International Journal of Computational Intelligence Systems, Vol. 6, No. 6 (November, 2013), 1163-1188

Network measures for information extraction in evolutionary algorithms

Roberto Santana[1] , Rubén Armañanzas[2] , Concha Bielza[2] , Pedro Larrañaga[2]

[1] *Intelligent Systems Group*
Department of Computer Science and Artificial Intelligence
University of the Basque Country (UPV/EHU)
Paseo Manuel de Lardizábal 1. 20018 Donostia-San Sebastian, Spain
[2] *Computational Intelligence Group*
Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid,
Campus de Montegancedo, 28660 Boadilla del Monte, Spain
E-mail: roberto.santana@ehu.es; r.armananzas@upm.es; mcbielza@fi.upm.es; pedro.larranaga@fi.upm.es

Received 23 August 2012

Accepted 28 January 2013

Abstract

Problem domain information extraction is a critical issue in many real-world optimization problems. Increasing the repertoire of techniques available in evolutionary algorithms with this purpose is fundamental for extending the applicability of these algorithms. In this paper we introduce a unifying information mining approach for evolutionary algorithms. Our proposal is based on a division of the stages where structural modelling of the variables interactions is applied. Particular topological characteristics induced from different stages of the modelling process are identified. Network theory is used to harvest problem structural information from the learned probabilistic graphical models (PGMs). We show how different statistical measures, previously studied for networks from different domains, can be applied to mine the graphical component of PGMs. We provide evidence that the computed measures can be employed for studying problem difficulty, classifying different problem instances and predicting the algorithm behavior.

Keywords: Knowledge extraction, network theory, optimization, evolutionary algorithms, computational intelligence.

1. Introduction

Although the primary expected outcome of an evolutionary algorithm (EA) is the solution to a given optimization problem, there is an increasing need to add new capabilities to these algorithms. There are several situations in which the end-user may expect other benefits from the computational effort spent in the optimization. The user may be interested in acquiring a better problem understanding (knowledge extraction),[1,2] reusing the experience gained by the algorithm to solve similar optimization problems (transfer learning),[3] or evaluating and improving the way in which different components of the algorithm interact by analyzing its output (algorithm enhancement).[4,5]

In principle, since EAs use populations of solutions, they can be seen as automatic generators of information about the search space. This valuable information can be mined to extract knowledge about the problem domain. The conception of methods to such information extraction from EAs constitutes a relevant topic and it expands the natural scope of these algorithms. Mining the information gener-

ated by EAs may be useful in real-world applications such as bioinformatics, engineering or design, where structural interactions between problem components are usually only partially known. The key issues are then, first, how to capture and process this information and, secondly, how to transform such information into useful answers to question the end-users may face.

In this paper, we propose the aplication of methods and algorithms coming from network theory[6,7] in order to extract valuable information from the EA evaluation. Networks are convenient tools to represent relationships between the components of very diverse nature systems. The proposed framework can be seen as structural mining in a space of networks which are related to the objective function and the EA variation operators. The application of network theory to the analysis of EAs intends to capture complex relationships that arise between the problem variables in different subspaces of the search space. Research on the use of networks as an effective tool to model complex systems has experienced an important upsurge in recent years.[6–10] Network theoretic measures have been proposed to extract informative descriptors of such networks. These measures allow to abstract from the particular characteristics of the system's components and to reveal patterns of interactions between the variables of such system.

We focus on the network-based analysis of two types of evolutionary algorithms: Genetic algorithms (GAs)[11,12] and estimation of distribution algorithms (EDAs).[13–16] EDAs are population based optimization methods that share a number of similarities with GAs. However, while GAs use genetic operators, EDAs intensively apply learning of probabilistic models to domains where the underlying interactions between the variables are often unknown. Network measures can be applied to identify properties about the problem being solved by EAs, but also to serve as descriptors of the algorithms behavior. Topological measures extracted from the networks could be used to compare the difficulty of different optimization problems (e.g. number of optima), extract problem information (e.g. average number of generations needed to converge), or predict the behavior of the EA (e.g. number of times the optimum was reached). Once a mapping between the network measures and the characteristics of a problem is constructed from a set of training examples, it is possible to predict problem's features such as difficulty, favorable search regions or expected values for optimal solutions. End-user questions might be also answered.

Througout this paper, our main aims are, first, to show that the structural components of graphical models extracted from GAs and EDAs can be characterized as networks. Valuable information about the optimization problem and the behavior of the algorithms used to solve it can be extracted from the analysis of the networks. Our second aim is to introduce a methodological framework for practical implementation of network analysis of GAs and EDAs. This framework identifies *a priori*, *online*, and *a posteriori* structural knowledge on the application of network analysis.

The content of the paper is divided as follows. Section 2 presents network measures for information extraction with local and global approaches. The concepts of *a priori*, *online*, and *a posteriori* structural knowledge on the application of network analysis are introduced. The experimental framework and the numerical results of our experiments are introduced in Section 3. Subsections 3.1, 3.2 and 3.3 respectively present three optimization problems with different characteristics and discuss the results of our general approach when applied to these problems. Previous work in this field is discussed in Section 4 and conclusions and future lines are drawn in Section 5.

2. Network analysis

In recent years, results from graph theory have been developed and integrated in the modern theory of networks.[6–10] The term network is an informal description for a set of elements with connections or interactions between them.[17] We see an optimization problem as a network where different kinds of relationships between the variables of the problem are represented as connections in the network. To deal with networks in a formal way, they are usually

modeled as graphs. Similarly to networks, graphs comprise vertices and arcs representing elements and connections. They are mathematical structures that abstract for the particular interpretation given to a network.

There is an extensive set of measures defined for networks. Some of them have a direct interpretation within the probabilistic graphical models (PGM) application context. For instance, the tree-width of the junction tree, or equivalently, the size of the maximum-clique for the associated undirected graph, are related to the complexity of the inference step.[18] Apart from this and other few examples, most network measures have not been used in the context of PGMs.

For our analysis, we divide network measures into two groups. First, local measures which identify local patterns in network topologies (e.g. number of arcs outgoing from a given vertex). Second, global measures that serve to capture the global structure of a network (e.g. number of edges). In this group we include *meso* measures such as those related to the community structure and *contrastive measures* that reflect differences between two or more networks. A number of definitions related to network theory are following presented, focussing on the analysis of networks represented by directed graphs.

2.1. Local network measures

The *range* g_{ij} of an arc e_{ij}[19] is the length of the shortest path from j to i after arc e_{ij} has been removed from the graph. If $g_{ij} > 2$, then the arc forms a *shortcut* from j to i.

Node eccentricity is the maximal shortest path length between a node and any other node. Let k_i be the number of neighbors for vertex v_i. The maximal number of arcs between the neighbors of v_i is $\frac{k_i(k_i-1)}{2}$.

The *clustering coefficient* C_i of v_i is defined as the fraction of the existing number of node arcs ($|A_i|$) to the total possible number of neighbor-neighbor arcs:[20]

$$C_i = \frac{2|A_i|}{k_i(k_i-1)} \,. \tag{1}$$

Let λ_{st} be the total number of shortest paths between vertices s and t and let $\delta_{st}(v)$ denote the fraction of shortest paths between s and t that pass through a particular vertex v, i.e. $\delta_{st}(v) = \frac{\lambda_{st}(v)}{\lambda_{st}}$. *Betweenness centrality* of a vertex v is defined as[21]

$$BC(v) = \sum_{s,t:s\neq v\neq t} \delta_{st}(v) \,. \tag{2}$$

Similarly, *edge betweenness centrality* is the fraction of all shortest paths in the network that traverse a given edge.[22] A *clique* is a set of nodes that are all adjacent to each other, i.e. a maximal fully-connected sub-graph.

2.2. Global network measures

The *assortativity coefficient* is a correlation coefficient for the degree of nodes that are joined by an arc (linked nodes). A positive assortativity coefficient indicates that nodes tend to link to other nodes with the same or similar degree.[23] Assortativity coefficients can be also used to capture dependencies between any node property that might be of interest. However, in this paper we restrict the analysis to the correlations between node degrees.

The *characteristic path length* of a graph is the average shortest path length between every pair of reachable vertices in the graph. *Network radius* is the minimum eccentricity and *network diameter* is the maximum eccentricity.

A *structural motif of size* Z[24] is a connected graph with Z vertices. For each Z there is a limited set of distinct structural motifs which are called motif classes.

A *motif frequency spectrum* records the number of occurrences of each motif of a given class for a size Z. *Motif number* is the total number of all motifs of any class (for a given size Z) encountered in a network. The motif number is obtained as the sum over the motif frequency spectrum. Motif analysis can be seen as a generalization of the clustering coefficient.[25]

Figure 1 shows all structural motifs for a motif class of size $Z = 3$. Figure 2 shows an example of a directed network. The motif frequency spectrum of this network is $(1,5,1,0,1,0,0,0,0,0,0,0,0)$, where

motifs are ordered as in Figure 1. The network motif number is 8.

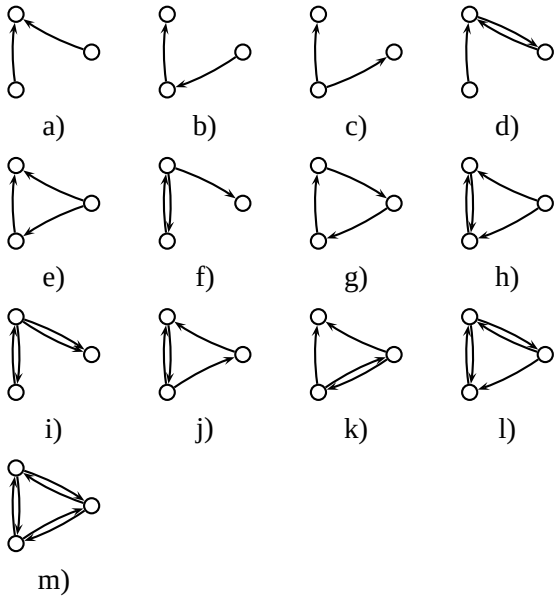

Figure 1: All structural motifs for a motif class of size $Z = 3$.

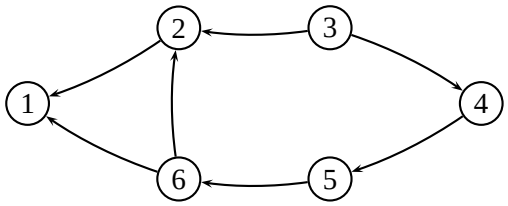

Figure 2: Example of a directed network.

A *module* is usually recognized as a unit that is a component part of a larger system and yet possesses its own structural or functional identity.[26] There is no generally accepted definition of what constitutes a module within a complex network. It is generally associated to a densely connected subset of nodes that is only sparsely linked to the remaining network. Score functions are defined to measure the *degree of network modularity*.

We propose the use of two different approaches for modularity detection: Newman's spectral algorithm[27] and the Louvain method for community detection in large networks.[28] Newman's algorithm is based on the idea that modularity can be expressed in terms of the eigenvectors of a characteristic matrix for the network (modularity matrix). The leading eigenvector of the modularity matrix is computed and the vertices are divided into two groups according to the signs of the elements in this vector. The decomposition process is continued until the entire network has been decomposed into indivisible subgraphs.[27] The Louvain method is a greedy algorithm divided into two phases that are repeated iteratively. First, the method looks for small communities by optimizing modularity locally. Second, it aggregates nodes belonging to the same community and builds a new network whose nodes are the communities.[28]

2.3. Collection of network measures

On the basis of the introduced concepts, we chose a set of key network measures which are able to capture important characteristics of the EA dynamics. The whole set of selected measures follows:

1. *dagdif*: Number of different arcs between the directed acyclic graphs (DAGs)[29] learned at generations t and $t+1$.
2. *Ndensity*: Connection density of the network, i.e. the number of connections present in the network out of all possible ($n^2 - n$), where n is the number of vertices.
3. *indegree*: Indegree of a vertex.
4. *outdegree*: Outdegree of a vertex.
5. *betw. cent.*: Edge betweenness centrality.
6. *pair dist.*: For a vertex, average distance to the other vertices. Disconnected vertices are assigned a very high, unattainable, distance value.
7. *reachability*: For a vertex, average reachability to the other vertices. The reachability value between vertices i and j is 1 if i is reachable from j, 0 otherwise.

8. *clust. coef.*: The clustering coefficient of a vertex.

9. *shortcut prob.*: The shortcut probability is the fraction of shortcuts in the network.[30]

10. *n. motifs, $Z = 3$*: Motif frequency for all motifs of size $Z = 3$.

11. *n. motifs, $Z = 4$*: Motif frequency for all motifs of size $Z = 4$.

12. *max. modularity*: The maximum modularity gives a modularity value corresponding to a network module decomposition computed with the Louvain modularity algorithm[28] and the Newman's spectral optimization method,[27] generalized to directed networks.*

13. *vert. participation coef.*: The vertex participation coefficient[31] defines how well distributed the arcs of a node are among different modules.

In the previous list, network measures 3, 4, 6, 7, and 8 are computed as the average of the local measures computed for each vertex. Similarly, network measure 5 is the average of the measures computed for each edge. The other measures correspond to global values.

2.4. Network approach to the modelling of evolutionary algorithms: a priori, online and a posteriori

Algorithm 1 describes the main steps of our general proposal for the analysis of EAs. Step 1 corresponds to a general collection of information or the induction of networks from the outputs of the EAs. Step 2 computes the relevant measures for each particular case. In Steps 3 and 4 the information extracted could be used with different purposes. Variations of this general scheme are tested in the section of experiments.

Algorithm 1: **Network approach for modelling**

1. Create the network from the available information or extract them from the PGMs.
2. Compute a set of relevant measures for each network.
3. Map the network measures to the problem characteristics using machine learning.
4. Apply the machine learning algorithms for classification and inference.

In Figure 3 we illustrate a possible application of Algorithm 1. It shows a schematic representation of an EDA for an optimization problem of four variables. Five generations are represented with different colors. In Figure 3, the populations selected at different generations are mapped to Bayesian networks learned from them where the nodes correspond to the variables of the optimization problem (Step 1 in Algorithm 1). Similarly, each Bayesian network structure is mapped to a vector of network descriptors (Step 2 in Algorithm 1). After the network measures have been computed, a classifier that maps the network measures for the characterization of the problem (Class) is learned (Step 3, Algorithm 1). Finally, network measures extracted from an unknown class problem are passed to the previous classifier. The model then uses these measurements to predict the class of that new problem (Step 4, Algorithm 1).

Machine learning algorithms used in the context of EAs should exhibit a good balance between the computational time spent to learn them and their classification accuracy. The exact balance between the time complexity and the accuracy will depend on the particular task being solved within the EA. In the experiments presented in this paper we have selected machine learning algorithms that are easy to compute and interpret, and not overly complex.

Structural modelling can be done in EAs at different stages of the algorithm. *A priori* modelling is done when some knowledge about the problem structure is available previous to the optimization process. In contrast, *A posteriori* modelling pursues the construction of networks learned after the search

*Louvain modularity algorithm was available in the C++ implementation only. Newman's spectral optimization algorithm was available only in Matlab.

is finished. They can correspond to the structural component of the PGM or constructed with other procedure using the information generated by the EA during the evolution. Notice that it is possible to construct more than one class of *a posteriori* probabilistic graphical model of the data, allowing to compare their different associated networks.

In EDAs, the network contained in the probabilistic model is learned from data during the optimization process, i.e. *online*. In this case, the main goal is to achieve an accurate description of the regularities (e.g. relationships between the variables) contained in the selected solutions and translated to probabilistic (in)dependences. The main use given to the network during online modelling is the sampling of new solutions.

While *online learning* is more suited to EDAs, *a posteriori* learning is able to induce probabilistic models of the selected populations produced by a GA. Regarding *a priori* modelling, if the structure of the problem is previously known, we could use the corresponding networks to evaluate to which extent the crossover operator respects the original problem interactions. However, we will focus the experiments on the *online* and *a posteriori* stages in which the considered networks are derived from the graphical structure of PGMs, rather than using expert knowledge, unavailable most of the times.

3. Experiments

The objective of our experiments is to show how the use of structural modelling within EAs expands the range of application of these algorithms and can be used to improve their efficiency. We take three optimization problems from disparate fields to illustrate different scenarios in which structural modelling can be applied, showing the gains that can be achieved by its application. We devote one section to each problem.

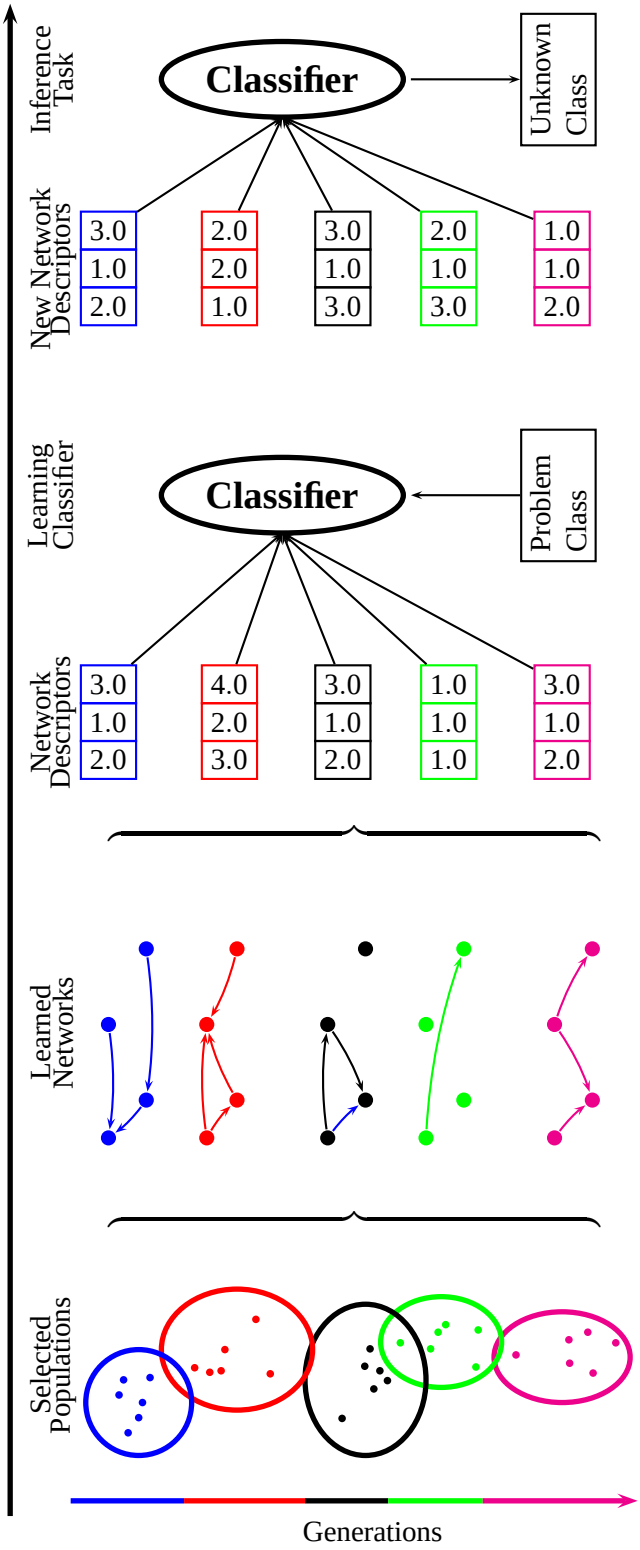

Figure 3: Modeling stages in the analysis of EAs.

3.1. NK fitness landscape

The *NK fitness landscape model* is a parameterized model of a fitness landscape that allows to explore the way in which the neighborhood structure and the strength of interactions between neighboring variables determine the ruggedness of the landscape. For given parameters, the problem consists of finding the global maximum of the function.

An NK fitness landscape[32] is defined by the following components:

- Number of variables n.
- Number of neighbors per variable, k.
- A set of k neighbors $\Pi(X_i)$ for X_i, $i \in \{1,\ldots,n\}$.
- A subfunction f_i defining a real value for each combination of values of X_i and $\Pi(Xi)$, $i \in \{1,\ldots,n\}$.

The objective function f_{nk} to maximize is defined as:

$$f_{nk}(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i, \Pi(x_i)). \qquad (3)$$

The complexity of the NK fitness landscape problem depends on all its components. For $k > 1$ it is NP-hard. This problem is particularly suitable to investigate the use of network measures since it has been extensively analyzed to investigate EAs and other heuristic algorithms.[33]

3.1.1. Instance generation

To generate our benchmark, we used an initial dataset of $9,000$ random instances as described in.[33] There are $1,000$ for every possible combination of $n \in \{20,28,34\}$ and $k \in \{4,5,6\}$. Instance generation was addressed as an optimization problem defined in the space of instances, i.e., our goal in this case is to find instances that maximizes a criterion of easiness (respectively hardness). The function to optimize was the number of times the estimation of Bayesian networks algorithm (EBNA)[34] found the optimum in 30 runs. Notice that in this context EBNA is used as auxiliar algorithm to compute the fitness $g(I)$ of instance I. An instance that maximizes this function, i.e. for which EBNA converged in the 30 runs ($g(I) = 30$), is labeled as an easy instance. Similarly, when the number of successful runs reached by EBNA is minimized ($g(I) = 0$), the corresponding instance is labeled as hard. The easiness and hardness of the instances generated in this way are related to the characteristics of the algorithm used to evolve them. Starting from each of the $9,000$ instances we generated two additional instances, one easy and one hard instance. The final benchmark comprised $18,000$ instances, including only those easy and hard instances.

To search in the space of instances, we used a greedy algorithm that starts from the original random instance and randomly modifies its neighborhood structure. If the optimization function is improved ($g(I') > g(I)$), then the new instance I' is accepted. The maximum number of moves allowed to the greedy algorithm was set to 50. Notice that every time a new instance is generated we need to run a branch-and-bound algorithm to compute the new optimum of the NK fitness landscape function. This was necessary in order to identify the number of successful runs by EBNA.

For the NK fitness landscape we apply *online* structural modelling of the structures generated by an EDA. Two different issues are investigated:

- The use of network measures to predict the class of an instance.
- The evaluation of the sensitivity of the prediction methods to the EA parameters.

3.1.2. Prediction of the class instance

For each of the $18,000$ instances we run EBNA using three different population sizes ($N = n, 2n, 4n$). Ten different runs of the algorithm were conducted for each instance. The total number of runs was $18,000 \times 3 \times 10 = 540,000$. All experiments were computed in a cluster of over 240 computers and we used C++ implementations of the NK fitness model,[33] EBNA,[34] and of the brain connectivity toolbox.[35] The networks produced by the algorithms in the first three generations were stored and from each network the corresponding network measures described in Section 2.4 were computed. Network measures that corresponds to each ten runs of an

EBNA configuration for the same instance were averaged. These were the features used for our classification experiments.

In the NK landscape problem, the problem classes can be defined by fixing the parameters n and k. For a fixed n and with the objective of restricting the analysis to a subset of instances with similar problem difficulty (easy or hard), we group instances into three classes according to the number of neighbors k per variable:

- Class 1: $k = 4$
- Class 2: $k = 5$
- Class 3: $k = 6$

For a fixed n, as the number of neighbors per variable (k) grows, also the difficulty of the problem for an optimization algorithm grows. Empirical evidence showing this effect in the behavior of different EAs is presented in.[33]

To solve the classification task we used regularized multi-logit regression which is an intuitive and relatively simple classifier.[36] It is used to address the general case where the response variable C can have m possible values, i.e. the cardinality of C is $m \geqslant 2$. In this case, the multi-logit model is expressed as:

$$log\frac{Pr(C=l|\mathbf{y})}{Pr(C=m|\mathbf{y})} = \beta_{0l} + \mathbf{y}^T\beta_{\mathbf{l}}, l = 1,\ldots,m-1 \quad (4)$$

where β_{0l} and $\beta_{\mathbf{l}}$ are the parameters of the linear model for class l, and $\mathbf{y}$ is a p-vector of predictor variables. In our application, the predictor variables correspond to the network measures.
Following,[37,38]

$$Pr(C=l|\mathbf{y}) = \frac{e^{\beta_{0l}+\mathbf{y}^T\beta_l}}{\sum_{j=1}^{m} e^{\beta_{0j}+\mathbf{y}^T\beta_j}} \quad (5)$$

The model is fitted using the regularized maximum multi-logit likelihood by means of the elastic net approach.[39] This is an algorithm applied in different domains, that allows to combine the lasso and ridge regularization and for which an efficient implementation was available. We used elastic net regularized multi-logistic regression[37] with $\alpha = 0.9$ and 5-fold-cross-validation to compute the accuracy at predicting the problem class.

Table 1 presents the accuracy results for each combination of the factors considered. In the table, $gens = i$ refers to a situation where only networks up to the ith generation were used to compute the network measures. For $gens = 1$, only networks from the first generation were considered in the analysis. This way we evaluate the influence that limiting the amount of information passed to the classifier has in the accuracy results. Similarly, considering different population sizes $N \in \{n, 2n, 4n\}$, we evaluate to what extent a higher population size has an effect in the amount of structural information captured by the networks and in the associated network measures.

An analysis of the results shown in Table 1 reveals that it is possible to accurately extract the problem class by using the network measures as features. Even when a very small population size is used ($N = n$) and one single network is used to extract the network measures ($gens = 1$), the classification accuracy can be twice the accuracy of a random predictor (0.33). As the population size and the number of networks considered for the analysis grow the classification accuracy also increases reaching 0.99 when $n = 34$, $N = 4n$, and networks from the three first generations are used to extract the features. Also remarkably, accuracy improves with the number of variables.

3.2. HP protein model

The *HP functional protein problem* consists of finding a configuration of a simplified protein model that minimizes an energy representing the interaction between hydrophobic (H) and polar (P) residues. The HP simplified protein model[40] is used in bioinformatics to investigate protein folding. In the HP model, a protein is considered a sequence of hydrophobic (H) and hydrophilic or polar (P) residues which are located in regular lattice models forming self-avoided paths. Figure 4 shows the graphical representation of one possible configuration for the sequence $HHHPHPPPPPH$.

Table 1: Estimated classification accuracies for the NK fitness landscape problem.

type		$n=20$			$n=28$			$n=34$		
	$N/gens$	1	2	3	1	2	3	1	2	3
easy	n	0.69	0.71	0.73	0.59	0.76	0.80	0.84	0.78	0.82
easy	$2n$	0.66	0.64	0.79	0.79	0.83	0.84	0.78	0.86	0.87
easy	$4n$	0.72	0.68	0.87	0.71	0.95	0.95	0.98	0.98	0.99
hard	n	0.64	0.68	0.69	0.86	0.75	0.79	0.63	0.79	0.82
hard	$2n$	0.63	0.72	0.77	0.83	0.82	0.85	0.81	0.86	0.87
hard	$4n$	0.62	0.75	0.85	0.95	0.95	0.93	0.98	0.98	0.98

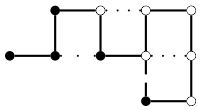

Figure 4: One possible configuration of sequence $HHHPHPPPPPH$ in the HP functional model. Hydrophobic proteins are represented by black beads and polar proteins, by white beads. There is one HH (represented by a dotted line with wide spaces), one HP (represented by a dashed line) and two PP (represented by dotted lines) contacts.

Interactions between neighbor residues (adjacent in the lattice, but not connected in the sequence) contribute to the total energy of the HP lattice configuration. The energy values associated with the functional HP model[41] contain both attractive $\varepsilon_{HH} = -2$ and repulsive interactions ($\varepsilon_{PP} = 1$, $\varepsilon_{HP} = 1$, and $\varepsilon_{PH} = 1$). The HP problem consists of finding the solution (HP chain topological configuration) that minimizes the total energy. The energy that the functional model protein associates with the configuration shown in Figure 4 is 1 because there is one HH interaction, one HP interaction, and two PP interactions.

An HP protein configuration can be represented as a walk in the lattice (sequence of moves). In the sequence of moves, the two initial residues are located adjacent in the lattice. Each other residue is located to the left, to the right, or forming a line with the previous two residues. For a given HP sequence and lattice, X_i will represent the relative move of residue i in relation to the previous two residues. Taking as a reference the location of the previous two residues in the lattice, X_i takes values in $\{0,1,2\}$. With respect to the location of the previous two residues, $X_i = 0$ means that residue i is located to the left. Similarly $X_i = 1$ and $X_i = 2$ respectively mean that residue i will be located in line with the previous two residues and to their right. Values for X_1 and X_2 are meaningless and they are arbitrarily set to 0. This codification is called relative encoding.[42] The representation of the configuration in Figure 4 is thus $\mathbf{x} = (0,0,0,2,2,0,0,2,2,1,0)$.

Relative encoding can also represent illegal or unfeasible solutions (e.g., $\mathbf{x} = (0,0,0,0,0,0,0,0,0,0,0)$) which correspond to self-intercepting protein sequences. To address this situation, in the evaluation phase of the EA we use repairing algorithms[1,43] to transform unfeasible solutions into feasible ones. Protein folds corresponding to proteins from the same family usually share common structural patterns. We expect that two similar HP sequences will have similar optimal lattice configurations. This fact explains the choice of this problem for the experiments. Therefore, we again propose the use of an *online modelling* problem, with the following goals:

1. Can we predict whether EBNA has converged to the optimum value without knowing which the value of the optimum actually is?
2. Can we predict the number of local optima for a given HP protein instance?
3. Given a predefined similarity measure between instances, can we predict the most similar and most different characterized HP instance with respect to a given uncharacterized instance?

We assume that prediction is done based on the networks learned from previous, characterized problems, and the networks obtained from the current,

uncharacterized problem. Also, notice that the questions stated above address three distinct types of information about the problems: 1) Information about the algorithm behavior; 2) Information about the problem characteristics; 3) Information about the similarity between the problems.

3.2.1. *Instance selection and data exploratory analysis*

In this example, our final goal is to be able to predict problem features from the network measures derived from the probabilistic models learned by the EDAs. We assume that we have collected information from the structures learned in the optimization of a group of *characterized* problems and we also have the structures generated during the optimization of a given *uncharacterized* problem. The characterization is given as a description of some problem attributes or features. What we want to obtain is the characterization of the new, uncharacterized, problem.

We will start from a dataset of characterized problems and use a subset of them to characterize the other problems as in a classical supervised classification problem. Classification accuracy is used as a measure of the informativeness of the descriptors used. It also gives a measure of the EDA ability to produce information that can be used to characterize similar problems. We do not ignore the fact that the classification accuracy will also depend on the type of classifier used. We acknowledge that better classifiers could improve the results presented here. However, the focus of our investigation is to assess the usability of the descriptors employed.

We use a dataset of 611 functional HP proteins corresponding to different sequences of 23 residues. This dataset is a subset of an original database of $17,681$ sequences introduced in.[42] These instances have a suitable characteristic: We know their optimal value, which is reached at a single configuration (disregarding symmetric representations). In addition, we know the closest suboptimal value and the number of configurations where this suboptimal value is reached. This information is used as a characterization of the problem.

Figure 5 shows a histogram of the optimal values for the 611 instances. It can be seen that their optimal values lie between -26 and -8 which is a wide range of values for instances that have the same number of variables. Similarly, there are important differences in the distribution of the number of suboptima for all instances in the dataset (data not shown). 374 instances have a number of suboptima between 1 and 4 and the other 237 instances have a number of suboptima in between 193 and $2,532$.

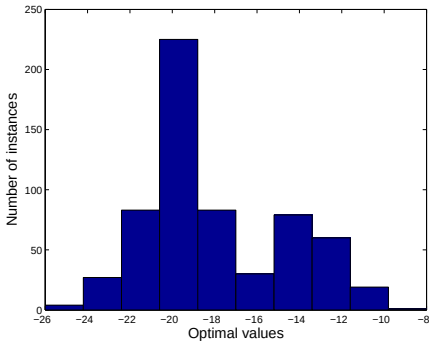

Figure 5: Histogram of the optimal values for the 611 instances.

To evaluate the EDA behavior and collect the networks, 30 independent runs of EBNA were executed for each HP protein instance. We used the MATEDA-2.0 software[44] implementation of EBNA. The learning and sampling steps of the Bayesian networks included in this program are implemented using the Matlab Bayes-Net (BNT) toolbox.[45] The scoring metric used was the BIC with uniform priors, and each node was allowed a maximum number of five parents. Truncation selection was used with a parameter of truncation $T = 0.5$. The population size was $N = 500$ and the termination criterion was to reach the maximum number of generations ($gens = 50$).

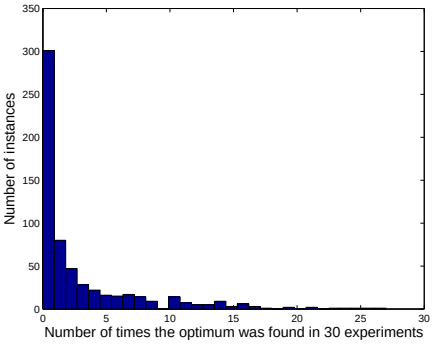

Figure 6: Histogram of the number of times the optimum was found in the 30 experiments for the 611 instances.

For each instance, we computed how many times the optimum was found in the 30 experiments, the average generation at which it was found and the average fitness of the best solutions found in all runs. Figure 6 shows a histogram of the number of times the optimum was found in the 30 experiments for the 611 instances. For 310 out of the 611 problems the optimum was found at least once. For 80 instances it was found only once and for 62 it was found 10 or more times. There were also differences in the average generation at which the optimum was found for the successful runs of the algorithm (data not shown). Most of the times the optimum is found, on average, between generations 10 and 15. There are important differences between the behavior of the EDA for the instances. In general, most problems are hard for the EDA since the success rate of the algorithm is low. However, some problems, in which the EDA finds no optimum in any run, are clearly even harder to solve.

From the DAG of each Bayesian network learned in each generation, we computed the network measures described in Section 2.4.

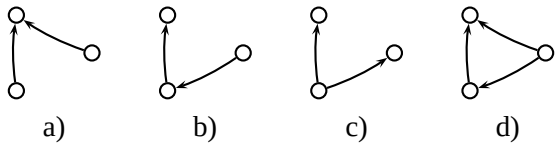

Figure 7: All motifs ($Z = 3$) which appear in the DAGs learned by EBNA.

3.2.2. Prediction of convergence and number of local optima

The first problems considered are the determination of the algorithm convergence and the number of suboptima of the problem. For these two classification tasks, we specify two classes. In the first case, classes are: 1A) Instances for which EBNA did not converge to the optimum in any of the 30 experiments. 1B) The other instances. For the second classification problem, classes are: 2A) Instances with 4 or fewer suboptima. 2B) Instances with 193 or more suboptima. To get some clues about possible characteristic patterns associated to each of the classes, we computed and analyzed the average network measures from networks in each of the classes.

Figure 8 shows the motif frequencies for problems with a successful rate 0 (Class 1A) and with a successful rate higher than 0 (Class 1B). In addition, the charts display information for a subset of instances of Class 1B. This subset is comprised by instances where the EDA converged 9 or more times out of the 30 experiments. An initial observation is that the frequencies of all motif classes get higher for instances that are easier to solve by the EDA. A similar pattern was observed for the problem of classifying the number of suboptima of the instance (results not shown). In this case, instances with a lower number of suboptima produce networks with a higher frequency of all types of motifs.

To determine good predictors of the problem characteristics, we use a multivariate Gaussian classifier[46] which is a simple classifier that only requires to learn two multivariate Gaussian distributions. Additionally, this classifier allows to consider potential interactions between the features that can be relevant for the classification process. The conditional

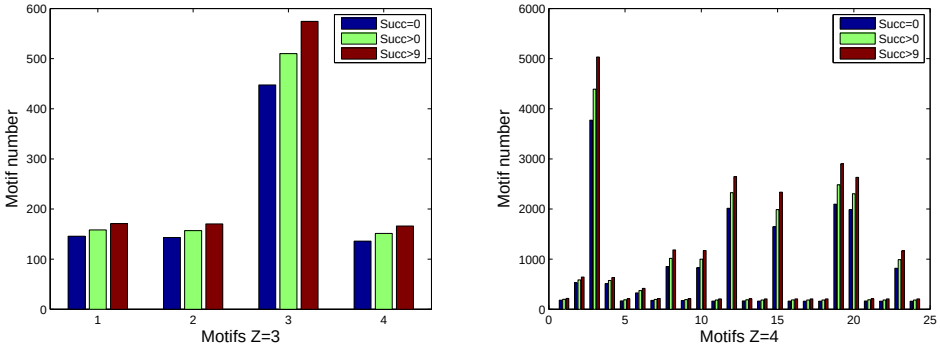

Figure 8: Motif frequencies computed from the networks of instances in which EBNA respectively has successful rate 0, higher than 0 and equal or higher than 9. Left) Motif frequencies for all motifs ($Z = 3$) shown in Figure 7. Right) Motif frequencies for all motifs ($Z = 4$) which appear in the networks learned by EBNA.

density of a solution given the class value A_i is computed as

$$p(\mathbf{z}|A_i) = (2\pi)^{-\frac{n_c}{2}} |\Sigma_{A_i}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{z}-\mu_{A_i})^t {\Sigma_{A_i}}^{-1} (\mathbf{z}-\mu_{A_i})} \tag{6}$$

where $Z_i \in \{X_1, \ldots, X_n\}$, i.e. $\mathbf{Z}$ is a subset of $n_c = |\mathbf{Z}|$ components (or features of $\mathbf{X}$). A_i denotes a class, and μ_{A_i} and Σ_{A_i} are the parameters of a multivariate Gaussian distribution estimated from the points in class A_i. In the simplest case $n_c = 1$, i.e. only one variable of the problem is used as a predictor. In this case, Equation (6) only involves univariate Gaussian distributions.

For a given set of features, we estimated the classifier accuracy using k-fold cross-validation with $k = 5$. The parameters of the multivariate Gaussians were learned using maximum likelihood estimation. In the classification step, we used that $p(A_i|\mathbf{z}) \propto p(\mathbf{z}, A_i) = p(\mathbf{z}|A_i)p(A_i)$ and assumed all classes A_i are equiprobable. Therefore the assigned class was the one with the highest $p(\mathbf{z}|A_i)$. The k-fold cross-validation procedure was repeated 50 times and from these experiments we computed the mean and standard deviation of the classifier accuracy estimator.

For the first two classification problems, finding the optimum and estimating the number of suboptima, the predicted accuracy given by each of the features were independently computed. These results are shown in Table 2. For the sets of network motifs ($Z = 3$ and $Z = 4$), the table only includes the accuracy corresponding to the network motif with the highest accuracy. The best classification results achieved by single features are highlighted in bold. It can be seen that the best accuracy is achieved by the betweenness centrality in the first problem, and by the clustering coefficient in the second problem. Accuracies are higher for the second problem than for the first. It seems easier here to predict whether the problem has few or many suboptima than determining if the algorithm has converged to the optimum.

In order to improve the classification accuracy, we consider interactions between the predictors. In this case, we search for a set of features that maximizes the classification accuracy. This feature subset selection problem, with 39 variables, is addressed using a tree-based EDA as implemented with MATEDA.[44] EDAs have shown to be a good alternative for feature subset selection problems.[47] Only one run of Tree-EDA was used to compute the best set of features. Since Tree-EDA is a stochastic algorithm, we expect that more runs would improve the best solution found. Therefore subsets of features with a better accuracy are likely to exist for this problem. The accuracies obtained with the best combination of features are shown in the last row of Table 2. For both problems, improvements over the best single classifiers were achieved. The clas-

Table 2: Classification accuracy and standard deviation of each single predicting feature and best combination for the EDA convergence to the optimum and the prediction of the number of suboptima.

		Convergence		Suboptima	
feature	*name*	*acc.*	*std.dev.*	*acc.*	*std.dev.*
1	dagdif	0.6023	0.0027	0.7601	0.0020
2	Ndensity	0.6635	0.0022	0.8841	0.0014
3	indegree	0.6637	0.0025	0.8838	0.0014
4	outdegree	0.6621	0.0031	0.8842	0.0018
5	betw. cent.	**0.6789**	0.0025	0.7323	0.0025
6	pair dist.	0.6151	0.0023	0.8593	0.0018
7	reachability	0.6137	0.0020	0.8581	0.0014
8	clust. coef	0.6597	0.0026	**0.8901**	0.0017
9	shortcut prob.	0.6097	0.0043	0.6065	0.0068
10 : 13	n. motifs, $Z = 3$	0.6761	0.0025	0.8796	0.0024
14 : 37	n. motifs, $Z = 4$	0.6783	0.0022	0.8772	0.0016
38	max. modularity	0.6748	0.0034	0.7761	0.0020
39	vert. part. coeff.	0.6376	0.0032	0.7875	0.0031
Best combination		0.7084	0.0065	0.9132	0.0035

sification accuracies of these sets of predictors were respectively above 0.70 and 0.91. Recall that a random predictor that always picks the majority class would respectively achieve accuracies of 0.5075 and 0.6121. This fact shows that all single predictors improve the accuracy of the random predictor, and that a good combination of features substantially outperforms the results of the random predictor. The sets containing the best predictors were $\{2,5,6,12,14,15,26,27,28,33,37,39\}$ for the first problem, and $\{8,12,20,22,25,27,29,33,35,36,39\}$ for the second.

We highlight that although the two classification tasks were conducted independently, we could think of simultaneously classifying the EDA convergence and the number of suboptima of the problem. This could be a suitable alternative considering the interactions between the two problems.

We have empirically shown that the information learned during the optimization of past problems for which some particular features are known can be employed to predict features of new problems for which we do not have the same class of information.

3.2.3. Prediction of most similar and most different instances

In the next step, we intend to use the structures to distinguish, in a data set of characterized problems, similar from dissimilar problems. Two different measures of similarity between instances are used: 1) The sequence similarity, which is the number of common residues in the two instances, and, 2) The fitness correlation between two fitness functions, computed from a random sample taken from the solutions space.

Figure 9 shows the sequence similarity against the fitness correlation for all pairs of instances. Recall that each instance defines a different fitness landscape. We use $10,000$ points to compute the correlation between fitness functions. It can be observed that there is a strong relationship between both similarity measures. However, as expected, pairs of instances with equal sequence similarity can have very different fitness correlation.

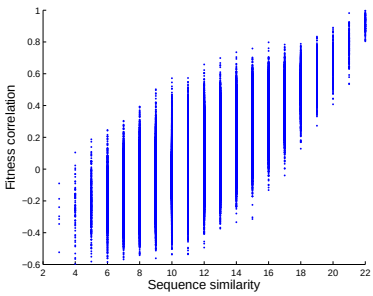

Figure 9: Sequence similarity against the fitness correlation for all pairs of instances.

To construct the database of cases, we identified,

for each instance, the most similar and most different instance. We used two different similarity measures for selecting pairs of most similar and most dissimilar instances. After the identification process concluded, there were two dataset, one for each similarity measure. We then were interested in detecting those instances with highest values of similarity and dissimilarity at the same time. To detect those pairs of instances, we took the difference $\mathbf{y} = \mathbf{x}^i - \mathbf{x}^j$ as features, i.e. the difference between the network descriptors corresponding to instances i and j.

The values of $\mathbf{y}$ were categorized as very similar instances as class label 1 or very dissimilar instances as class label 0. This supervised set of values constitued the classification problem to deal with. Since an instance may have more than one most similar or dissimilar matches. we chose an arbitrary instance among those being closest. As a result, there was a database of $611 \times 2 = 1,222$ points for each similarity measure, equally distributed between the two classes

We used the same type of classifier and experimental protocols than in the previous classification experiments. Results are shown in Table 3. In this prediction problem, the single classifiers have more similar performance. The best individual predictor, when sequence similarity measure is used, is the reachability measure (0.6558). When the fitness correlation measure is used, the best predictor is the outdegree (0.6683). In general, single predictors do not give a high accuracy. However, when interactions between features are considered, the accuracy in the prediction is much higher for both problems (an increase of 0.07 for the first problem and of 0.13 for the second). The main conclusion from the experiment is that information extracted from the networks can be used to distinguish similar and dissimilar pairs of instances, particularly if interactions between the network measures are considered.

3.3. *Neuroproteomics multi-objective feature selection problem*

The *neuroproteomics multi-objective feature selection problem* consists of selecting a minimal subset of features that gives the highest classification accuracy in the diagnosis of Alzheimer's disease (AD). For this multi-objective feature selection problem, we have a dataset of cases with a set of attributes and an observed class label. The minimal subset of features will be used as input to a predefined classifier, giving a correct classification rate as high as possible. There are two objectives to be fulfilled: 1) Maximize the correct classification rate, and 2) Minimize the number of selected features. Therefore, the problem is to obtain a good approximation of the Pareto set of solutions.

The database used for this experimentation comes from the work presented in.[48] A panel of biomarkers is investigated for the clinical diagnosis of AD. The work makes use of a proteomic dataset of 120 signaling proteins measured in individuals with proved AD diagnosis and in various control samples. We use the training set of this work as input to our heuristic search. It comprises 120 features on a dichotomic supervised problem with 43 AD samples and 40 healthy controls.

Each solution is represented using a binary vector where $X_i = 1$ means that the corresponding feature is included in the classifier. Two different classification strategies, that can be interpreted as two different problem formulations with two fitness functions, are used:

- k-nearest neighbor - The k-NN algorithm[49] performs the classification task in terms of similarity: unlabeled examples are classified based on their distance to the examples in the training set. k-NN has no explicit classification model and, hence, there is no learning stage. It finds the k closest examples in the data and assigns the class that most frequently appears within such k-subset. For our experiments, k-NN was computed with Euclidean distance and $k = 1$.
- Naïve Bayes - Continuous naïve Bayes[50] is based upon the Bayes formulation of conditional dependencies. The simplest structure is based on the assumption of conditional independence between the predictor variables given the class variable, that is, the naïve Bayes structure. Then, the model parameters are estimated with a factorization based on the normal distribution assumption for each variable given each value of the class variable.

Table 3: Classification accuracy and standard deviation of each single predicting feature and best combination for the prediction of the most similar and dissimilar pairs of instances.

		Seq. similarity		Fitness correlation	
feature	*name*	*acc.*	*std.dev.*	*acc.*	*std.dev.*
1	dagdif	0.5639	0.0039	0.5608	0.0052
2	Ndensity	0.6516	0.0019	0.6207	0.0035
3	indegree	0.6516	0.0021	0.6211	0.0036
4	outdegree	0.6514	0.0024	**0.6683**	0.0017
5	betw. cen.	0.6126	0.0031	0.6113	0.0033
6	pair dist.	0.6554	0.0021	0.6100	0.0037
7	reachability	**0.6558**	0.0023	0.6457	0.0026
8	clust. coeff.	0.6495	0.0026	0.6208	0.0030
9	shortcut prob.	0.5959	0.0025	0.6097	0.0039
10 : 13	n. motifs, $Z = 3$	0.6469	0.0026	0.6103	0.0030
14 : 37	n. motifs, $Z = 4$	0.6435	0.0024	0.6193	0.0027
38	max. modularity	0.6164	0.0028	0.6164	0.0030
39	vert. part. coeff.	0.6056	0.0027	0.5822	0.0034
Best combination		0.7271	0.0041	0.8143	0.0043

To tackle this problem, we made use of the *a posteriori* modelling to a multi-objective optimization problem. The main goal of our experiment is to show that *a posteriori* modelling can provide useful information when applied to evolutionary algorithms other than EDAs. To this end, we analyzed the selected populations generated by a GA (GA-KNN and GA-NB), constructing networks from these populations and comparing them to those generated by EBNA (EDA-KNN and EDA-NB). Our analysis intends to show the convenience of analyzing the network theoretic measures extracted from the evolution of EAs. In particular to this application, we want to find answers to the following questions:

1. Can network theoretic measures be used for characterizing the effect of different EA variation operators and problem formulations?

2. From a given time in the evolution, is it possible to predict dynamics of the algorithm in subsequent generations?

3.3.1. Generation of the data

We applied a GA with single-point crossover and also an EBNA to find a Pareto set approximation of the bi-objective function under consideration. GA and EBNA were modified to address the multi-objective problem. The modification is based on the use of Pareto ranking selection. This selection method begins by computing all the Pareto sets in the population, then solutions within each Pareto set are sorted according to the average rank of the fitness function for all the objectives. Finally, the selected population is taken from the sorted population applying truncation selection. We have successfully applied this selection method in previous applications to multi-objective problems.[5,51]

The idea is to compare the networks produced during the evolution of the GA with those generated by EBNA. The *k*-NN and naïve Bayes classifiers were used as two alternative feature subset selection (FSS) problem formulations. Each classifier can assign a different accuracy to the same set of features. However, there is a high correlation between the fitness given by both classifiers. For each fitness function, we executed 16 runs of the two EAs. The GA used one-point crossover and bit-flip mutation with crossover and mutation probabilities respectively equal to $P_c = 1.0$ and $P_m = 0.01$. The population size used for the two algorithms was $N = 250$. Truncation selection with $T = 0.5$ was used. The stopping condition was to reach a number of generations $gens = 50$.

In the case of the GA, the selected sets of each generation were stored and *a posteriori* modelling was done by learning the Bayesian networks from the selected populations. For EBNA, the networks generated during the evolution were used for the

analysis. As a result, there were $16 \times 50 = 800$ Bayesian networks for each fitness function and each algorithm. The set of networks measures described in Section 2.4 were computed for each of the networks.

3.3.2. *Influence of the variation operators and the fitness function*

The most straightforward analysis of the topological characteristics of the networks is the analysis of the frequency matrices. These matrices simply compute how many times each arc appears in all the networks learned by the EAs. The rationale is that the most frequent arcs may correspond to relevant pairwise interactions between the problem variables. Therefore, we start by computing these matrices for each of the fitness functions and algorithm.

Frequency matrices shown in Figure 10 store the frequency of the arcs from the set of 800 DAGs learned for each problem. In the figure, the colorbars indicate the mapping between the arc frequencies and the colors. For the two fitness functions, it can be seen in Figure 10 (a and b) that the GAs produce networks in which arcs corresponding to adjacent nodes have a much higher frequency. This is a known effect of one-point crossover: it tends to respect relationships between variables closer in the representation, but variables distant from each other are more vulnerable to the disruption. The networks learned by EBNA, Figure 10 (c and d), are less prone to disrupt dependencies between distant nodes and less likely to promote dependencies between adjacent variables in the representation. This example clearly shows how *a posteriori* modelling of the GA can serve to detect the effect of the crossover operators in the arousal of dependencies. In cases where the potential bias that the crossover operator induces in the generation of solutions is less evident, frequency matrices may be useful to detect this bias and to modify the choice of the crossover operator accordingly.

Further analysis of Figures 10 reveals that the k-NN classifier (a) induces more dependencies between the variables of the problem than the naïve Bayes classifier (b). This can be deduced from the higher frequency of the learned arcs. In this case, *a posteriori* modelling is useful to reveal differences in the number of problem interactions determined by the different fitness functions. Based on this information, the user could be able to decide whether to employ a problem formulation that induces more or less dependencies between the variables. Notice that the number of dependencies between the variables is usually associated to the complexity of the problem for the EA.

A more detailed characterization of the influence that the variation operators and the fitness function have in the evolution of the algorithms can be obtained by analyzing the network measures. We focus on two alternative approaches to the network measures analysis:

1. Vertex approach: Focuses on the differences between the vertices of the networks. The information learned from all the EA generation networks is condensed to compute the statistics of each vertex. This approach does not focus on differences between generations but on the salient characteristics of the vertices, allowing to identify differences between the roles played by the variables.

2. Generation approach: Focuses on the differences between the networks learned at each generation. Global network and local measures computed at each generation are used. When the network measure is local (e.g. associated to a vertex), it is the average of all local measures computed for the network. Therefore, this approach does not look at differences between the vertices or edges. It focuses on salient characteristics of the generations captured by the network measures.

Figure 11 shows examples of the two approaches using the betweenness centrality local measure. Figure 11 (left) shows the betweenness centrality values for each of the 120 vertices computed from the 800 networks learned for each fitness function. Figure 11 (right) shows the average betweenness centrality at each generation of the algorithms. The betweenness centrality of a network is computed by averaging the betweenness centrality of its 120 nodes. The final average betweenness centrality at generation g

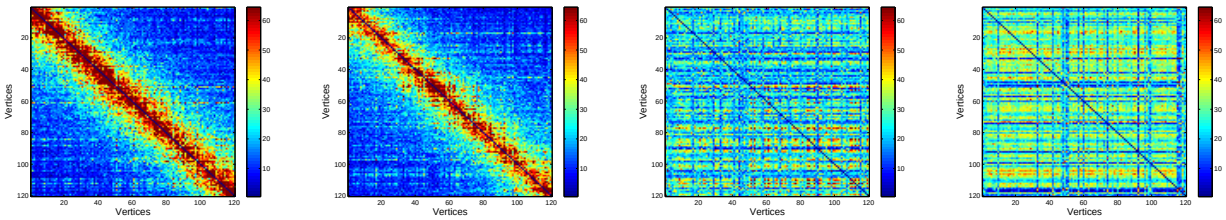

Figure 10: Frequency matrices of the Bayesian networks learned from the GA selected populations using *k*-NN (a) and naïve Bayes (b) fitness functions. Frequency matrices learned by EBNA using *k*-NN (c) and naïve Bayes (d).

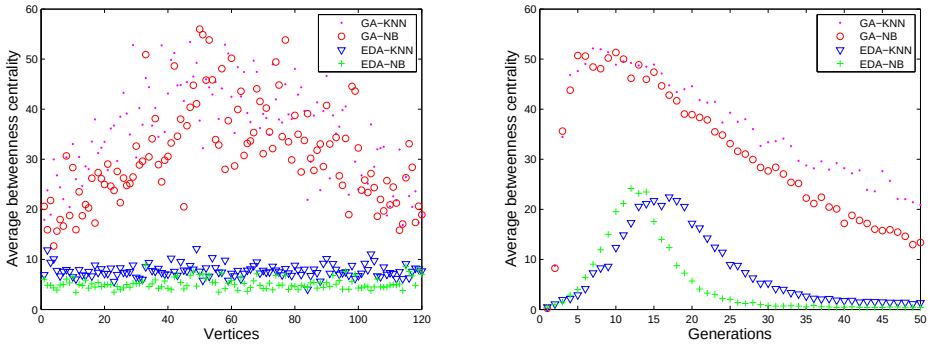

Figure 11: Two possible representations of the betweenness centrality values computed from the networks. Left: Average betweenness centrality values of each vertex computed from all generations. Right: Average betweenness centrality values at each generation computed from all vertices.

is computed as the mean of the network betweenness connectivity of the networks learned at generation g in all the 16 runs. The same procedures are used to compute the average of the other network measures. Global network measures (e.g. modularity coefficient) are only computed for the generation approach.

The analysis of betweenness centrality of the vertices shown in Figure 11 (left) also reveals the effect of the crossover operator in the topologies of the networks. Vertices corresponding to the first and last variables in the representation have a lower betweenness centrality since their potential dependencies to the other variables are more likely to be disrupted. The characteristic bell shape of the betweenness centrality values for the GA-produced networks contrasts with the flat distribution of the centrality values achieved by the EDA networks. The generation approach to the analysis of the network measures shown in Figure 11 (right) reveals clear differences in the behavior of the GA with respect to the EDA. The average betweenness centrality of the networks produced by the GA is always higher than those produced by the EDA. This fact may be related to the different role played by the variables in both variation operators.

The generation approach to the analysis of the network measures also supports a dynamical view of the EA behavior. It is possible to classify or group generations according to the similarities between the computed network measures. In particular, extreme values of the network measures may point to critical periods for the EDA behavior. In Figure 11 (right), the maximal betweenness centrality values for the

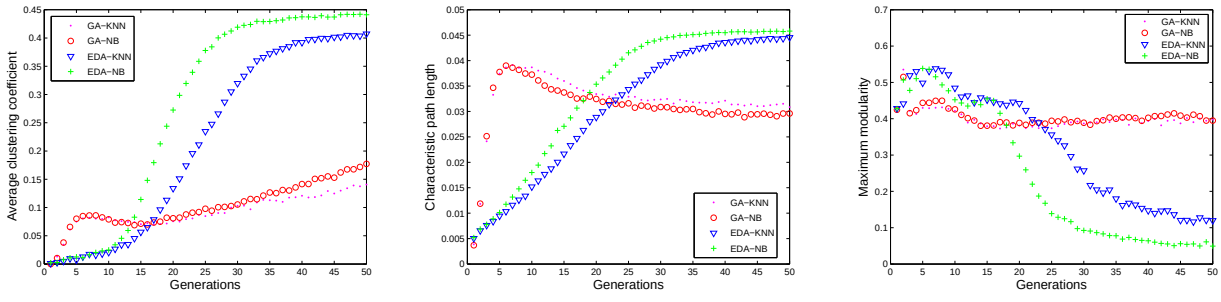

Figure 12: Network measure descriptors of the EA behavior. a) Clustering coefficient b) Path length c) Minimized modularity.

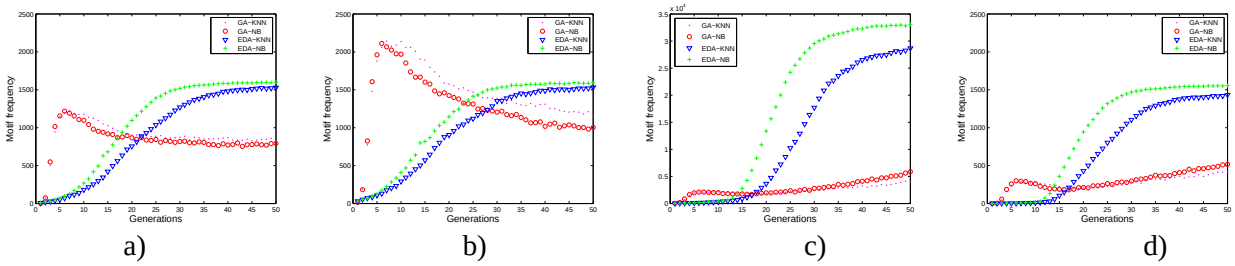

Figure 13: Evolution of the motifs frequencies for all possible motifs of size 3 included in DAGs. The order of the charts corresponds to the motifs shown in Figure 7.

GAs are rapidly reached around generation 5. EDAs take more time to reach the maximal values (around generation 15). As evolution advances the betweenness centrality tends to decrease in the two algorithms. Convergent behavior of the network measures may denote the stagnation of the evolutionary process. More importantly, they could be used as predictors of the future algorithm behavior, as analyzed later on Section 3.3.3.

Three network measures that have a relevant place in the analysis of complex networks are the clustering coefficient, the path length and the modularity. We use these measures to compare the effect of the different problem formulations. Figure 12 shows the values of the three measures in each generation of the EAs and for the two fitness functions. The differences between the networks generated for *k*-NN and naïve Bayes depend for the three measures on the type of algorithm used to optimize the functions. While the network measures of *k*-NN and naïve Bayes are very similar for the GA, there are bigger differences between them when the EDA is used. A similar behavior was observed for other network measures (results not shown). These results seem to indicate that, in general, the translation of the problem differences to the network measures will highly depend on the used variation operator. However, some network measures may be more sensitive to the differences in problem formulation. In addition, the joint use of different network measures can serve to better characterize the differences between problem formulations. One example of this joint use of measures appears in the examination of different motif frequencies.

We computed the network motif frequencies for all motifs ($Z = 3$ and $Z = 4$) present in the DAGs. Figure 13 shows the motif frequencies for the only four motifs ($Z = 3$) which can be found in the DAGs. These motifs are shown in Figure 7. The analysis of Figure 13 reveals that the motif frequencies for the networks produced by the EDAs tend to be higher than the motif frequencies of networks generated by the GAs for three out of the four motifs (Figure 7 a), c) and d)). The exception is the second motif from the left in Figure 13, for which the corresponding frequencies are higher in the case of the EDA.

We have not been able to find a mechanistic explanation for the unequal distribution of motif frequencies between algorithms. Nevertheless, the increase of motif frequencies with the number of generations in GA could be explained by the increase in the network density. In the case of motifs with $Z = 4$ (results not shown), it is possible to identify a wider variety of behaviors for the motifs frequencies. This variability suggests the use of motifs as signatures of the characteristic evolutionary path followed by an EA in the solution of a given problem. These signatures depend on the relationships between the variables captured by the networks and therefore they would allow to compare not only different formulations of the same optimization problem but also single and multi-objective problems from completely different domains.

3.3.3. Prediction of the algorithm behavior

Vertex network theoretic measures can also be used to predict the behavior of the algorithms in future generations. To investigate this use of the network measures, we focus on the evolution of the univariate probability distributions computed from the selected sets. We show that univariate probabilities of the selected set can be used as descriptors of the population homogeneity allowing the implementation of diversity preserving methods.

Let $\widehat{p}_i^j$ be the empirical probability of $X_i = x_i$ at generation j. Let $(y_{i,j}^1, \ldots, y_{i,j}^r)$ the set of the r vertex network measures computed for vertex i from the network learned at generation j. Then we want to find a mapping γ^g such that $\widehat{p}_i^{j+g} = \gamma^g(y_{i,j}^1, \ldots, y_{i,j}^r, \widehat{p}_i^j)$ where g is the number of generations from the current generation, from which the prediction is to be made.

The idea of including the current univariate value $\widehat{p}_i^j$ as a source of information for prediction is because univariate probabilities usually experiment only minor changes from one generation to the few next generations. Therefore, the current univariate probability of the selected population is a good candidate for a predictor variable.

We find γ^g using linear regression which can be computed very fast. The minimum square er-

ror (MSE) between $\widehat{p}_i^{j+g}$ and $\gamma^g(y_{i,j}^1,\ldots,y_{i,j}^r,\widehat{p}_i^j)$ is evaluated for different values of g and the two EAs. In our experiments, we set the number of vertex network measures $r = 8$. The used network measures were: indegree, outdegree, clustering coefficient, betweenness centrality, pair distance, reachability, shortcut probability, and the vertex participation coefficient.

Figure 14 shows the average MSE value, computed from all the variables, for GA-KNN, GA-NB, EDA-KNN, and EDA-NB, when $g \in \{1,5,10,20\}$. It can be seen that the error of the approximation is in general very low and converges to zero in the last generations. As expected, the error gets higher with g.

We further investigated whether the good approximation is exclusively due to the use of $\widehat{p}_i^j$ as a regressor or because of the vertex network measures contribute to the prediction. Figure 15 shows the prediction gain achieved in the estimation of the univariate probabilities by including vertex networks descriptors. To compute the prediction gain, we found the difference between the prediction error given using $\widehat{p}_i^j$ as the only regressor variable, and the prediction given when all vertex network measures are included as regressors. Therefore, in Figure 15, positive values indicate a prediction gain. Not surprisingly, prediction gain is very small when $g = 1$, i.e. prediction is made on the univariate probabilities of the next generation. However, as g increases the contribution of the vertex network measures is more important.

If the charts showing the prediction gains are further inspected, it is possible to see that between generations 15 and 20 GA-NB (Figure 15 b)) there is a negative prediction gain from the use of the network measures. This case demonstrates that not in all the situations the predictions will be improved. There are also marked differences between the algorithms: the prediction gain is clearer for EDA-KNN and EDA-NB (Figure 15 c) and d)) than for the GAs. We applied also regression methods that consider the interactions between the predictors. Using these methods it was possible to further improve the prediction gain (results not shown). However, at some generations the low density of the networks and the similarity between the measures computed from them makes the behavior of the regression methods more unstable. Also due to their simpler complexity, we recommend the use of the linear regression methods.

4. Discussion on the structural findings

We have not found previous references on the structural modelling of GAs from the perspective of network theory. Therefore, the following analysis covers previous work on EDAs that addresses different facets of the models learned by these algorithms. We review some of this work, stressing the differences with the results introduced in this paper. We also include an account of work on meta-learning that is relevant to our research. Note that current work on EDAs is a very productive field[52] and our review is by no means exhaustive.

4.1. Relationship between the problem structure and the structure of dependencies

There are many papers that study and describe the main characteristics of the structural component of the probabilistic models learned by EDAs.[53–56] This description is mainly done in terms of the number of edges or arcs that appear in the model structures. Another popular approach has been the classification of the graph edges in correct or spurious, in accordance with their relationship with the (known) problem structure. This has been done by computing the most frequent edges appearing in the structural component of the probabilistic models in EDAs and analyzing their mapping with the structure of interactions of the problem.[55,57–62]

4.2. Using the models to improve the search in EDAs

In EDAs, the use of structural information has been mainly constrained to bias model building and approximate fitness functions. Two main approaches have been identified to bias model building in EDAs:[63] Impose soft restrictions by biasing the scoring metric to prefer models that closely correspond to the problem structure[4] or impose hard

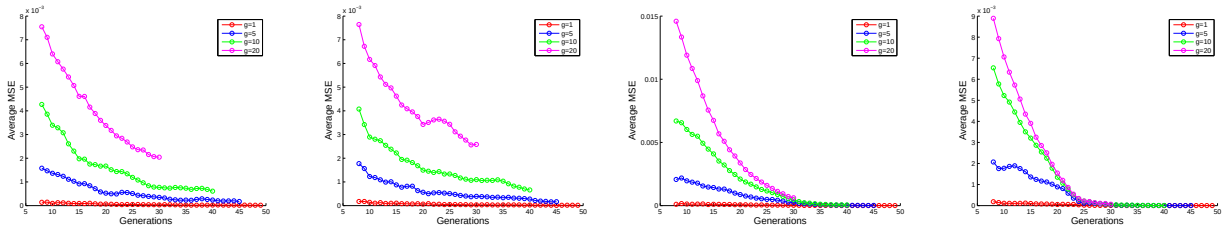

Figure 14: MSE in the estimation of the univariate probabilities. a) GA-KNN, b) GA-NB, c) EDA-KNN, d) EDA-NB.

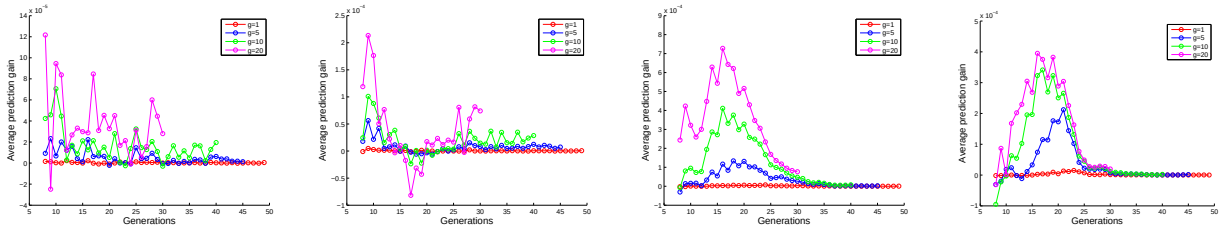

Figure 15: Prediction gain achieved in the estimation of the univariate probabilities by including vertex networks descriptors. a) GA-KNN, b) GA-NB c) EDA-KNN, d) EDA-NB.

restrictions by strictly disallowing some dependencies.[4,64–66] The question of how to extract the structural information to bias the model is reviewed in the next section.

For estimating the fitness function using the models, statistical analysis of the structures, of the type presented in this paper, is not usually made. Instead, the marginal probabilities or the probability associated by the model to a given point are used to estimate the fitness.[67–70]

4.3. Extracting information from the learned models by EDAs

The first work where automatic procedures for extracting and reusing information in the future of the solution of similar problems was recently presented in.[4] Two different approaches were introduced to extract the problem information: a) Computation of the probability coincidence matrix and b) Computation of a distance-based metric between variables from the known problem structure.

In both approaches the extracted information was used to bias model building in problems which are known *a priori* to be similar. The information is not used to infer, predict or characterize other instances. Another difference to our approach is the class of information extracted from the models. The coincidence matrix extracts local information about edge frequencies in the model structures. The distance based metric computes the distance from the original problem structure (e.g. lattice and SAT based interaction graph between the variables). The first approach can be considered as a particular case of *online* structural learning where only a particular network measure is employed. The second approach can be included as one of the possible *a priori* structural modelling strategies. Addressing the question of knowledge exchange in EAs using network measures as problem structural descriptors, together with classifiers as inference techniques, is both a more general and flexible approach.

In parallel to the work presented in this paper, recent works have described a variety of ways for

knowledge extraction and its exploitation in EDAs.[2] presents an example of how structural information extracted from Bayesian networks by simple analysis of edge frequencies can be used to reconstruct networks of metabolites that reflect physiological metabolite interconnections. In,[71] peakbin selection in mass spectrometry data is addressed using an ensemble approach that integrates information from different probabilistic models learned by an EDA. In,[72] structural transfer between instances is implemented combining Bayesian network edge frequencies computed from instances (single nucleotide polymorphisms of 58 human populations) with different sizes and only partially overlapping variables. Extraction of information is also possible by analyzing the parameters in regularized Gaussian models,[73] or models that learn dependencies between variables and objectives.[74] Lastly,[3] shows that a distance-based bias technique can be applied for knowledge exchange between instances that have different sizes.

4.4. Other related work in evolutionary computation

While we have focused on structural modelling, recent work takes profit of all the information contained in the probabilistic model to investigate the efficiency of the EDAs components, and the relationship between the model structure and the quality of the solutions found. This is done by computing different measures associated to the probabilistic models such as the most probable configurations[75] and the correlations between the model probabilities and the fitness function.[57,76] Using measures that contain information about the model structure and parameters can be seen as a possible way to generalize structural modelling. However, this type of measures have not been applied to problem characterization or instance classification within EDAs.

One of the few papers that treats different issues related to EDAs complexity in terms of the topological characteristics of the models learned by EDAs is.[18] By analyzing some topological characteristics of the interaction graphs for random additively decomposable functions, the authors prove that the space complexity of the factorized distribution algorithm and Bayesian network-based algorithms may be exponential in the problem size even if the optimization problem has a very sparse interaction structure. The network descriptors used by[18] are just one case of the informative network measures that could be of application in EDAs.

The identification of network motifs and the computation of their frequencies have been extensively applied to the analysis of biological,[24,77] artificial[24,78] and evolved networks.[51,79] However, we have not found previous references to the systematic analysis of network motif frequencies in Bayesian networks or graphical models in general.

4.5. Meta-Learning

Our work is also related to meta-learning,[80,81] an area where active research is pursued by several research groups. The goal in meta-learning is twofold:[80] 1) To provide automatic and systematic user guidance by mapping a particular task to a model (or combination of models) and 2) To allow the learning mechanism itself to relearn, taking into account previous experience.

There are some main differences between meta-learning approaches and the method we propose to extract relevant problem information from the data generated by the EAs. The primary difference from meta-learning methods is that we do not need explicit structural (a priori) information about the problem. This information can be extracted from the networks learned from the data. In this sense, our work is relevant for the question of identifying features that correlate with the empirical hardness of problem instances. This is a fundamental question that arises in meta-learning and other areas.

5. Conclusions and future work

In this paper we have shown that network measures computed from networks produced from the analysis of EAs can capture problem characteristic information and also support evidence about the behavior of the algorithms. We argue that the use of these measures could serve to devise “intelligent” optimization methods, able to learn from past experience to recognize and solve related problems. In

addition, the application of network measures opens a different perspective for the analysis of problem difficulty and algorithm behavior in EAs. There are a number of directions where further research is worth.

Network measures can be employed to compare different algorithms that produce networks, describing their behavior. Collaborative problem solving between different optimizers (e.g. EAs that use different types of variation operators) can be organized using the information derived from the networks produced by the optimizers for different classes of problems. It could be possible to assign optimization problems based on previous performance of the optimizers for similar classes of problems.

One assumed fact is that the identification of strongly related subsets of variables can be used as a way for problem decomposition. In directed and undirected networks, a set of strongly related variables can be associated to a network module, community, or cluster of vertices. In weighted networks,[82] modules can be considered as sets of related variables whose edges add to a high weight. When applied in the context of EAs, a possibility would be to investigate algorithms available for network community detection[83] as a way to achieve problem decomposition.

In some of the conducted experiments, the analysis of the classifiers learned from the network measures allowed us to capture information about which were the features relevant for classification. This type of information could support additional knowledge about the optimization problem and its relationship with the structural characteristics captured by the network measures.

In previous work,[1,5] we showed that marginal probabilities extracted from Bayesian networks can help to detect valuable partial configurations. The combination of structural and quantitative information from Bayesian networks could help to improve classification accuracy and we envision this topic as worth for future work.

Another possible development is the application of network measures defined for weighted networks using as weights some parameters derived from probability distributions in order to obtain other descriptors of probability distributions.

Acknowledgments

This work has been partially supported by projects TIN2010-20900-C04-04, TIN2010-14931, Consolider Ingenio 2010-CSD2007-00018 and Cajal Blue Brain of the Spanish Ministry of Economy and Competitiveness (MINECO). R. S. is also supported by the Etortek, Saiotek and Research Groups (IT-609-13) programs (Basque Government). R.A. is supported by a Juan de la Cierva postdoctoral fellowship (MINECO).

References

1. R. Santana, P. Larrañaga, and J. A. Lozano. Protein folding in simplified models with estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 12(4):418–438, 2008. http://dx.doi.org/10.1109/TEVC.2007.906095 doi:10.1109/TEVC.2007.906095.
2. H. Franken, A. Seitz, R. Lehmann, H.U. Häring, N. Stefan, and A. Zell. Inferring disease-related metabolite dependencies with a Bayesian optimization algorithm. In *Proceedings of the Conference Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 62–73, Malaga, Spain, 2012. Springer.
3. M. Pelikan and M. Hauschild. Transfer learning, soft distance-based bias, and the hierarchical BOA. MEDAL Report No. 2012004, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), March 2012. Also available as `http://medal-lab.org/files/2012004.pdf`.
4. M. Hauschild, M. Pelikan, K. Sastry, and D. E. Goldberg. Using previous models to bias structural learning in the hierarchical BOA. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2008*, pages 415–422, New York, NY, USA, 2008. ACM. http://dx.doi.org/10.1145/1389095.1389228 doi:10.1145/1389095.1389228.
5. R. Santana, C. Bielza, J. A. Lozano, and P. Larrañaga. Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2009*, pages 445–452, New York, NY, USA, 2009. ACM. http://dx.doi.org/10.1145/1569901.1569963 doi:10.1145/1569901.1569963.

6. L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences (PNAS)*, 97(21):11149–11152, 2000.
7. S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275–1335, 2008.
8. L. A. N. Amaral and J. M. Ottino. Complex networks: Augmenting the framework for the study of complex systems. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38:147–162, 2004.
9. M. Barthélémy and L. A. N. Amaral. Small-world networks: Evidence for a crossover picture. *Physical Review Letters*, 82(15):3180–3183, 1999.
10. S. Boccaletti, V. Latorab, Y. Moreno, M. Chavez, and D.-U Hwanga. Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308, 2006.
11. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
12. J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.
13. P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.
14. J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer, 2006.
15. 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.
16. M. Pelikan, K. Sastry, and E. Cantú-Paz, editors. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Studies in Computational Intelligence. Springer, 2006.
17. B.H. Junker and F. Schreiber. *Analysis of Biological Networks*. Wiley, 2008.
18. Y. Gao and J. C. Culberson. Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143, 2005.
19. D. J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, 1999.
20. D. J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.
21. S. Kintali. Betweenness centrality: Algorithms and lower bounds. *arXiv.org*, arXiv:0809.1906v1 [cs.DS], 2008.
22. U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.
23. M. E. J. Newmann. Mixing patterns in networks. *Physical Review E.*, 67:026126, 2003.
24. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.
25. J. C. Reijneveld, S. C. Ponten, H. W. Berendse, and C. J. Stam. The application of graph theoretical analysis to complex networks in the brain. *Clinical Neurophysiology*, 118(11):2317–2331, 2007.
26. L. Moss. *What Genes Can't Do*. MIT Press, 2001.
27. E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Physical Review Letters*, 100:118703, 2008.
28. V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.
29. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.
30. O. Sporns. *Neuroscience Databases. A Practical Guide*, chapter Graph theory methods for the analysis of neural connectivity patterns, pages 171–186. Kluwer, 2002.
31. R. Guimera and L. A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.
32. S. Kauffman. *Origins of Order*. Oxford University Press, 1993.
33. M. Pelikan. Analysis of estimation of distribution algorithms and genetic algorithms on NK landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2009*, pages 1033–1040. ACM, 2008.
34. R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 151–173, 1999.
35. M. Rubinov, S. A. Knock, C. J. Stam, S. Micheloyannis, A. W. F. Harris, L. M. Williams, and M. Breakspear. Complex network measures of brain connectivity: Uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.
36. D. Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.
37. J. Friedman, T. Hastie, and R. Tibshirani. Regulariza-

tion paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.
38. J. Zhu and T. Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427, 2004.
39. H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320, 2005.
40. K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.
41. J. D. Hirst. The evolutionary landscape of functional model proteins. *Protein Engineering*, 12:721–726, 1999.
42. N. Krasnogor, B. P. Blackburne, E. K. Burke, and J. D. Hirst. Algorithms for protein structure prediction. In *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 769–778. Springer, 2002.
43. C. Cotta. Protein structure prediction using evolutionary algorithms hybridized with backtracking. In J. Mira and J. R. Alvarez, editors, *Artificial Neural Nets Problem Solving Methods*, volume 2687 of *Lecture Notes in Computer Science*, pages 321–328. Springer, 2003.
44. R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armañanzas, and S. Shakya. Mateda-2.0: A MATLAB package for the implementation and analysis of estimation of distribution algorithms. *Journal of Statistical Software*, 35(7):1–30, 2010. Available from: `http://www.jstatsoft.org/v35/i07`.
45. K. Murphy. The BayesNet toolbox for Matlab. *Computer Science and Statistics: Proceedings of Interface*, 33, 2001. Available from: `http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html`.
46. A. K. Jain and W. G. Waller. On the optimal number of features in the classification of multivariate Gaussian data. *Pattern Recognition*, 10(5-6):365–374, 1978.
47. I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence*, 123(1-2):157–184, 2000.
48. S. Ray, M. Britschgi, C. Herbert, Y. Takeda-Uchimura, A. Boxer, K. Blennow, L.F. Friedman, D.R. Galasko, M. Jutel, A. Karydas, et al. Classification and prediction of clinical Alzheimer's diagnosis based on plasma signaling proteins. *Nature Medicine*, 13(11):1359–1362, 2007.
49. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.
50. G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, volume 1, pages 338–345. San Mateo, 1995.
51. R. Santana, C. Bielza, and P. Larrañaga. Optimizing brain networks topologies using multi-objective evolutionary computation. *Neuroinformatics*, 9(1):3–19, 2011. http://dx.doi.org/10.1007/s12021-010-9085-7 doi:10.1007/s12021-010-9085-7.
52. P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on probabilistic graphical models in evolutionary computation. *Journal of Heuristics*, 18(5):795–819, 2012.
53. E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, and C. Boeres. Inexact graph matching by means of estimation of distribution algorithms. *Pattern Recognition*, 35(12):2867–2880, 2002.
54. C. F. Lima, M. Pelikan, D. E. Goldberg, F. G. Lobo, K. Sastry, and M. Hauschild. Influence of selection and replacement strategies on linkage learning in BOA. In *Proceedings of the 2007 Congress on Evolutionary Computation CEC-2007*, pages 1083–1090. IEEE Press, 2007.
55. H. Mühlenbein and R. Höns. The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27, 2005.
56. R. Santana, P. Larrañaga, and J. A. Lozano. The role of a priori information in the minimization of contact potentials by means of estimation of distribution algorithms. In *Proceedings of the Fifth European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, volume 4447 of *Lecture Notes in Computer Science*, pages 247–257. Springer, 2007.
57. S. Brownlee, J. McCall, Q. Zhang, and D. Brown. Approaches to selection and their effect on fitness modelling in an estimation of distribution algorithm. In *Proceedings of the 2008 Congress on Evolutionary Computation CEC-2008*, pages 2621–2628, Hong Kong, 2008. IEEE Press.
58. C. Echegoyen, J. A. Lozano, R. Santana, and P. Larrañaga. Exact Bayesian network learning in estimation of distribution algorithms. In *Proceedings of the 2007 Congress on Evolutionary Computation CEC-2007*, pages 1051–1058. IEEE Press, 2007.
59. C. Echegoyen, R. Santana, J. A. Lozano, and P. Larrañaga. The impact of probabilistic learning algorithms in EDAs based on Bayesian networks. In *Linkage in Evolutionary Computation*, Studies in Computational Intelligence, pages 109–139. Springer, 2008.
60. M. Hauschild, M. Pelikan, C. Lima, and K. Sastry. Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In D. Thierens et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2007*, volume I, pages 523–530, London, UK, 2007. ACM Press.

61. C. F. Lima, F. G. Lobo, and M. Pelikan. From mating pool distributions to model overfitting. In Maarten Keijzer, editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2008*, pages 431–438, New York, NY, USA, 2008. ACM.
62. R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.
63. M. Hauschild and M. Pelikan. Enhancing efficiency of hierarchical BOA via distance-based model restrictions. MEDAL Report No. 2008007, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), 2008. Available from: `http://medal-lab.org/files/2008007.pdf`.
64. S. Baluja. Incorporating a priori knowledge in probabilistic-model based optimization. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 205–222. Springer, 2006.
65. H. Mühlenbein and T. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal on Approximate Reasoning*, 31(3):157–192, 2002.
66. R. Santana, P. Larrañaga, and J. A. Lozano. Side chain placement using estimation of distribution algorithms. *Artificial Intelligence in Medicine*, 39(1):49–63, 2007.
67. M. Pelikan and K. Sastry. Fitness inheritance in the Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2004*, volume 3103 of *Lectures Notes in Computer Science*, pages 48–59. Springer Berlin Hidelberg, 2004.
68. K. Sastry, M. Pelikan, and D. Goldberg. Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *Proceedings of the 2004 Congress on Evolutionary Computation CEC-2004*, pages 720–727, Portland, Oregon, 2004. IEEE Press.
69. 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.
70. S. Shakya, J. McCall, and D. Brown. Using a Markov network model in a univariate EDA: An empirical cost-benefit analysis. In H. G. Beyer and U. M. O'Reilly, editors, *Proceedings of Genetic and Evolutionary Computation Conference GECCO-2005*, pages 727–734, Washington, D.C., USA, 2005. ACM Press.
71. R. Armañanzas, Y. Saeys, I. Inza, M. García-Torres, C. Bielza, Y. van de Peer, and P. Larrañaga. Peak-bin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):760–774, 2011.
72. R. Santana, A. Mendiburu, and J. A. Lozano. Structural transfer using EDAs: An application to multi-marker tagging SNP selection. In *Proceedings of the 2012 Congress on Evolutionary Computation CEC-2012*, pages 3484–3491, Brisbane, Australia, 2012. IEEE Press. http://dx.doi.org/10.1109/CEC.2012.6252963 doi:10.1109/CEC.2012.6252963.
73. H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Regularized continuous estimation of distribution algorithms. *Applied Softcomputing*, 2012. In press.
74. H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Multi-objective optimization with joint probabilistic modeling of objectives and variables. In *Evolutionary Multi-Criterion Optimization: Sixth International Conference, EMO 2011*, Lecture Notes in Computer Science, pages 298–312. Springer Berlin-Heidelberg, 2011.
75. C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano. Analyzing the probability of the optimum in EDAs based on Bayesian networks. In *Proceedings of the 2009 Congress on Evolutionary Computation CEC-2009*, pages 1652–1659, Norway, 2009. IEEE Press. http://dx.doi.org/10.1109/CEC.2009.4983140 doi:10.1109/CEC.2009.4983140.
76. C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano. A quantitative analysis of estimation of distribution algorithms based on Bayesian networks. *IEEE Transactions on Evolutionary Computation*, (99):1–17, 2011. http://dx.doi.org/10.1109/TEVC.2010.2102037 doi:10.1109/TEVC.2010.2102037.
77. O. Sporns. *Networks of the Brain*. The MIT Press, 2010.
78. N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences (PNAS)*, 102(39):13773–13778, 2005.
79. O. Sporns and R. Kötter. Motifs in brain networks. *PLoS Biology*, 2(11):e369, 2004.
80. P. B. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer-Verlag New York Inc, 2009.
81. K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):1–25, 2008.
82. A. Barrat, M. Barthélémy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences (PNAS)*, 101:3747–3752, 2004.
83. G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.