
A Study on the Evolution of Bayesian Network Graph Structures

Jorge Muruzábal¹ and Carlos Cotta²

¹ Departamento de Estadística e Investigación Operativa, ESCET, Universidad Rey Juan Carlos, 28933 - Móstoles, Spain

² Departamento de Lenguajes y Ciencias de la Computación, ETSI Informática, Universidad de Málaga, Campus de Teatinos, 29071 - Málaga, Spain

Summary. Bayesian Networks (BN) are often sought as useful descriptive and predictive models for the available data. Learning algorithms trying to ascertain automatically the best BN model (graph structure) for some input data are of the greatest interest for practical reasons. In this paper we examine a number of evolutionary programming algorithms for this network induction problem. Our algorithms build on recent advances in the field and are based on selection and various kinds of mutation operators (working at both the directed acyclic and essential graph level). A review of related evolutionary work is also provided. We analyze and discuss the merit and computational toll of these EP algorithms in a couple of benchmark tasks. Some general conclusions about the most efficient algorithms, and the most appropriate search landscapes are presented.

1 Introduction

A Bayesian Network (BN) is a graphical model postulating a joint distribution for a target set of discrete random variables. Critical qualitative aspects relate to stochastic dependencies and are determined by the underlying graphical structure, a *Directed Acyclic Graph* (DAG). To deal with the problem of learning sensible BN models from data (a problem known to be NP-hard), a number of algorithms have been considered to search various target spaces, including most notably the space of DAG structures (b-space) and the space of equivalence classes of DAG structures (e-space), see e.g. [33; 2]. The field is very active and further representation schemes keep emerging in the literature, see e.g. Studený's algebraic approach [42]. For the most familiar search spaces, some key insights and guiding principles of interest have emerged over time [25; 11; 9; 36]. We adhere here to these principles as we try to evaluate their components in a rich evolutionary framework.

Evolutionary algorithms have been successful by now in many applications; in particular, they have been considered in this context as well [31]. This family

of algorithms can be seen as an interesting class of population-based score-and-search methods, where the *fitness* measure is equated to some standard scoring metric like the marginal likelihood [25], and we can enjoy the benefits of our experience and theoretical results in the evolutionary computation field. We focus here on the *evolutionary programming* (EP) paradigm, see [18; 17] for general reference. The EP paradigm is based on the pressure exerted by selection and mutation alone, i.e., no recombination is used. Recombination is of course an important, often useful heuristic for mixing genetic material and indeed it has been often explored in b-space [31; 14; 43]. To the best of our knowledge, however, no previously proposed evolutionary algorithm (be it EP or otherwise) follows the aforementioned theoretical principles as our EP algorithms do.

The specific approaches we consider differ in either search space or type of *neighborhood*. The latter turns out to be a key concept for search algorithms: the neighborhood of a graph (in b- or e-space) equals the set of DAGs that can be reached from that DAG in a single mutation. This clearly depends on the battery of operators available. In this work, we consider on one hand the approach based on essential graphs (equivalence classes or e-space) suggested by the results of [11]. On the other hand, we consider two families of algorithms working directly in b-space inspired by the results of [9]. We denote these three algorithms as EPQ, EPNR and EPAR respectively. We are specifically interested in analyzing the relative performance of these approaches, and the computational tradeoffs involved in their application to the induction of BN structures.

2 Learning Bayesian Networks

A Bayesian Network (\mathbf{G}, θ) encompasses the Directed Acyclic Graph (DAG) \mathbf{G} and a set of probability distributions attached to \mathbf{G} , say $\theta = \theta(\mathbf{G})$. The DAG is the set of links or *arcs* among variables or *nodes*. If we denote the whole set of discrete variables as $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$, each X_i has a set of *parents* denoted by $\Pi_i = \{X_j \in \mathbf{X} \mid (X_j \rightarrow X_i) \in \mathbf{G}\}$. Then, the DAG \mathbf{G} represents the joint distribution

$$P(\mathbf{X}) = \prod_{i=1}^n P(X_i \mid \Pi_i)$$

with the parameterization $P(X_i = k \mid \Pi_i = j) = \theta_{ijk}$, $j = 1, \dots, q_i$; $k = 1, \dots, r_i$; r_i is the number of distinct values that X_i can assume, and q_i is the number of different configurations that Π_i can present.

Two DAGs are (*Markov*) *equivalent* if they encode the same set of independence and conditional independence statements. Each equivalence class, say $[\mathbf{G}]$, can be represented by the *essential* graph [2; 11], a unique *partially* directed acyclic graph or PDAG, say $\tilde{\mathbf{G}}$. If an arc $X \rightarrow Y$ shows up in all

$\mathbf{H} \in [\mathbf{G}]$, then that arc is *compelled* in $[\mathbf{G}]$. If an arc is not compelled, then it is *reversible*, i.e., there exist $\mathbf{H}, \mathbf{K} \in [\mathbf{G}]$ such that \mathbf{H} contains $X \rightarrow Y$ and \mathbf{K} contains $Y \rightarrow X$. The unique PDAG $\bar{\mathbf{G}}$ representing $[\mathbf{G}]$ contains a directed arc for each compelled arc in $[\mathbf{G}]$ and an undirected arc for each reversible arc in $[\mathbf{G}]$. Our e-space refers precisely to the space of those PDAGs which represent some $[\mathbf{G}]$, see below.

There exist two different approaches to learning graphical structure from data, namely, those based on prior conditional independence testing and those usually referred to as score-and-search approaches. The first approach seeks to establish well-founded constraints on the graphical structure, thus simplifying the search space considerably, see [19; 23; 3]. Score-and-search methods omit this step and proceed directly to evaluate all tentative graph structures provided by some method via a suitable scoring metric [25; 24; 12; 7]. While there are also proposals that try to combine the best from each class of methods, here we shall be concerned with the latter class of methods almost exclusively. Besides the scoring metric itself, which is known not to make a big difference in practice for large sample size N , we need to specify the set of *traversal* operators that will be used to search for better solutions locally.

Given a BN (\mathbf{G}, θ) and a data matrix \mathbf{D} with n columns and N (exchangeable) rows, there are several ways to measure the quality of fit to the data [24]. We focus here on the *marginal likelihood*:

$$P(\mathbf{D}|\mathbf{G}) = \int P(\mathbf{D}|\mathbf{G}, \theta)\pi(\theta|\mathbf{G})d\theta .$$

A closed-form expression is available for $P(\mathbf{D}|\mathbf{G})$ in the case of suitable Dirichlet-based priors $\pi(\theta|\mathbf{G})$ under certain assumptions [25]. Specifically, we take

$$\pi(\theta|\mathbf{G}) \propto \prod_{i,j} \prod_k \theta_{ijk}^{\alpha_{ijk}-1}$$

where $\alpha = \{\alpha_{ijk}\}$ is the *virtual count* hyperparameter ($\alpha_{ijk} > 0$). These α_{ijk} must be supplied by the user (just like the complete data set \mathbf{D}), but we denote our fitness or basic DAG scoring metric as $\Psi = \Psi(\mathbf{G}; \mathbf{D}) = \log P(\mathbf{D}|\mathbf{G})$ for simplicity.

A given measure Ψ is called *score-equivalent* if it is constant over each equivalence class $[\mathbf{G}]$. The present Ψ is score-equivalent if $\alpha_i = \sum_{j,k} \alpha_{ijk} \equiv \alpha$ for some $\alpha > 0$, the so-called BDe metric [25]. We consider below the BDeu(α) metric $\alpha_{ijk} = \alpha/r_i q_i$, see e.g. [8]. Another typical option is $\alpha_{ijk} = 1$, the well-known (but not score-equivalent) K2 metric [13]. Note that the score of a given PDAG $\bar{\mathbf{G}}$ is taken as the constant value assumed by members of the associated $[\mathbf{G}]$; genuine equivalence class metrics can be defined too [10].

2.1 Learning Equivalence Classes

Let $\bar{\mathbf{G}}$ denote the unique PDAG structure representing some equivalence class $[\mathbf{G}]$. Among other things, we know that $\bar{\mathbf{G}}$ and any $\mathbf{G} \in [\mathbf{G}]$ share the same

skeleton or connectivity pattern (ignoring directionality) and the same *v-structures*. A *v-structure* is a substructure $X \rightarrow Z \leftarrow Y$ where X is not linked to Y directly. Note that not all PDAGs represent equivalence classes, only *completed* PDAGs (CPDAGs) do. A related class of PDAG models is discussed in [1]; however, this class does not exhibit the nice one-to-one correspondence that we have between equivalence classes and CPDAGs.

Chickering [11] presents six operators for introducing local variation in existing CPDAGs, namely, `InsertU`, `DeleteU`, `InsertD`, `DeleteD`, `ReverseD` and `MakeV`. The first five operators are rather self-explanatory. As to the sixth, it transforms a substructure $X - Z - Y$ (where X is not linked to Y directly) into the *v-structure* $X \rightarrow Z \leftarrow Y$. Note that each of these operators changes either the skeleton or the number of *v-structures* and thus guarantees that a new equivalence class is visited. An example of the application of each of these operators is provided in Figure 1.

The modified CPDAGs need not be evaluated from scratch: efficient score-updating formulae are provided for each operator [11]. The key idea behind this local scoring is that a *decomposable*, score-equivalent metric Ψ is typically used (for example, both $K2$ and $BDeu(\alpha)$ are decomposable). A metric Ψ is said decomposable if, for some function σ (and implicit data), $\Psi(\mathbf{G}) = \sum_{i=1}^n \sigma(X_i, \Pi_i)$, where calculation is restricted in each summand to a single node X_i and its parents Π_i . Thus, only those nodes whose Π_i is changed by the operator need to be updated. To illustrate, the change in score attributed to a particular (valid) move deleting $X \rightarrow Y$ in \mathbf{G} (and leading to some \mathbf{H}) can be expressed as $\Psi(\mathbf{H}) - \Psi(\mathbf{G}) = \sigma(Y, \Lambda_1) - \sigma(Y, \Lambda_2)$, where $\Lambda_1 \subset \mathbf{X}$ is the set of nodes connected to Y (with either a directed or undirected arc), and $\Lambda_2 = \Lambda_1 \cup \{X\}$. Similar or slightly more complex expressions hold for the remaining operators.

While the six operators are all local in principle, there may be “cascading” implications in some moves. For example, as seen in Figure 1, `DeleteD` and `ReverseD` may make other directed arcs undirected. Or, after applying `MakeV`, many arcs may switch from undirected to directed. Hence, it is difficult to predict the behavior of the tentative graphs produced along the way, and indeed surprises may arise in some cases, see [15] and below. In practice, we find the outcome $\bar{\mathbf{H}} = \omega(\bar{\mathbf{G}})$ by applying two key algorithms in turn [11]. We first use the PDAG-to-DAG routine to extract a member DAG \mathbf{H} from the raw result of the mutation, say $\bar{\mathbf{H}}_r$. If no such \mathbf{H} can be found, the intended mutation is not valid (the PDAG can not be completed; a compact validity test is provided for each operator to prevent unnecessary computations). Otherwise we call the DAG-to-CPDAG routine (with input this \mathbf{H}) to determine the resulting (validated) $\bar{\mathbf{H}}$. As discussed below, these two routines can be used in reverse order to move within the same equivalence class in a random way (we have included a stochastic component in the PDAG-to-DAG routine, namely, the order in which the nodes will be traversed so as to assign directionality to undirected arcs).

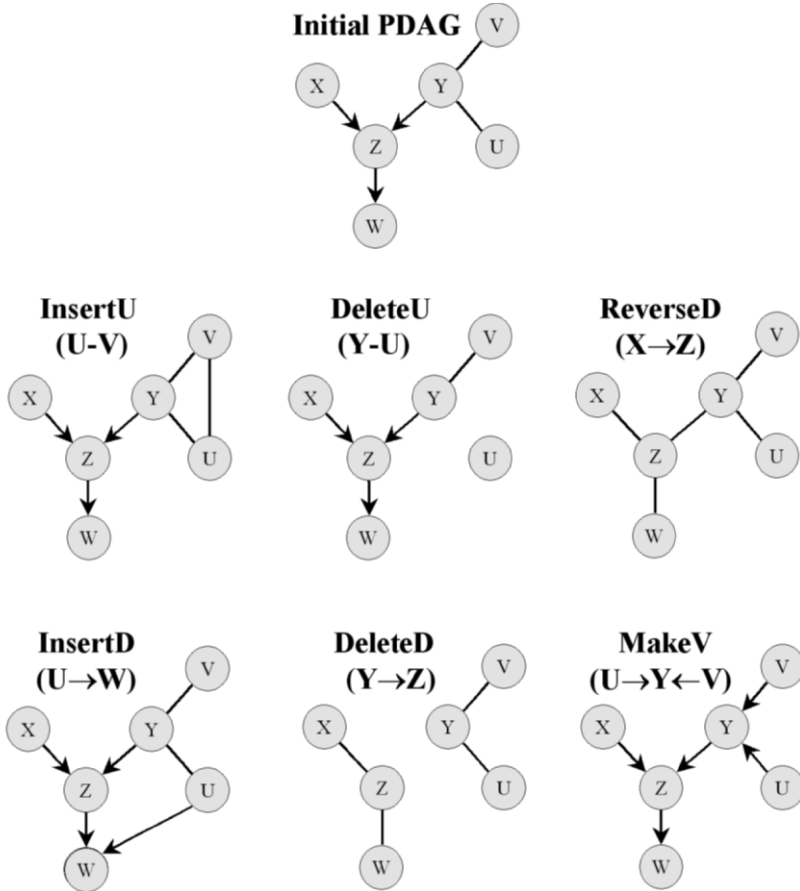


Fig. 1. Operators for traversing e-space.

2.2 Inclusion-Driven Learning

Castelo and Kočka [9] and others argue that traversal operators which respect the *inclusion boundary* (IB) condition or principle have appealing theoretical properties. Briefly, if the true distribution generating the data can be expressed as a BN model, and if certain reasonable assumptions concerning the score function are made, then, in the long run (for large sample size N) we are guaranteed to reach the target model when we use traversal operators that verify this principle. A traversal operator respects the IB condition if the neighborhood associated to a given \mathbf{G} always contains its IB, say $\mathcal{IB}(\mathbf{G})$. The $\mathcal{IB}(\mathbf{G})$ collection of models contains all those “immediately next” to \mathbf{G} in a precise distributional sense. The bottom line is that traversal operators should be designed so that they possibly visit any “sufficiently close neighbor” in this sense. On this matter, the standard NR and AR (No and All Reversals

respectively) neighborhoods are a primary reference. In the case of NR, only the usual `InsertD` (Insert directed arc at a random valid location, that is, wherever the insertion does not form a cycle) and the similarly defined `DeleteD` operators are allowed, whereas AR incorporates also `ReverseD`. Natural questions are: How well do the basic NR and AR neighborhoods do with regard to the IB condition? Can we find a traversal operator whose neighborhood coincides exactly with $\mathcal{IB}(\mathbf{G})$? If so, how much should $\mathcal{IB}(\mathbf{G})$ be augmented (if at all)?

The answer to the second question is given by the particular neighborhood ENR (Equivalence class-based NR jump), see e.g. [9]. For a given \mathbf{G} , this consists of the union of all DAGs that belong to the standard NR neighborhood of \mathbf{G} together with all DAGs that belong to the same neighborhood of all other $\mathbf{H} \in [\mathbf{G}]$. The idea in the implicit intra-class navigation is that certain areas of $[\mathbf{G}]$ may be closer to some intermediate equivalence classes of interest than others. A tentative improvement over ENR is provided by the ENCR neighborhood [9]. This is defined just like ENR, except now the `ReverseD` operator is allowed and restricted to non-covered arcs. A given arc $Y \rightarrow X$ is said to be *covered* in the DAG \mathbf{G} if $\Pi_X = \{Y\} \cup \Pi_Y$ holds. In other words, if there exists another arc $Z \rightarrow X$ then $Z \rightarrow Y$ must also exist (and vice versa). Hence, a covered $Y \rightarrow X$ can not be part of any v-structure. It follows that the reversal of a covered arc does not change neither the skeleton nor the number of v-structures. Therefore, non-covered arc reversals are guaranteed to leave the current equivalence class.

Of course ENR encompasses a huge number of graphs and hence needs to be *simulated* by a random walk or otherwise. Given a DAG \mathbf{G} , we can move within $[\mathbf{G}]$ by iterated (random) covered arc reversal. Let r the number of calls to be made for each move. It is argued in [9] that r need not be very large because equivalence classes contain an average of less than four DAGs [21]. In practice, the algorithms will need to handle (DAGs from) equivalence classes close to the target. Hence, if the target equivalence class is believed to be large, then r may need to be larger. In any case, once the r stipulated random reversals have taken place, the resulting structure is modified according to the standard NR (implementing ENR) or whatever neighborhood is implied by the traversal operators (implementing also ENCR).

Note that there exist other learning algorithms which also respect the IB condition. These include the GES algorithm proposed by Chickering [12] (a fully greedy algorithm which begins with an empty graph) and the KES generalization considered by Nielsen et al. [36]. Furthermore, complying Markov Chain Monte Carlo (MCMC) algorithms can also be devised [9]. We believe that our current EP approach is likely to cooperate effectively along these alternative lines too, see the concluding section.

3 Evolutionary Approaches to Network Induction

Several evolutionary algorithms have been proposed for the present graphical model induction task. The seminal paper by Larrañaga et al. [31] presents the first genetic algorithm (GA) in b-space, see also [41]. The parallel paper by Larrañaga and coworkers [29] examines the role of the GA when restricted to explore the space of topological orderings (essentially permutations) of the variables. A popular heuristic algorithm like K2 [13] takes one such topological order as input and returns a fitted BN that respects that order.

More recent work in the area by this research team involves the so-called Estimation of Distribution Algorithms, see e.g. [7]. Evolutionary algorithms of this sort replace crossover with sampling from a model fitted to the best individuals in the current population [39; 30]. Model distributions here refer in principle to the space of DAGs; hence, they must remain relatively simple to be tractable. For example, it is not uncommon to model univariate (marginal) arc behavior or simple arc-to-arc dependencies. It follows that graphs sampled from these models may include cycles and thus need repairing [7].

Wong et al. [44] use an EP approach that aims to enjoy the advantages of both the prior testing and the score-and-search approaches mentioned earlier. They preprocess the data and use the standard Mutual Independence measure to compute a matrix that evaluates the strength of every possible arc in the emerging DAG. Mutation operators include the classical operators considered here as well as other, less frequent operators. These operators are sensitive to the MI information, in the sense that, for instance, the weakest arc is more likely to be deleted when the deletion operator is called and so on. This bias acts in the same way throughout the run. Note that the MI measure can only capture the value of a given arc *taken in isolation*. In actual learning runs, however, we should find that the value of a given arc $Y \rightarrow X$ depends more naturally on whether and which other arc(s) $Z \rightarrow X$ are concurrent. In fact, it has been pointed out that the offspring produced by this so-called MDLEP method are often worse than their parents [46]. We shall explicitly consider this improvement rate in our experiments below.

Harwood and Scheines [23] propose an *annealed* GA to search over the space of equivalence classes of DAGs or e-space. These authors provide a good discussion of the pros and cons of the various strategies available for inducing networks from data. They handle the slightly different problem in which variables are continuous and linear local regressions are computed at each node so as to provide arcs with a certain coefficient. These coefficients come along with the graph and play the role of the conditional distributions θ in the discrete setting. Harwood and Scheines suggest to improve the standard evolutionary process by adding an annealing scheme that slowly increases the penalty for complex models in the fitness function (so that the system works initially with relatively dense networks). They further prune the search space along the run by permanently banning adjacencies from future consideration if these adjacencies become extinct in the current population. Harwood and

Scheines also tackle the issue of dealing with relatively few data (compared to the number of available variables, $N \ll n$).

Cotta and Muruzábal [14] propose various crossover operators in b-space. Their approach is based on *preventing* the formation of cycles (to avoid the costly repairing, see [16]). It also involves a (surrogate of) the *conditional* mutual independence measure, which assesses the value of arcs in the context of their relevant parent sets Π in the parent DAGs. This measure is used to rank the goodness of the various arcs possibly transmitted from parents to offspring. The proposed crossover operators exploit this information in various ways making important arcs more likely to be transmitted from parent to offspring DAGs. Cotta and Muruzábal also suggest that *respectful* strategies (transmitting routinely all arcs shared by both parent DAGs) might prove advantageous to crossover operators in this setting.

Wong and Leung [46] present the hybrid evolutionary algorithm HEA. Their HEA is based on the so-called *merge* operator, a parent-set-based version of crossover. Given two parent DAG structures, merge chooses the parent set Π_i of variable i in the offspring from the corresponding Π_i in the parent DAGs, with the goal that offspring exhibit better overall scores than their progenitors. HEA is also based on cycle-prevention. The merge operator is given priority over the mutation operators used in [44] since it exhibits several desirable properties, most notably, that score information related to conserved Π_i can be reused economically.

Van Dijk and Thierens [43] recognize the potential gain provided by the PDAG-based non-redundant encoding and discuss a GA that allows searching in e-space. They point out that the standard DAG representation may jeopardize the fusion of useful building blocks and thus lead to a poor crossover operator in general. However, in their implementation PDAGs are instantiated as DAGs prior to crossover; then the DAG offspring are cleaned up (cycles are broken) and reinserted as PDAGs. Van Dijk and Thierens [43] dismiss the framework of Chickering [11] (adopted here) arguing that this “requires a more complicated implementation and is only of practical interest”. While they also acknowledge the computational cost of the numerous DAG-to-PDAG and CPDAG-to-DAG calls in their proposal, the type of neighborhood implemented by the set of traversal (and/or crossover) operators is crucial for our score-and-search algorithms to be able to escape local optima.

Another type of parent-set-based evolutionary algorithm has been proposed by Wong et al. [45], namely the cooperative co-evolutionary GA or CCGA. This builds upon the two-phase approach discussed earlier [44], so that the search space is constrained by the “verified assertions” made in the prior testing phase. The idea is then to search (separately) over the space of parent configurations Π_i in each case. The maximum size of Π is kept limited throughout. The optimal parent sets for the different variables (species) are assembled together to form the final complete DAGs (again, some postprocessing may be needed for cycle removal). The fitness measure for the individual search processes does include a component that evaluates the

degree of cooperation of tentative Π_i . This component is based on the scores of the “collaborative structures” assembled from good representatives of each species along the way. Similar ideas have been known for a while in the wider evolutionary context, see e.g. [34].

To summarize, evolutionary approaches have flourished in recent years for the graph induction problem. These algorithms can be tailored to deal with this problem in various ways, and a host of ideas frequently found useful are brought to play to our advantage. However, no conclusive assessment has been reached yet, so more detailed comparative work and benchmarking are in order.

4 Evolutionary Search Landscapes for Optimal Network Induction

In this section we review the details of our evolutionary algorithms. We first review EPQ in e-space, then continue with EPNR and EPAR in b-space. It is useful to begin by describing the common skeleton in these EP algorithms.

4.1 Standard EP Setting

The common steps in our algorithms are the following: (i) we begin with a population of P randomly initialized graphs and we evaluate them using the fitness or scoring metric Ψ . (ii) At each generation, P members of the current population are selected by means of binary tournament (two graphs are randomly drawn and the highest score wins). (iii) Each selected graph may be preprocessed. (iv) Either the original or the preprocessed graph is mutated once by selecting an operator ω from the available battery Ξ according to some distribution Ω , and applying it at a random (valid) entry point in the target graph. (v) All P mutated graphs are (locally) evaluated and stored. (vi) Finally, the best P out of the $2P$ available structures at this point are selected for the next generation, the remaining P are discarded and a new iteration takes place.

The probability distribution Ω over the battery Ξ may be fixed in evolutionary time, or it may be dynamic (in various ways). At the moment, we use a stationary, uniform Ω throughout the process and for all individuals.

Initialization of DAG structures can be pursued either in a purely random way or heuristically. In the first case, we have devised a simple randomization routine in which parameter $\delta \in [0, 1]$ controls the arc density of the resulting graph. More sophisticated approaches to uniformly distributed DAG generation exist, see e.g. [27]. In the second case, the K2 heuristic is used, taking a random permutation of the variables as seed. The process is further controlled by π_{\max} , the maximum number of parents allowed per variable. Note that this limit is set only on the initial structures, it is not enforced along the

run. Initial (valid) PDAGs are easily generated from random DAGs by using the DAG-to-CPDAG routine mentioned in Section 2.1.

4.2 Neighborhoods in E-Space and B-Space

No preprocessing is carried out in the case of EPQ. The particular neighborhood used here does not quite contain the target ENR (although it typically contains many other DAGs), so EPQ does not really enjoy the convergence property discussed in Section 2.2. Recall that the operators analyzed in [11] are `InsertU`, `DeleteU`, `InsertD`, `DeleteD`, `ReverseD` and `MakeV`. It was shown in [35] that all operators are needed for best performance, so we allow all in Ξ here. Recall also that all of them change the current equivalence class and all can be scored efficiently.

Note that some operators may not find a suitable entry point in the selected CPDAG and hence may become non-applicable (in which case a different operator should be selected etc.). If, on the other hand, one or more appropriate entry points can be found for the selected ω , then the operator is tentatively applied at a randomly selected point. If the mutated CPDAG $\bar{\mathbf{H}} = \omega(\bar{\mathbf{G}})$ passes the corresponding validity test, it is incorporated to the offspring population. We track the *success* ratio of each operator $\varepsilon = \varepsilon(\omega)$ during the replacement stage, i.e., the number of CPDAGs that ω produced and made it to the next population.

We now continue with our EP algorithms in b-space, EPNR and EPAR. Note that preprocessing of a selected DAG \mathbf{G} refers to the *navigation* within the class $[\mathbf{G}]$ containing \mathbf{G} . As explained in Section 2.2, this navigation is achieved via a series of covered arc reversals.

Castelo and Kočka [9] discuss implementations of the ENR, ENCR and other neighborhoods. They refer to the version covering ENR as RCARNR r (for r Repeated Covered Arc Reversals followed by a NR jump); it goes hand by hand with our EPNR(r) algorithm. As noted above, ENCR is simulated similarly and the corresponding algorithm is denoted as RCARR r . We adopt below a simpler implementation of ENCR allowing all arc reversals, which we call EPAR(r). The case $r = 0$ (no navigation at all) transforms radically the associated neighborhoods, with the result that the theoretical support is lost [9]. However, we still consider EPNR(0) (equal, of course, to the standard NR) and EPAR(0) for the sake of reference. We also consider the case in which all navigation steps are collapsed into two as follows: firstly DAG-to-CPDAG is applied to \mathbf{G} to obtain $\bar{\mathbf{G}}$; then, PDAG-to-DAG is used on $\bar{\mathbf{G}}$ to extract another DAG $\mathbf{H} \in [\mathbf{G}]$. This scheme is denoted as $r = \infty$. It had not been proposed previously, although we feel it is a natural competitor for the $r > 0$ alternatives [35].

5 Experiments and Results

The algorithms described above have been deployed on two conspicuous networks: ALARM –a 37-variable network for monitoring patients in intensive care [4]– and INSURANCE –a 27-variable network for evaluating car insurance risks [6]. The equivalence class [ALARM] is represented by a CPDAG with 4 undirected and 42 directed arcs. As to [INSURANCE], it is a larger and denser equivalence class, represented by a CPDAG with 18 undirected arcs and 34 directed arcs. In both cases, a training set of $N = 10,000$ examples was created once by random probabilistic sampling as customary. The BDeu($\alpha = 1$) metric $\Psi(\mathbf{G}|\mathbf{D}, \alpha) = \log P(\mathbf{D}|\mathbf{G})$ is the fitness function (to be maximized). Previous work [35] indicates that this setting $\alpha = 1$ provides the best results (fake dependencies abound for larger values of α , whereas for lower values some true dependencies are lost).

All experiments have been performed using a population size of $P = 100$ individuals. The termination criterion is reaching a number of 500 generations, i.e., 50,000 networks generated. Such a termination criterion follows the common practice in evolutionary computation, where fitness computation is the basic cost unit. Nevertheless, this particular application has a distinctive feature: the goodness of a generated structure is not calculated from scratch, but by means of local evaluations (recall the decomposability of our fitness function). Since the number of such local evaluations depends on the operator and on the value of r , we have monitored the accumulated number of local evaluations across the run, to obtain another –possibly more representative– figure of cost. Two different initialization settings have been considered: random initialization using density value $\delta = 0.05$, and K2 initialization with maximum number of parents per variable $\pi_{\max} = 2$.

The results are shown in Figure 2. Notice firstly the results of EPNR(0). These are remarkably inferior to those of any EPNR($r > 0$) for both networks. This confirms the limitations of the basic NR neighborhood. As soon as $r > 0$, there is a sharp performance improvement. This improvement clearly supports the usefulness (for this particular neighborhood) of intra-class navigation, for it increases the connectivity of the search space (and hence decreases the number of local optima). This is also true for EPAR, although the difference among the several values of r is not so large in this case. The effect of using the denser AR neighborhood is here dominant. Indeed, by taking $r > 0$ new paths in b-space are possible, although the enhanced inter-class navigation capability offered by ReverseD remains the prime feature (as it can be noted by comparing the behavior of EPAR(0) with that of EPNR(r)). Actually, the performance of EPAR(r) is always superior to that of its EPNR(r) counterpart. EPQ is also better than EPNR(0) and tends to perform similarly to EPAR. Since the connectivity in e-space is very rich, it is worth investigating which operators are most useful.

As can be seen in Figure 3, there exists naturally a general decreasing trend in success rates (improvements are less frequent in the latter stages of

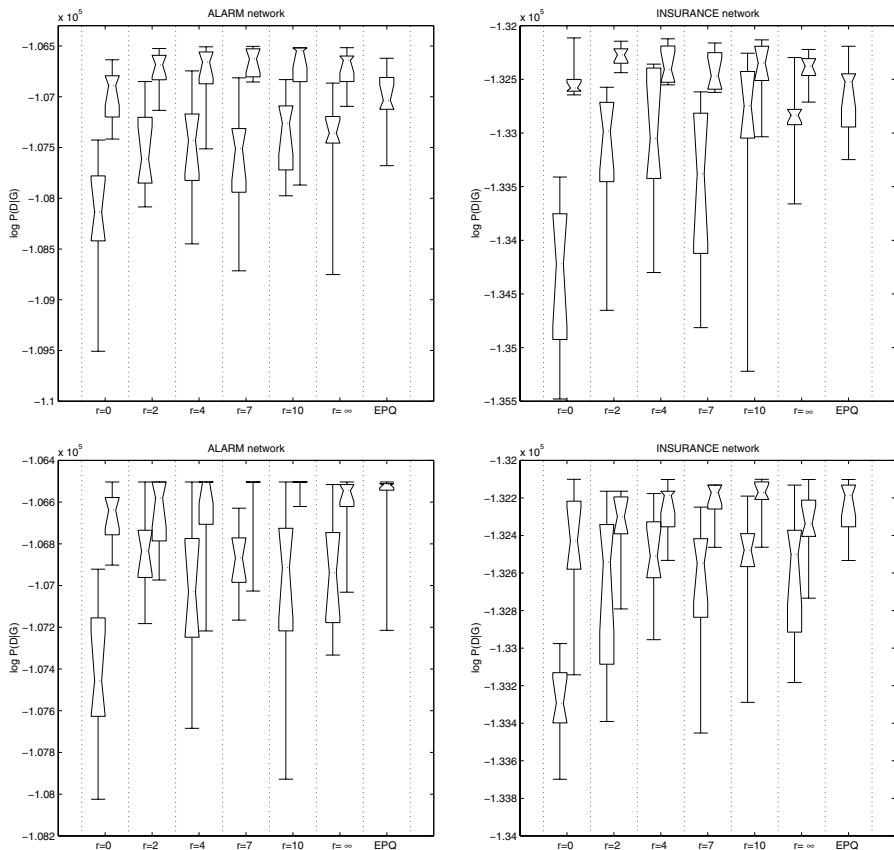


Fig. 2. Results by the proposed EP algorithms (each boxplot summarizes ten runs). (Top) Random initialization (bottom) Heuristic initialization. For each value of r , the left boxplot corresponds to EPNR, and the right one to EPAR. Notice the use of different scales in each plot.

the run). Also for this reason, lower success levels are obtained when using heuristic initialization (the algorithm performs its run at a higher fitness level). The different decreasing rates shed some light on the relative contribution of operators. In particular, note that the undirected-arc-based, “brick and mortar” operators `InsertU` and `MakeV` tend to maintain the highest success ratios by the end of the run. The injection of v -structures appears thus crucial for balancing the adequate proportion of directed and undirected arcs. Indeed, if `MakeV` were removed from the set of available operators, directed arcs would begin to vanish very quickly [35]. On the other hand, `DeleteD`, `ReverseD` and `DeleteU` are the ultimately least useful.

This overall success picture may suggest the following EPQ dynamics. It appears that the typical behavior of this algorithm is to first trim massively

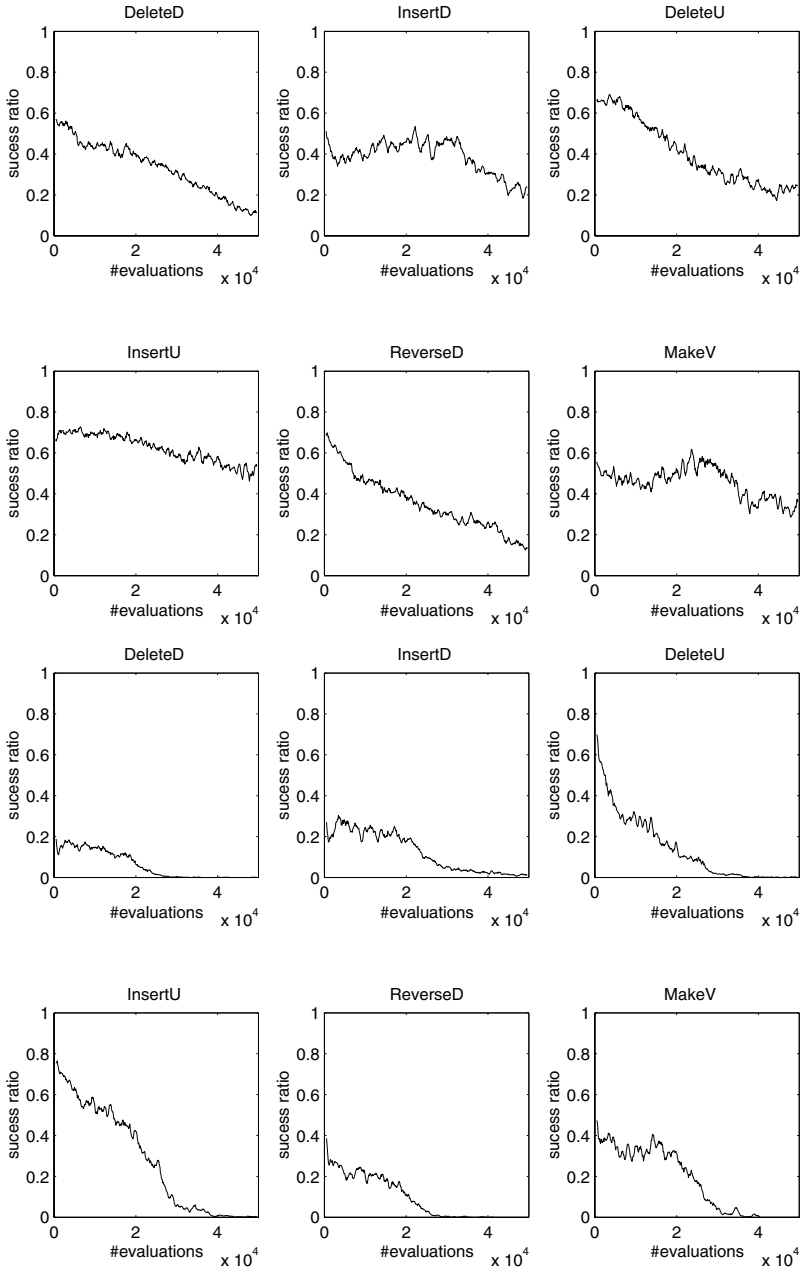


Fig. 3. Mean success ratio (averaged for ten runs) of e-space operators when using random initialization (two upper rows) and heuristic initialization (two lower rows). Results correspond to the ALARM network.

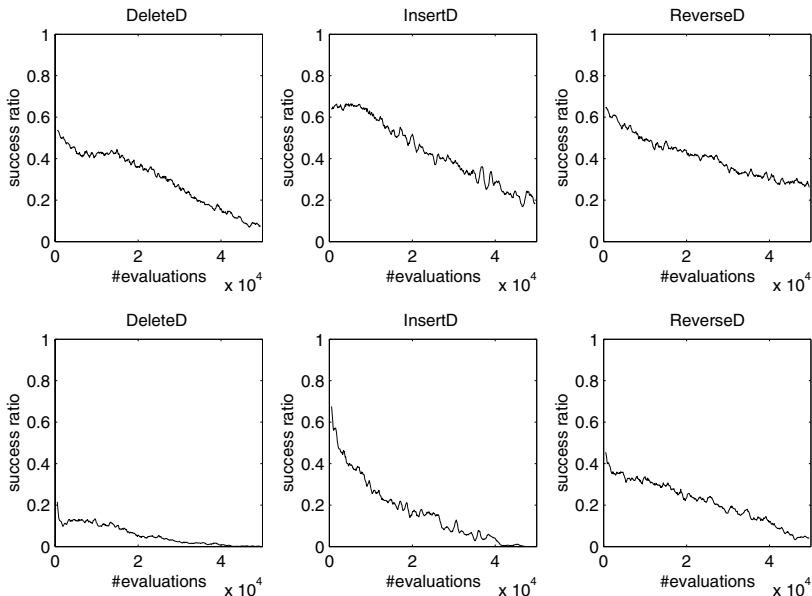


Fig. 4. Mean success ratio (averaged for ten runs) of b-space operators when using random initialization (upper row) and heuristic initialization (lower row). Results correspond to the ALARM network.

all irrelevant arcs, ending up with a relatively small structure with mostly undirected arcs. As the structure grows from this basis, some useful set of both compelled and reversible arcs is secured. Tentative directionality is assigned by **MakeV** and a higher number of sensible directed arcs and v-structures emerge over time.

It must also be noted that **ReverseD** plays an important role in helping the networks size down appropriately, as indicated by the denser networks obtained when this operator is removed [35]. This important role is also clear in the case of traversing b-space –see Figure 4– where its removal results in handicapped search capabilities as indicated by the poor results of EPNR with respect to EPAR.

Tables 1 and 2 show the structural properties of the networks evolved using heuristic initialization. Two facts must be highlighted: firstly, the number of recovered arcs (in the true equivalence class) is always bigger for EPAR; also, networks tend there to be smaller (like in the case of EPQ). The best run for ALARM recovers all but one of the arcs. For INSURANCE, the best network has a Hamming distance of 11 with respect to the original one. From an absolute point of view, the quality of these results is high, and comparable to the state-of-the-art.

A final comment must be done regarding the somewhat hidden cost of performing intra-class navigation, namely the fact that local score-updates

Table 1. Structure of the networks generated by EPNR and EPAR using heuristic initialization (averaged for ten runs). From left to right in each case: number of recovered undirected arcs, number of recovered directed arcs, and total number of arcs.

r	EPNR						EPAR					
	ALARM			INSURANCE			ALARM			INSURANCE		
	S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}
0	2.4	17.8	66.0	4.9	15.4	53.3	3.8	32.8	50.2	6.4	24.5	47.5
2	3.4	34.1	50.9	7.3	23.6	48.5	3.7	38.5	48.5	7.4	25.9	46.7
4	3.4	35.3	51.9	6.0	25.6	47.4	3.9	38.0	48.0	9.9	27.2	46.6
7	3.2	34.2	52.7	5.9	23.8	48.0	4.0	39.4	46.7	9.4	27.9	46.2
10	3.0	34.6	54.0	8.3	24.5	47.3	4.0	40.5	46.3	10.0	28.2	46.2
∞	3.0	32.4	53.6	6.9	23.7	48.1	3.8	35.0	48.7	7.9	25.9	46.7

Table 2. Structure of the networks generated by EPQ using heuristic initialization (averaged for ten runs). Interpretation is as before.

ALARM			INSURANCE		
S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}
4.0	36.4	48.0	10.6	27.0	46.4

are required whenever a covered arc is reversed. Recall that, while the sum of local scores $\sum \sigma(X_i, \Pi_i)$ would remain unchanged, the inner terms would change. That is, when reversing the covered arc $X_i \rightarrow X_j$ we would have

$$\sigma(X_i, \Pi_i) + \sigma(X_j, \Pi_j) = \sigma(X_i, \Pi_i \cup \{X_j\}) + \sigma(X_j, \Pi_j \setminus \{X_i\}), \quad (1)$$

but each summand would be different in general (and hence they need being computed in order to keep the consistency of the overall score, see [40] for a related argument). Figure 5 shows the evolution of fitness for the first 70,000 such local evaluations. It turns out that EPAR(0) provides the best tradeoff between computational cost and quality achieved. The difference is remarkable for ALARM; in the case of INSURANCE, EPAR(∞) manages to catch up with EPAR(0) at around 60,000 local evaluations. The remaining settings of r result in slower convergence. Slightly better networks may be attained at the end of the run, but each new network generated required a larger computational effort.

6 Conclusions, Discussion, and Future Work

We have considered a variety of EP algorithms for learning Bayesian Network graph structures from data. Our primary aim has been to investigate the role of intra-class navigation in the enhanced AR and NR neighborhoods,

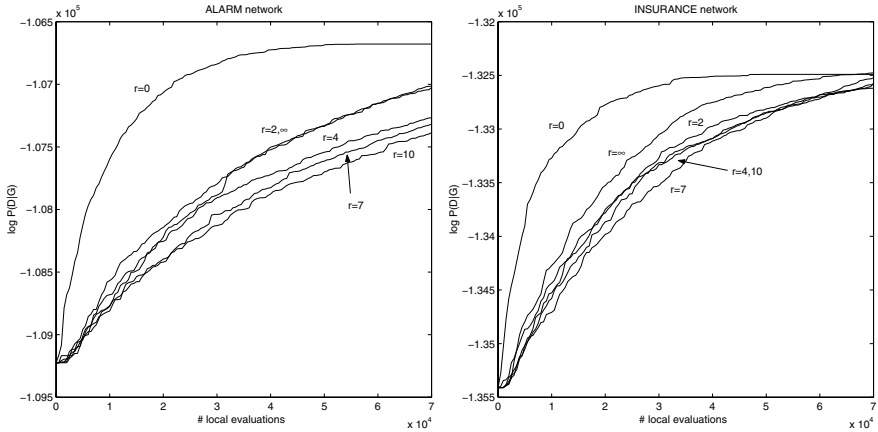


Fig. 5. Evolution of fitness in EPAR as a function of the number of local evaluations. (Right) ALARM (left) INSURANCE.

and the adequacy of the corresponding fitness landscapes for evolutionary exploration. We have reproduced and confirmed in this new context previously reported phenomena such as the poor performance of EPNR(0) and the usefulness of intra-class navigation in this case. Our assessment of the behavior of EPAR indicates that the inter-class navigation capability featured by *ReverseD* dominates the situation though. Furthermore, the extra cost due to intra-class navigation needs to be taken into account. As a result, it might be profitable to disable this latter feature when using the enhanced AR neighborhood, at least during the initial stages of evolution. After all, it is only in the latter stages of the run when the algorithm is more likely to be in a local optimum (or in the basin of attraction thereof), and the increased connectivity provided by covered arc reversals may be more useful. In earlier stages, the benefit would be probably overcome by its associated computational cost. A related computational concern refers to r , the number of covered arc reversals made at each step. Precisely because there appears to be some uncertainty about the best values for this parameter, we believe that the use of an adaptive (or even *self-adaptive* [5]) scheme for varying r across the run may be highly interesting. This is a line for future developments.

There is also a huge potential for exploiting phenotypic information in this context. Our current operators are essentially genotypic (all decisions are fully randomized), and hence blind to quality. The usage of such information can have a positive effect in the convergence properties of the learning algorithms [14]. Confirming the results obtained in this work for these phenotypic operators, and indeed inquiring about their limits with respect to the IB condition, is another appealing line of work.

Another interesting approach mimics the chain irreducibility requirement in MCMC algorithms [33; 20]. MCMC algorithms constitute a major reference

for EP and other evolutionary algorithms in the graphical model induction arena [32; 22]. The reason is perhaps best seen by noticing that, in the multiple-chain case, valid jump proposal distributions can be advantageously based on information from several individuals (chains), see e.g. [26]. The idea is then close to the recombination aim of GA-based algorithms. It may be suspected that insights provided in either area can transfer profitably to the other. Specifically, to encompass the basic ENR neighborhood in our simulation, we can use algorithms that behave like EPNR with probability $p < 1$ only, jumping to e-space and following (selected operators in) EPQ with probability $1 - p$. Perhaps the most useful operators `InsertU` and `MakeV` should be given priority here, at least for the various initialization methods that we have implemented. Again, parameter p can vary along the evolutionary run.

Switching to more practical matters, it is often acknowledged that, for the purpose of data mining, not one but several good candidate graphs will have to be produced. The idea is to first train a *diverse* set of BN models, then isolate recurring features in these models. The significance of these features (not only single edges but also more complex constructs like *Markov blankets*, see below) can be assessed reliably, see e.g. [38]. This problem seems also particularly prone to benefit from an evolutionary approach, since many techniques have been proposed to maintain diversity in the population [17]. In [28], for example, *speciation* based on the so-called fitness-sharing technique is enforced in b-space. The basic similarity measure between DAGs is based on the number of recovered, reversed and missing arcs. Several representative models are extracted from the final population, and these models are combined leading to more robust predictions.

Finally, it has been argued sometimes that, when the main goal is feature extraction and classification, Markov blanket learning may be more appropriate (perhaps the only hope for scalability in some cases) than full BN learning. Given a DAG \mathbf{G} , the Markov blanket (or boundary, MB) of a given variable X (with respect to \mathbf{G}), say $MB(X)$, is defined as the union of all X 's parents, all X 's children, and all parents of X 's children. It is always the case that X is conditionally independent of all other variables given $MB(X)$. Thus, if explaining and predicting the behavior of X is the primary concern, then the graphical substructure depicting the dependencies between X and variables in its MB is all we care about. This observation simplifies the problem considerably. For examples of techniques capable of searching for MB directly and critical reviews of the literature on this subject, see the recent contributions [3] and [37]. The authors of the latter paper also argue that, conversely, full BN learning can be greatly facilitated by learning first each MB separately. In a similar vein, Riggelsen [40] has recently suggested that a MB-based MCMC approach improves upon the more usual, single-arc-based variants. These ideas appear indeed likely to continue to play an important role in future developments in the area.

Acknowledgement

The authors would like to express their gratitude to Robert Castelo, David Chickering, Paolo Giudici, Juan Antonio Lozano and David Ríos. Financial support by grants from Spanish and European agencies is greatly appreciated.

References

- [1] S. Acid and L.M. de Campos. Searching for Bayesian Network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.
- [2] S.A. Andersson, D. Madigan, and M.D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25:505–541, 1997.
- [3] Xue Bai, Clark Glymour, Rema Padman, Joseph Ramsey, Peter Spirtes, and Frank Wimberly. PCX: Markov blanket classification for large data sets with few cases. Technical Report CMU-CALD-04-102, Center for Automated Learning and Discovery, School of Computer Science, Carnegie Mellon University, 2004.
- [4] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In J. Hunter, J. Cookson, and J. Wyatt, editors, *Proceedings of the Second European Conference on Artificial Intelligence and Medicine*, volume 38 of *Lecture Notes in Medical Informatics*, pages 247–256, Berlin, 1989. Springer-Verlag.
- [5] H.-G. Beyer. Toward a theory of evolution strategies: Self adaptation. *Evolutionary Computation*, 3(3):311–347, 1996.
- [6] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.
- [7] R. Blanco, I. Inza, and P. Larrañaga. Learning Bayesian Networks in the space of structures by Estimation of Distribution Algorithms. *International Journal of Intelligent Systems*, 18:205–220, 2003.
- [8] W. Buntine. Theory refinement in Bayesian Networks. In P. Smets, B. D’Ambrosio, and P.P. Bonissone, editors, *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann, 1991.
- [9] R. Castelo and T. Kočka. On inclusion-driven learning of Bayesian Networks. *Journal of Machine Learning Research*, 4:527–574, 2003.
- [10] R. Castelo and M.D. Perlman. Learning essential graph Markov models from data. In J.A. Gámez and A. Salmerón, editors, *First European Workshop on Probabilistic Graphical Models*, pages 17–24, 2003.
- [11] D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

- [12] D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.
- [13] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.
- [14] C. Cotta and J. Muruzábal. Towards a more efficient evolutionary induction of Bayesian Networks. In J. J. Merelo et al., editors, *Parallel Problem Solving From Nature VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 730–739. Springer, Berlin Heidelberg, 2002.
- [15] C. Cotta and J. Muruzábal. On the learning of Bayesian Network graph structures via Evolutionary Programming. In P. Lucas, editor, *Second European Workshop on Probabilistic Graphical Models*, pages 65–72, 2004.
- [16] C. Cotta and J.M. Troya. Analyzing Directed Acyclic Graph recombination. In B. Reusch, editor, *Computational Intelligence: Theory and Applications*, volume 2206 of *Lecture Notes in Computer Science*, pages 739–748. Springer-Verlag, Berlin Heidelberg, 2001.
- [17] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin Heidelberg, 2003.
- [18] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley, New York NY, 1966.
- [19] N. Friedman, I. Nachman, and D. Pe’er. Learning Bayesian Network structures from massive datasets: The sparse candidate algorithm. In H. Dubios and K. Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215, San Francisco CA, 1999. Morgan Kaufmann.
- [20] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.
- [21] S.B. Gillespie and M.D. Perlman. Enumerating Markov equivalence classes of acyclic digraph models. In M. Goldszmidt, J. Breese, and D. Koller, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 171–177, Seattle WA, 2001. Morgan Kaufmann.
- [22] P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1–2):127–158, 2003.
- [23] S. Harwood and R. Scheines. Genetic Algorithm search over causal models. Technical Report CMU-PHIL-131, Department of Philosophy, Carnegie Mellon University, 2002.
- [24] D. Heckerman. A tutorial on learning with Bayesian Networks. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. Kluwer Academic, Dordrecht, 1998.
- [25] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian Networks: the combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

- [26] C.C. Holmes and B.K. Mallick. Parallel Markov chain Monte Carlo sampling: an evolutionary based approach. Technical report, Mathematics Department (Statistics Section), Imperial College, London, 1998.
- [27] J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian Networks with constraints on induced width. In R. López de Mántaras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 323–327, 2004.
- [28] Kyung-Joong Kim, Ji-Oh Yoo, and Sung-Bae Cho. Robust inference of Bayesian Networks using speciated evolution and ensemble. In Mohand-Said Hacid et al., editors, *Foundations of Intelligent Systems: 15th International Symposium*, volume 3488 of *Lecture Notes in Computer Science*, pages 92–101, Berlin Heidelberg, 2005. Springer-Verlag.
- [29] P. Larrañaga, C.M.H. Kuijpers, R.H. Murga, and Y. Yurramendi. Learning Bayesian Network structures by searching for the best ordering with Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 26(4):487–493, 1996.
- [30] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic, 2002.
- [31] P. Larrañaga, M. Poza, Y. Yurramendi, R.H. Murga, and C.M. H. Kuijpers. Structure learning of Bayesian Networks by Genetic Algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(9):912–926, 1996.
- [32] K.B. Laskey and J. Myers. Population Markov chain Monte Carlo. *Machine Learning*, 50(1–2):175–196, 2003.
- [33] D. Madigan, S.A. Andersson, M.D. Perlman, and C.T. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics - Theory and Methods*, 25:2493–2520, 1996.
- [34] D. E. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(4):373–399, 1997.
- [35] J. Muruzábal and C. Cotta. A primer on the evolution of equivalence classes of Bayesian-Network structures. In X. Yao et al., editors, *Parallel Problem Solving from Nature VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 612–621. Springer-Verlag, Berlin Heidelberg, 2004.
- [36] J. D. Nielsen, T. Kočka, and J. M. Peña. On local optima in learning Bayesian Networks. In C. Meek and U. Kjærulff, editors, *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 435–442, 2003.
- [37] J. M. Peña, J. Björkegren, and J. Tegnér. Scalable, efficient and correct learning of Markov boundaries under the faithfulness assumption. In *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty*, volume 3571 of *Lecture*

- Notes in Artificial Intelligence*, pages 136–147, Berlin Heidelberg, 2005. Springer-Verlag.
- [38] J. M. Peña, T. Kočka, and J. D. Nielsen. Featuring multiple local optima to assist the user in the interpretation of induced Bayesian Network models. In *Proceedings of the Tenth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1683–1690, 2004.
 - [39] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. Linkage problem, distribution estimation, and Bayesian Networks. *Evolutionary Computation*, 8(3):311–340, 2000.
 - [40] Carsten Riggelsen. MCMC learning of Bayesian Network models by Markov blanket decomposition. In J. Gama et al., editors, *Proceedings of the 16th European Conference on Machine Learning*, volume 3720 of *Lecture Notes in Computer Science*, pages 329–340, Berlin Heidelberg, 2005. Springer-Verlag.
 - [41] B. Sierra and P. Larrañaga. Predicting survival in malignant skin melanoma using Bayesian Networks automatically induced by Genetic Algorithms. an empirical comparison between different approaches. *Artificial Intelligence in Medicine*, 142:215–230, 1998.
 - [42] M. Studený. *Probabilistic Conditional Independence Structures*. Springer, 2005.
 - [43] S. van Dijk and D. Thierens. On the use of a non-redundant encoding for learning Bayesian Networks from data with a GA. In X. Yao et al., editors, *Parallel Problem Solving from Nature VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 141–150, Berlin Heidelberg, 2004. Springer-Verlag.
 - [44] M.L. Wong, W. Lam, and K.S. Leung. Using Evolutionary Programming and Minimum Description Length principle for data mining of Bayesian Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):174–178, 1999.
 - [45] M.L. Wong, S.Y. Lee, and K.S. Leung. Data mining of Bayesian networks using cooperative coevolution. *Decision Support Systems*, 38:451–472, 2004.
 - [46] M.L. Wong and K.S. Leung. An efficient data mining method for learning Bayesian Networks using an evolutionary algorithm-based hybrid approach. *IEEE Transactions on Evolutionary Computation*, 8(4):378–404, 2004.