used. Other related issue is confronted: the difficulty to design the genetic operators in such a way that the resulting offspring (a) is likely to inherit desirable properties of its parents, and (b) is capable of improving on its parents' solution quality. The GMS-MGA presented in [18], fulfill the (a) part. In the present paper the mutation and the recombination operators for discrete graphical models introduced in [6] and used in [18] are modified to fulfill both parts (a) and (b). The class of Evolutionary Algorithms obtained in this form could be put in the class of the some times called Memetic Algorithms and some other times Genetic Local Search Algorithms [14]. But the algorithm presented here is added with two stressors over the exploration and exploitation capability of the genetic algorithm, so, it will be named Graphical Markov Model Learning with a two Stressed Genetic Algorithm (GML-2SGA).

## 2 Content

The content of the paper is:
(1) to define the graphical Markov model representation, (Section 3)
(2) to define the multiobjective graphical Markov model learning problem, (Section 4)
(3) to define the stressing methods used: Multi-start method and Metropolis step with Boltzman distribution, (Section 5)
(4) to define the Graphical Markov Model Learning with a two Stressed Genetic Algorithm (GML-2SGA), (Section 6)
(5) to perform an experiment to compare the two algorithms, GMS-MGA and GML-2SGA (Section 7),
(6) to discuss the results and obtain conclusions (Sections 8, 9).

# 3 The Graphical Markov Model Class

In this section first, the graphical Markov model and its hypergraph representation are defined, and second, the hypergraph operators to handle the models are defined.

## 3.1 The Graphical Markov Model Class

A graphical Markov model consists of an undirected graph and a subclass of probabilistic distributions for random variables that can be represented by this undirected graph [13]. "Represented" means that the set of nodes of the graph are the random variables, and that an edge between two nodes is absent, when theses two random variables are conditionally independent given the rest of the variables in the graph. The graph that has this last property (represent the structure of interactions of the random variables) is known as Markov graph and the graph together with the class of distributions is known as a probabilistic graphical Markov model.

In this paper only discrete binary random variables are considered, but the probabilistic graphical model class used in this paper is unrestricted, meaning it that every simple graph can be used to represent the probabilistic conditional independences between the random variables of the probability model. In this type of models, it is necessary to use some iterative algorithm to perform the parameter estimation. In this paper a modification of the generalized iterative scaling [4] that will appear soon, is used. The concepts, definitions and properties of graphical models are taken from the book of Lauritzen [13].

## 3.2 Hypergraph Representation of a Discrete Graphical Markov model

The graphical models can be characterized by a set E= $\{E_1,...,E_k\}$ of pair-wise incomparable (w.r.t. inclusion) subsets from V, named the generating class, that is to be interpreted as the maximal sets of permissible interactions [13].

To represent a graphical model in a convenient way a hypergraph structure is used. A hypergraph is a collection of subsets, $H = \{H_1, H_2, ..., H_k\}$ named hyper edges, of a finite set V of nodes of a graph. Each hyper edge corresponds to a clique of the graph. This collection of subsets satisfies the same property as the generating class, that is, they are pair-wise incomparable (w.r.t. inclusion) [2]. So, a one to one correspondence can be established between the generating class E and the hypergraph H. [13]

A discrete graphical Markov model can be represented by M = (H, P(X)) where H is a hypergraph and P(X) is a class of probability distributions [13].

The intersection between two hypergraphs $H_1$ and $H_2$, is the family of sets intersections taking one hyper edge from each of the two hypergraphs,

$$H_1 \wedge H_2 = \{h_i \wedge h_j, \forall h_i \in H_1 and \forall h_j \in H_2\} \quad (1)$$

This operation between two hypergraphs will be used in the next section to handle with hypergraphs models.

### 3.3 Operators to Handle with Hypergraphs Models

To handle with hypergraph models, different types of operators are needed.

The Nabla menus $\nabla$- operator is a hypergraph used to define a unary operator over a hypergraph. It is defined by two binary chains:

$\nabla$- = ($b_i$,$b_j$)', where $b_i$ = (1 1 1 1 0 1 1 1 1), and where the number 0 is in the position i, and where $b_j$ = (1 1 1 1 1 0 1 1), where the number 0 is in the position j.

Intersecting Nabla menus with an hypergraph, the edge (i, j) is taken out of the graph, that is,

$\nabla$- $\wedge$H is one edge less than H.

The Nabla plus is defined in a similar but more complicated way (See [6]). The intersection of two hyper graphs is used as the crossover operator, and the Nabla menus and Nabla plus operators are used as mutation operators to take an edge out and to put an edge in, in a graph [6].

## 4 Multiobjective Graphical Markov Model Selection

### 4.1 Model Learning from Data: A Multiobjective Optimization Problem

Model learning from data problem is an optimization problem with two objective functions. First, the learned model must be as near to the data sample as possible (best fitting), and second, the model must be, as simple as possible to evade the over fitting. Best fitting to the data means, that the Kullback-Leibler deviance [1], [11] from the model to the sample is minimum. The Kullback-Leibler deviance from the complete graph to the sample is null, but, at the same time this model contains all the data noise. Objective functions for each of the tasks are needed, that is, an objective vector function with two components must be considered. Let denote it as O= ($O_1$, $O_2$). For the first task, the Kullback-Leibler divergence, or relative entropy is used, and for the second task a measure of complexity, the number of edges, is used. The problem is, to minimize the two elements of O. The two objective functions are conflicting because the model that best fit the data is the complete graph, and the Kullback-Leibler divergence grows when the edges number decreases.

Classical multiobjective optimization methods convert multiple objectives into a single one by using a relative preference vector of preferences [6]. Unless a reliable and accurate preference vector is available, the optimal solution obtained by such methods is highly subjective. One way out of this problem is to use the Pareto optimality. The Pareto optimality is based on the concept of dominance [5]. A model is not dominated when there is no other model in a class that is better in all the objectives. The subclass of non dominated models in a class is the Pareto front. The

Pareto front is a solution set and not a single solution. So, a multiobjective optimization problem requires multiple trade-off solutions to be found. The task is then, to find as many different trade-off solutions as possible. A way to find many different solutions is to use an Evolutionary Algorithm (EA) that works with a population of solutions. This ability of an EA makes it unique in solving multiobjective optimization problems. The ability of a Genetic Algorithm (GA) to obtain multiple optimal solutions in one single simulation run makes it especially well prepared to play an important role in finding the Pareto front.

In this paper a multi-start GA is used to generate as many different trade-off solutions as possible, and then a multiobjective selection algorithm is used to obtain a Pareto front. In order to use a GA, the hypergraph model representation, and the operators to manipulate models defined in section 3 are used.

### 4.2. Objective Function Definition

Let define the components of the objective function O= ($O_1$, $O_2$).

The fitting function $O_1$, is defined based on Information Theory criteria [11], [14].

Let $L(\hat{m}_n^M \mid x)$ be the likelihood ratio statistic for a sample of size n from a multinomial distribution, where $\hat{m}_n^M \mid x$ is the maximum likelihood estimate [3], [4], [9] assuming that the true model is $M$ . The log likelihood ratio in the multinomial case is

$$G^2(M) = \log L(\hat{m}_n^M \mid x) = -2\sum x_i \log \frac{\hat{m}_i^M}{x_i} \tag{2}$$

Then the objective function to minimize is:

$$O = (O_1, O_2) = (G^2(M), l(M)) \tag{3}$$

where l(M) is the number of edges of the model M.

Let

$$G(M, M_0) = \frac{G^2(M_0) - G^2(M)}{G^2(M_0)} \tag{4}$$

where $M_0$ is the equiprobability model.

Let

$$l(v, M) = \frac{\max l_v - l(M)}{\max l_v} \tag{5}$$

where $\max l_v$ is the maximal number of edges formed with $v$ vertices.

The aggregated convex fitting index for the model $M$ is defined as in [6] by

$$CFI(M) = p(G(M,M_0)) + (1-p)(l(v,M)) \tag{6}$$

This index is used as a tool in the genetic algorithm.

## 5 Stressing the GA

### 5.1 Multi-start Method

The multi-start method consists of a strategy for searching parts of the space, starting from different initial solutions [10]. The objective of these methods is to avoid falling in a local optimum. These methods provide an appropriate tool to introduce diversity to the population of the evolutionary algorithm. In this paper the objective of a multi-start method is to visit different parts of the search space, widening the exploration. The parameter values of the genetic algorithm are used to define the multi-start method. Fixing the per cent selected to reproduce, the probability to mutate, and the weight assigned to fitting against the simplicity of the models, the genetic algorithm is repeated for each combination of values. (See Table No. 1). The exploitation is widened when more values of the parameters are tested. The multi-start method was tested with three values for each parameter (27 start points) and with two values for each parameter (8 start points) and there was no difference in the results. So, the method with two values for each parameter, were used to perform the full experiment. For each combination of the parameters values, 5 random samples are generated. A total of 40 independent samples are taken.

### 5.2 Metropolis Step and the Boltzman Distribution

The well-known Metropolis step is a fundamental part of the Simulated Annealing (SA) algorithm. It was first introduced by Metropolis [16] to simulate the physical annealing process of solids. With the same words of Metropolis, let E(X) be the energy of a solid X, then the solid is perturbed, let E(X') be the energy of the perturbed solid and the objective of the algorithm is to accept a new state X' if its energy E(X') is lower than the energy E(X). The decision of accepting a new state is made by the $\alpha$ criterion defined as:

$$\alpha = \exp\left(-\frac{\delta E(X)}{kT}\right) \tag{7}$$

where

$$\delta E(X) = E(X') - E(X) \tag{8}$$

T denotes the temperature and k is the Boltzman constant. For k=1, at each T the SA algorithm aims to draw samples from the Boltzman equilibrium distribution:

$$\pi_T(x) \propto \exp(-\delta(E(X))/kT) \quad (9)$$

For the experiments performed in this research k=1 and T=1 are used.

## 6 Two Stressed Genetic Algorithm (GML-2SGA)

In this section a description and a pseudocode of the GML-2SGA algorithm and each of its parts are given.

### 6.1 The Main Stressed Algorithm

The main stressed genetic algorithm is sketched in Algorithm 1.

To learn a graphic Markov model an approximate Pareto front is initialized and updated at each step of the stressed genetic algorithm, to obtain after k generations a list of models (the approximate Pareto front) that contains the approximate best model for each number of edges. To diversify the searching part of the algorithm (stress exploring), the genetic algorithm is added with a multi-start method that obtains a sample with one of a list of genetic parameters combinations. The genetic populations obtained are used to update the approximated Pareto front [5]. To stress the exploitation part of the algorithm an M-step algorithm is added to the genetic operators.

Algorithm 1. Main stressed GA (GML-2SGA)

```
begin
   Initialize the Pareto Front
   repeat
      Initialize the diversity parameters
      Choose an initial population
      Calculate the number of edges and the fitness of
      each model
      repeat
         Perform truncation selection for the population
         i
         Perform M-step crossover or mutation
         Calculate the number of edges and the fitness
         of each model
        Actualize the Pareto front
      until (Some stopping criterion applies (k
            populations))
   until (Multi-start parameters combination are over)
end
```

### 6.2 The M step Crossover and Mutation Operators

The hybridized crossover and mutation operators include an M-Step. They leave the class of graphical Markov models closed, and conserve (specially the crossover operator) the heritability of the common interaction structures of the parents.

Algorithm 2. M-Step crossover algorithm

```
begin
   Select  X, Y from Pop
   repeat
       Z←crossover(X, Y)
       δE(Z|X,Y)←E(Z)-E(X,Y)
       U←rand(0,1)
       if (U< min{1,exp(-δE(Z|X,Y))}) then Pop←Z
   until (Some stopping criteria applies)
end
```

Algorithm 3. M-Step mutation algorithm

```
begin
   Select  X from Pop
   repeat
      Z←mutation(X)
      δE(Z|X)←E(Z)-E(X)
      U←rand(0,1)
      if (U< min{1,exp(-δE(Z|X))}) then Pop←Z
   until (some stopping criteria applies)
end
```

Algorithm 4. M-step crossover or mutation

```
begin
   With probability p select two parents: obtain an
   offspring by crossover
   Assess the offspring fitness
   if (The offspring is not dominated by one of the
       fathers)
      then Add it to the new population
      else Add it to the new population with M step
           criterion for crossover
   With probability 1-p select one parent: with
   probability q take one edge out
   Assess the offspring fitness
   if (The child is not dominated by the father)
      then Add it to the new population
      else Add it to the new population with M step
           criterion for mutation
   With probability 1- q: put one edge in
   Assess the offspring fitness
   if (The child is not dominated by the father)
      then Add it to the new population
```

```
        else Add it to the new population with M step
             criterion for mutation
end
```

### 6.3 Algorithm Components Description

Multi-start estates: The parameter combinations of the genetic algorithm are used as diversity factors. They are: $\tau$ % of individuals selected to reproduce (25, 50); p % of individuals selected to mutate (25, 45). Parameter of the convex index (0.60, 0.45) Initial population: We form the initial population taking one edge out from the saturated model. To attain this objective the Nabla menus operator is used. Then, $\xi$ % of the best evaluated models is taken to begin the genetic algorithm.

Fitting: The convex fitting criterion acts as the genetic algorithm fitness.

The Pareto front: for each number of edges, the best adjusted model is saved at each population of the genetic algorithm.

In a random fashion the algorithm decides to do crossover with probability p or mutation with probability 1- p. The crossover operation of two models is defined by the binary intersection operator.

The mutation operator could take one edge out at random with probability 0.7 (using the operator Nabla menus) or put an edge in (using the operator Nabla plus) at random with probability 0.3.

## 7 Experimental Design

To compare the two algorithms with models of different complexities [8], an experiment was designed and run. A conditional independence restrictions structure was selected at random of the type dense, mean and sparse. The structures are defined by its cliques (See table 2). To asses the performance of the algorithms, simulated samples from known models of 10 and 12 variables, sparse, medium and dense (see Table 2), are generated with a Markov Structure Random Sampler (MSRS) presented in [7].

Each algorithm is run 6 times with each type of model. The experiment and the algorithm is programmed in C++ and executed in a Pentium IV PC at 1.6 MHz.

**Table 1**. Starting values

| Per cent selected to reproduce | Per cent to mutate | Convex fitting index parameter |
|---|---|---|
| 25 | 25 | 0.60 |
| 50 | 45 | 0.45 |

## 8 Results and Discussion

The mean execution time of the algorithm proposed in this paper is ≤ 5.12 minutes for 10 variables, and ≤ 14.28 minutes for 12 variables in C++, over a PC at 1.6 GHz. The sparse models are more difficult to identify because they have few edges, and are immersed in a huge space like a "Needle in a Haystack". The mean execution time for dense models is 21 seconds more than for medium models in the case of 10 variables, and 3.87 minutes more in the case 12 variables, indicating that the complexity of the models make the computations time longer. Some explanation for that could be that the independent sample sizes are not enough to produce convergence in more complex models.

The performance of the GML-2SGA algorithm is better than the performance of the GMS-MGA, in the case of the sparse and the medium models, in the case of the dense model the performance are essentially the same (See Table No. 2).

**Table 2**. GMS-MGA vs. GML-2SGA

| MODELS | MODEL'S MAXIMAL CLIQUES | GMS-MGA Correct Cliques Mean | GML-2SGA Correct Cliques Mean | Mean Run time |
|---|---|---|---|---|
| 10 VAR. SPARSE | AB BC CD DE EF FG GH HI IJ (9 cliques) | 6.65 | 7.35 | 5 min. |
| 10 VAR. MEDIUM | ABC CDE EFG GHI JA (5 cliques) | 3.95 | 4.92 | 4.91 min. |
| 10 VAR. DENSE | ABCD DEFG FGHI HIJA (4 cliques) | 2.75 | 2.75 | 5.12 min. |
| 12 VAR. SPARSE | AB BC CD DE EF FG GH HI IJ JK KL (11 cliques) | 9.43 | 10.12 | 11.71 min. |
| 12 VAR. MEDIUM | ABC CDE EFG GHI IJK JKL (6 cliques) | 4.11 | 5.83 | 10.41 min. |
| 12 VAR. DENSE | ABCD DEFG FGHI HIJK IJKL (5 cliques) | 3.98 | 4.27 | 14.28 min. |

## 9 Conclusions and Recommendations

The sparse models are more difficult to identify because they have few edges, and are immersed in a huge space like a “Needle in a Haystack”. The GML-2SGA for binary variables has a better performance than the GMS-MGA because the sparse models are not more difficult to determine than the medium and dense ones as was the case with the GMS-MGA. The medium complexity models are almost always completely determined. The execution mean run time grows with the number of nodes (variables) in the model and with the complexity (sparse, medium, and dense) of the graphical model.

## References

1. Akaike, H: A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (1974) 716-23.
2. Berge, C.: Graphs and Hypergraphs, North- Holland (1973).
3. Birch, M. W.: Maximum-likelihood in three way contingency tables, Journal of the Royal Statistical Society, Series B 25 (1963) 220-33.
4. Darroch, J. N., Ratcliff, D.: Generalized iterative scaling for log-linear models. Annals of Mathematical Statistics, 43 (1972) 1470-80.
5. Deb, K.: Multi-objective optimization using evolutionary algorithms, Wiley, Chichester, UK, (2001).
6. Díaz, E., Ponce de León, E.: Discrete Markov Model Selection by a Genetic Algorithm, Avances en Ciencias de la Computación e Ingeniería de Cómputo Vol II, Ed. Sossa Azuela, Aguilar Ibáñez, Alvarado Mentado, Gelbukh Khan, IPN, (2002) 315-324.
7. Díaz, E., Ponce de León, E.: Markov Structure Random Sampler Algorithm (MSRS) from unrestricted Discrete Models, Proceedings Fifth Mexican International Conference on Artificial Intelligence MICAI 2006, Special Session IEEE Computer Society, Eds. Alexander Gelbukh, Carlos Alberto Reyes García (2006) 199-206.
8. Díaz, E., Ponce de León, E.: Modelling and Simulation Complexity for Discrete Graphic Markov Models: an Experimental Study, WSEAS Transactions on Mathematics, Issue 1 Volume 3 January (2004) 288-291.
9. Díaz, E., Ponce de León, E.: Discrete Graphic Markov Model Selection by a Genetic algorithm based on Different Estimation of Distribution Algorithms, WSEAS Transactions on Mathematics, Issue 2 Volume 4 April (2005) 110-116.
10. Hickernell, F.J.,Yuan, Y.: A Simple Multistart Algorithm for Global Optimization, OR Transactions, Vol 1 (1997) 1-11.
11. Kullback, S., Leibler, R. A.: On information and sufficiency, Ann. of Math. Statistics, 22, (1951) 79-86.
12. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithm: A new tool for Evolutionary Computation, Kluwer Academic Publisher (2001).
13. Lauritzen, S.L.: Graphical Models, Oxford Science Publications, (1996).
14. Mc Kay, David J. C.: Information Theory, Inference and Learning Algorithms, Cambridge University Press (2003).
15. Merz, P., Freisleben, B.: Fitness Landscape and Memetic algorithm Design, In New Ideas in Optimization. Ed. By D. Corne, M. Dorigo, and F. Glover, Mc Graw Hill (1999) 245-260.

16. Metropolis N., Rosenbluth A. E., Rosenbluth M. N., Teller A. H., Teller, E.: Equation of state calculations by fast computing machines, J. Chem Phys 21 (1953) 1087-1092.
17. Ponce de León, E., Díaz, E., Padilla, F.: Evolutionary Algorithm based on Markov graphical models of promising solutions, Research on Computing Science. Advances in Artificial Intelligence, Computing Science and Computer Engineering, Vol. 10: Eds. Jesús Figueroa Nazuno, Alexander Gelbukh Khan, Cornelio Yánez Márquez, Oscar Camacho Nieto, IPN (2004) 341-347.
18. Ponce de León, E., Díaz, E.: Discrete Graphic Markov Model Selection by Multiobjective Genetic Algorithm (GMS-MGA). Proceedings 15th Internacional Conference on Computing CIC 2006, IEEE Computer Society, Eds. Alexander Gelbukh, Sergio Suarez Guerra (2006) 18-23.
19. Poli, I., Roverato, A.: A genetic algorithm for graphical model selection, J. Italian Statist. Soc. (2) (1998) 197-208.
20. Roverato, A., Paterlini, S.: Technological modeling for graphical models: an approach based on genetic algorithms, Comp.Statistic & Data Analysis 47, (2004) 323-337.