A Primer on the Evolution of Equivalence Classes of Bayesian-Network Structures

Jorge Muruzábal[1] and Carlos Cotta[2]

[1] Grupo de Estadística y Ciencias de la Decisión, ESCET,
University Rey Juan Carlos, 28933 - Móstoles, SPAIN
j.muruzabal@escet.urjc.es

[2] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, SPAIN
ccottap@lcc.uma.es

Abstract. Bayesian networks (BN) constitute a useful tool to model the joint distribution of a set of random variables of interest. To deal with the problem of learning sensible BN models from data, we have previously considered various evolutionary algorithms for searching the space of BN structures directly. In this paper, we explore a simple evolutionary algorithm designed to search the space of BN *equivalence classes*. We discuss a number of issues arising in this evolutionary context and provide a first assessment of the new class of algorithms.

1 Introduction

A Bayesian Network (BN) is a graphical model postulating a joint distribution for a set of discrete random variables. Critical qualitative aspects in this model are given by the underlying graphical structure, a *directed acyclic graph* (DAG) $\mathbf{G}$; quantitative aspects are provided by the set of marginal and conditional probabilities attached to this DAG, say $\boldsymbol{\theta} = \boldsymbol{\theta}(\mathbf{G})$. To deal with the problem of learning sensible BN models from data (a problem known to be $NP-$hard), a number of evolutionary algorithms (EAs) have been considered to search the space of DAG structures or *b-space*, see e.g. [1–3]. Just like other *score-and-search* methods [4], DAG structures $\mathbf{G}$ are often evolved according to some standard scoring measure based on the data. Promising results have been obtained in general following this evolutionary approach in b-space.

Two DAGs are (*Markov) equivalent* if they encode the same statistical model, that is, the same set of independence and conditional independence statements. If we denote the equivalence class containing $\mathbf{G}$ as $[\mathbf{G}]$, each $[\mathbf{G}]$ corresponds to a different statistical model (our true object of interest). Hence, as DAGs can be meaningfully grouped together in equivalence classes, we can also consider the alternative *e-space*, the space of equivalence classes, as a more direct target [5]. This strategy will be useful, of course, provided that we can traverse the more complex e-space in some computationally efficient manner. It must be noted that e-space is known to be not much smaller than b-space [6], and the computational load in the former may be somewhat heavier. While the issue seems far from settled at this point (see the concluding section), in this paper we follow what we think is an appealing framework for initiating the investigation of evolutionary search in e-space.

It turns out that equivalence classes $[\mathbf{G}]$ can be compactly represented by certain class of *partially* directed acyclic graphs or PDAGs, see e.g. [7]. PDAGs may include directed as well as *undirected* arcs. Let $\bar{\mathbf{G}}$ denote the *unique* PDAG structure representing some $[\mathbf{G}]$. Then $\bar{\mathbf{G}}$ and all $\mathbf{H} \in [\mathbf{G}]$ share the same connectivity pattern (ignoring directionality), and all directed arcs in $\bar{\mathbf{G}}$ show up in every $\mathbf{H} \in [\mathbf{G}]$. A complicating factor is that not all PDAGs represent equivalence classes, only *completed* PDAGs (CPDAGs) do. On this matter, Chickering [7] presents various operators designed to modify a given CPDAG $\bar{\mathbf{G}}$ (representing $[\mathbf{G}]$) so that the resulting PDAG $\bar{\mathbf{H}}$ represents (after being completed) some other $[\mathbf{H}] \neq [\mathbf{G}]$. Templates are also provided for calculating the corresponding change in score after the modification is done.

The design of efficient learning algorithms in e-space can thus be assisted by using the CPDAG space, and building on Chickering's basic results. We explore below what we believe is the first EA designed to adopt exactly this approach. The key operators look much like *mutations*, so we have adopted an *evolutionary programming* (EP) approach as the natural paradigm to get started [2, 8].

2 Background

This section provides basic ideas and notational details. We first introduce the central BN framework, then continue with the learning paradigm based on equivalence classes.

2.1 Bayesian Networks

A Bayesian Network is a tuple $(\mathbf{G}, \boldsymbol{\theta})$, where $\mathbf{G}$ is a Directed Acyclic Graph (DAG) and $\boldsymbol{\theta} = \boldsymbol{\theta}(\mathbf{G})$ is a set of probability distributions attached to $\mathbf{G}$. The DAG is the set of links or *arcs* among variables or *nodes*. If we denote the whole set of variables as $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, each X_i has a set of *parents*, denoted by $\Pi_i = \{X_j \in \mathbf{X} \mid (X_j \to X_i) \in \mathbf{G}\}$, and the DAG $\mathbf{G}$ represents the joint distribution $P(\mathbf{X})$ =$P(\mathbf{X}|\mathbf{G}) = \prod_{i=1}^{n} P(X_i \mid \Pi_i)$, where $\Pi_i = \emptyset$ at least once. A standard parametric model arises when this $P(\mathbf{X})$ is viewed as a family of distributions indexed by $\boldsymbol{\theta} = \boldsymbol{\theta}(\mathbf{G})$. In particular, it is often assumed that variables follow (conditionally) independent Multinomial distributions, that is, $P(X_i = k \mid \Pi_i = j) = \theta_{ijk}$, where $j = 1, ..., q_i$; $k = 1, ..., r_i$; 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. Hence, $\boldsymbol{\theta} = \{\theta_{ijk}\}$ collects all parameters in $\mathbf{G}$ with the constraint that $\sum_k \theta_{ijk} = 1$ for all (i, j).

A crucial issue is how to evaluate DAG structures. Given a standard Multinomial likelihood $P(\mathbf{D}|\mathbf{G}, \boldsymbol{\theta})$, estimates $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}(\mathbf{D})$ are usually based on the sufficient count statistics $\mathbf{N} = \{N_{ijk}\}$ (interpreted as $\boldsymbol{\theta}$). For example, the maximum likelihood (frequentist) approach leads to $\hat{\theta}_{ijk} = N_{ijk}/N_{ij}$, where $N_{ij} = \sum_k N_{ijk}$. The (intra-network) Bayesian approach assumes a *prior* density on parameter space, say $\pi(\boldsymbol{\theta}|\mathbf{G})$, and uses it to compute the *marginal likelihood* $P(\mathbf{D}|\mathbf{G}) = \int P(\mathbf{D}|\mathbf{G}, \boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathbf{G})d\boldsymbol{\theta}$. This integration will be difficult in general, but it is possible analytically (under certain assumptions) if $\pi(\boldsymbol{\theta}|\mathbf{G}) = \prod_{i,j} \pi(\theta_{ij})$, where θ_{ij} denotes the vector containing the r_i probabilities θ_{ijk}, $\pi(\theta_{ij}) \propto \prod_k \theta_{ijk}^{\alpha_{ijk}-1}$, and $\boldsymbol{\alpha} = \{\alpha_{ijk}\}$ is the *virtual count* or Dirichlet hyperparameter ($\alpha_{ijk} > 0$), see [9]. We adopt this criterion in what follows. Since $\boldsymbol{\alpha}$ must be supplied by the user just like the complete data set **D,** we write $\Psi(\mathbf{G}|\mathbf{D}, \boldsymbol{\alpha}) = \log P(\mathbf{D}|\mathbf{G})$ as our basic (standard) scoring metric in b-space.

2.2 Learning Equivalence Classes of BN Models

As mentioned earlier, an equivalence class of DAGs contains all structures yielding exactly the same set of independence and conditional independence statements. For example, DAGs $X \rightarrow Y \rightarrow Z$ and $X \leftarrow Y \rightarrow Z$ are equivalent, for they both express that X is conditionally independent of Z given Y. Equivalence classes $[\mathbf{G}]$ can be compactly represented by the class of completed partially directed acyclic graphs or CPDAGs [5, 7]. 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 CPDAG $\bar{\mathbf{G}}$ representing $[\mathbf{G}]$ contains a directed arc for each compelled arc in $[\mathbf{G}]$ and an undirected arc for each reversible arc in $[\mathbf{G}]$. In our previous example, the CPDAG $X - Y - Z$ represents the equivalence class containing $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \rightarrow Z$ as well as $X \leftarrow Y \leftarrow Z$. However, $X \rightarrow Y \leftarrow Z$ belongs to a different equivalence class, namely, that represented by the CPDAG containing the *v-structure* (or "inmorality") $X \rightarrow Y \leftarrow Z$. It follows that providing arbitrary directionality to reversible arcs in a CPDAG $[\mathbf{G}]$ does not necessarily result in a member of $[\mathbf{G}]$. On the contrary, very specific algorithms must be used to obtain a correct DAG from a given CPDAG and *vice versa* (see [7] and below).

With regard to some other research [4, 9, 10], Chickering's approach introduces a clear semantics and up to six different operators for performing local variation in existing CPDAGs. These operators can be scored locally, and score-updating formulae are provided for them. Hence, search algorithms can traverse much faster through different equivalence classes [7]. The basic operators are termed `InsertU`, `DeleteU`, `InsertD`, `DeleteD`, `ReverseD` and `MakeV`. The first four either increase or reduce the number of arcs by one, the rest maintain the number of arcs in the current CPDAG. Specifically, the fifth reverses the directionality of a single directed arc, whereas the sixth transforms the substructure $X - Y - Z$ (where X is not linked to Z directly) into the v-structure $X \rightarrow Y \leftarrow Z$.

When locally manipulating a given CPDAG $\bar{\mathbf{G}}$ in these ways, the resulting PDAG may not be initially completed. In essence, we could use the basic PDAG-to-DAG routine [7] to find out if any proposed operation is *valid*: if this routine failed to return a proper DAG structure, say $\mathbf{H}$, then the operation can not be carried out. Otherwise the operation would be valid, and we could call the basic DAG-to-CPDAG routine (with input $\mathbf{H}$) to determine the resulting $\bar{\mathbf{H}} \neq \bar{\mathbf{G}}$. In practice, each operator comes with its own *validity test*, a compact set of conditions to check that $\mathbf{H}$ exists in each case [7]. Note that there may be "cascading" implications in this process; for example, `DeleteD` may make other directed arcs undirected. Or, after applying `MakeV`, many arcs may switch from undirected to directed.

Chickering also provides the corresponding change in score after the modification is done [7]. Here it makes sense to use a basic DAG scoring metric which is *score-equivalent*, that is, constant over each equivalence class. Many familiar measures are score-equivalent — the present $\Psi(\mathbf{G}|\mathbf{D}, \boldsymbol{\alpha}) = \log P(\mathbf{D}|\mathbf{G})$ may or may not be depending on $\boldsymbol{\alpha}$. Let $\alpha_{ij} = \sum_k \alpha_{ijk}$ and $\alpha_i = \sum_j \alpha_{ij}$. Heckerman, Geiger and Chickering [9] show that $\Psi(\mathbf{G}|\mathbf{D}, \boldsymbol{\alpha})$ is score-equivalent if $\alpha_i \equiv \alpha$ for some $\alpha > 0$ (the BDe metric). Parameter α reflects strength of belief in the proposed priors for the θ_{ijk}. We consider below the well-known BDeu(α) metric $\alpha_{ijk} = \alpha / r_i q_i$. Another typical option is the K2 metric $\alpha_{ijk} = 1$, but this is not score-equivalent.

Once an initial CPDAG structure is evaluated, we can update the score via Chickering's results. The key idea behind this local scoring is that a *decomposable* scoring function Ψ — making use of local evaluations only — is typically used (for example, both K2 and BDeu(α) are decomposable). That is, 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 case to a single node X_i and its parents Π_i. To illustrate, the change in score attributed to a particular valid mutation deleting $X \to Y$ in $\bar{\mathbf{G}}$ and leading to some $\bar{\mathbf{H}}$ can be expressed as $\Psi(\bar{\mathbf{H}}) - \Psi(\bar{\mathbf{G}}) = \sigma(Y,\ \Lambda_1) - \sigma(Y,\ \Lambda_2)$, 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.

3 Evolutionary Framework

A number of NP results motivate the use of heuristic methods for the problem of learning BNs from data [7]. Score-and-search methods are relatively simple to use and constitute a major alternative to locate sensible models. In particular, a number of evolutionary algorithms have been proposed (see e.g., [1–3] and the references therein). In essence, these evolutionary approaches can be classified within two main categories: *direct* and *indirect*. Direct approaches are those in which the search is conducted over the space of all possible DAGs (b-space). Indirect approaches use an auxiliary space to conduct the search. Selected elements from this auxiliary space are fed to a suitable (decoder) algorithm to obtain the actual BNs they represent. The algorithm considered in this work falls within the first category, although with the twist that the search is conducted over the space of all possible CPDAGs (e-space).

The first issue to be tackled is the choice of evolutionary paradigm to work with. The graph modifications discussed in Sect. 2.2 can be seen as natural counterparts of the familiar mutation operators found in the evolutionary DAG arena, see e.g. [2]. Thus, our basic algorithm is currently of evolutionary programming (EP) nature (e.g., see [8, 11]). We begin with a population of P CPDAGs. At each generation, members of the population are selected by means of binary tournament to produce mutated structures (i.e., different CPDAGs), which we locally evaluate and store. Finally, the best P out of the $2P$ available structures are selected for the next generation (the rest are discarded) and a new mutation sweep takes place. We simply let this process continue for T generations in all experiments below. Initial CPDAGs are generated by the DAG-to-CPDAG routine on either randomly or heuristically constructed DAGs (by applying the K2 heuristic using a random permutation of the variables as seed). In the first case, the initialization process is parameterized by $\delta \in [0,1]$, the arc density of the random graph [3]; in the second case, it is controlled by $\pi_{\max}$, the maximum number of parents per variable in the initial population. In either case, each DAG is evaluated by our basic scoring metric $\Psi(\mathbf{G}|\mathbf{D},\boldsymbol{\alpha}) = \log P(\mathbf{D}|\mathbf{G})$ and the result passed to the associated CPDAG.

There are many interesting research questions in this scenario — we provide some discussion and empirical evidence now and we discuss some further issues later. Two key roles that we do investigate are those played by the fitness BDeu(α) metric and the various mutation operators. More specifically, what is the effect of α in the EP process? Are all six operators needed for best operation? If so, how to decide which operator ω will act on a selected CPDAG $\bar{\mathbf{G}}$?

At the moment, we try to gain some familiarity with this environment by analyzing a simple (default) variant. Specifically, mutation operators ω are selected by (independent) drawings from the uniform probability distribution Ω over the whole set of available operators. Note that some operators ω 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. Since the mutated CPDAG $\bar{\mathbf{H}} = \omega(\bar{\mathbf{G}})$ may not pass the corresponding validity test, we monitor the operators' *validity* ratio $\upsilon = \upsilon(\omega)$. If the mutated CPDAG is valid, it is incorporated to the offspring population. We also 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 believe some knowledge about these basic aspects of performance should be acquired before we can embark into more elaborate designs.

4 Experimental Results

The basic algorithm described above has been deployed on the familiar ALARM network, a 37-variable network for monitoring patients in intensive care [12]. The equivalence class [ALARM] is represented by a CPDAG with 4 undirected and 42 directed arcs. A training set of $N = 10,000$ examples was created once by random probabilistic sampling as customary. The BDeu(α) metric $\Psi(\mathbf{G}|\mathbf{D},\boldsymbol{\alpha}) = \log P(\mathbf{D}|\mathbf{G})$ is the fitness function (to be maximized). All experiments have been performed using a population size of $P = 100$ individuals and $T = 500$ generations (i.e., 50,000 individuals generated). Five different initialization settings have been considered: on one hand, random initialization using density values

Table 1. Results (averaged for ten runs) of the EA using a different set of operators. Random initialization with parameter δ is denoted by R_δ, and heuristic initialization with parameter $\pi_{\max}$ is denoted as $H_{\pi_{\max}}$. U and D indicate the average number of undirected and directed arcs in the final population.

	Basic Set				Basic Set $\cup$ {`ReverseD`}			
	best	mean $\pm$ SD	U	D	best	mean $\pm$ SD	U	D
$R_{0.05}$	-107006	-107499 ± 516	45.23	19.03	-106519	-106899 ± 357	33.79	23.30
$R_{0.10}$	-106680	-107236 ± 401	39.03	23.49	-106707	-107099 ± 470	35.14	24.85
H_2	-106656	-106933 ± 252	31.31	25.40	-106532	-106681 ± 113	28.07	25.00
	Basic Set $\cup$ {`MakeV`}				Basic Set $\cup$ {`ReverseD`, `MakeV`}			
	best	mean $\pm$ SD	U	D	best	mean $\pm$ SD	U	D
$R_{0.01}$	-107308	-108108 ± 533	6.34	56.73	-106506	-107039 ± 324	7.49	46.60
$R_{0.05}$	-106994	-108191 ± 559	8.02	59.82	-106621	-107045 ± 280	10.69	47.68
$R_{0.10}$	-107186	-108281 ± 820	6.22	59.45	-106703	-107175 ± 351	8.82	51.93
H_1	-106594	-106915 ± 267	9.59	45.58	-106503	-106723 ± 232	7.68	43.65
H_2	-106591	-106867 ± 208	7.57	46.17	-106503	-106602 ± 209	7.96	39.31

$\delta \in \{0.01, 0.05, 0.10\}$; on the other, heuristic initialization using a greedy (K2) heuristic with maximum number of parents per variable $\pi_{\max} \in \{1, 2\}$.

The initial experiments aim to assess the operators as follows. We have considered a *basic* set of operators for traversing the space of CPDAGs, namely `InsertD`, `DeleteD`, `InsertU`, and `DeleteU`. Subsequently, we have tested the addition of `ReverseD` and/or `MakeV`, implying a total of four operator sets. The results are shown in Table 1. In all cases, the BDeu($\alpha = 1$) metric has been used. To put these results in perspective, the fitness of the original network is -106587.

Note firstly the essential role played by `MakeV`: by inspecting the networks in the final populations, it can be seen that a high number of undirected arcs arise when this operator is not available. The injection of v-structures appears thus crucial for balancing the adequate proportion of directed and undirected arcs. Actually, whenever the population is initialized with low-density networks ($R_{0.01}$ or H_1), the lack of `MakeV` makes all directed arcs vanish in a few iterations. Besides turning `DeleteD` and `ReverseD` inapplicable, the population achieves in this situation a degenerate state of high-density undirected networks. For this reason, we have not included these low-density initial conditions in the experiments omitting `MakeV`. Note also that results are best when using `MakeV` together with `ReverseD`. Indeed, the best network found in this case just differs from the original network in a directed arc of the latter that is substituted by a different undirected arc in the former. The `ReverseD` operator seems also very important in practice. Here, it seems to help the networks size down appropriately.

Regarding the properties of the operators, consider first the validity ratio $\upsilon = \upsilon(\omega)$. We note that this ratio is close to 1 for `InsertU` and `DeleteU`. For `DeleteD` and `ReverseD`, it raises from 0.5 up to 1 in about 100 generations, then stays at that level for the rest of the run. For `InsertD`, it oscillates between 0.8 and 1. Finally, `MakeV` exhibits the lowest υ ratio (oscillating between 0.5 and

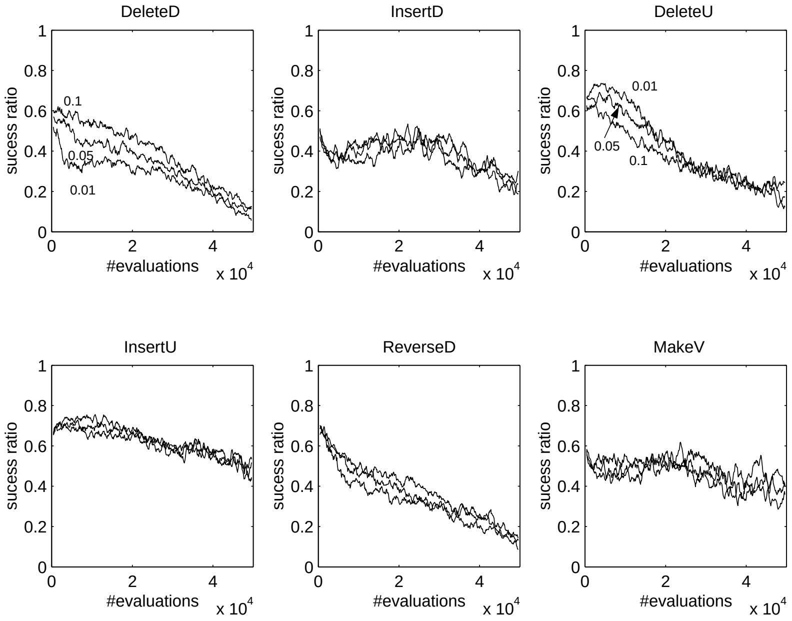

Fig. 1. Mean success ratio (averaged for ten runs) of each operator when using random initialization.

1). As refers to success ratios, consider the trajectories shown in Fig. 1. There exists naturally a general decreasing trend (improvements are less frequent in the latter stages of the run). Also for this reason, lower success levels (not shown) are obtained when using heuristic initialization (the algorithm performs its run at a higher fitness level). The overall aspect of the trajectories is nevertheless the same in this latter case. The different decreasing rates shed some light on the relative contribution of operators though. In particular, note that the undirected-arc-based, "brick and mortar" operators `InsertU` and `MakeV` tend to maintain the highest success ratios. On the other hand, `DeleteD` , `ReverseD` and `DeleteU` are the ultimately least useful. This suggests that the evolved structures manage to consolidate some useful set of both compelled and reversible arcs.

Using the full set of operators, our second set of experiments intends to test the influence of the BDeu parameter α. For this purpose, we compare the difference between the best structure found in each run and the original ALARM CPDAG. The values $\alpha \in \{0.01, 0.1, 1, 10\}$ have been tried, see Table 2. A first inspection indicates that there are some trends that can be appreciated in this range. For the randomly initialized runs, the number of shared (*true*) arcs of either kind tends to decrease for larger values of α. For the heuristically initialized runs, this is only true for the undirected arcs. On this matter, Kayaalp and Cooper [13] show that, for big enough sample size N, arcs are more likely to be incorporated into the network for larger values of α. Actually, the average number of arcs (either directed or undirected) for each value of α is 48.88, 49.84,

Table 2. Networks generated by the EA for different values of α. S_U and S_D stand for shared arcs (undirected or directed), i.e., arcs present in both the original and the evolved networks). The number of arcs in the evolved network is denoted as n_{arcs}. All results are the average of ten runs.

	$\alpha = 0.01$			$\alpha = 0.1$			$\alpha = 1$			$\alpha = 10$		
	S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}	S_U	S_D	n_{arcs}
$R_{0.01}$	3.3	32.9	48.0	3.0	36.1	49.1	2.2	30.9	54.0	1.5	29.5	61.1
$R_{0.05}$	3.1	32.6	49.3	2.8	33.0	51.3	2.2	28.9	56.5	1.3	26.9	67.6
$R_{0.10}$	2.4	29.4	53.5	1.8	30.1	54.9	1.6	28.5	59.1	1.3	25.1	70.8
H_1	3.8	34.6	46.7	3.6	36.8	47.4	3.7	35.0	51.3	1.5	37.8	50.9
H_2	3.8	33.9	46.9	4.0	37.5	46.5	4.0	36.4	48.0	1.1	34.6	53.5

53.78, and 60.78 respectively, an uprising pattern. The simple $\alpha = 1.0$ seems to provide the best overall results. For example, the best network (according to BDeu) found for $\alpha = 0.1$ captures all 4 original undirected arcs, and 39 original directed arcs from the ALARM structure. As to $\alpha = 10$, just 1 undirected and 37 directed arcs are reflected. By contrast, 4 undirected arcs and 41 directed arcs are captured in the best run for $\alpha = 1$. This level of performance is comparable to that achieved by the state-of-the-art algorithms [7, 14].

5 Discussion and Further Developments

We have considered a basic EP algorithm based on equivalence classes (EPQ say) and studied a number of performance issues in a novel evolutionary framework based on equivalence classes represented as CPDAGs. We have observed the adequacy of having all six operators introduced by Chickering [7] cooperating together in order to achieve satisfactory results in the benchmark ALARM problem. We have also illustrated the relatively low sensitivity of the evolved structures with regard to the α scaling of the BDeu fitness metric. Our results are thus encouraging albeit preliminary, for there are some key issues still in dispute in the broader machine learning context. We now briefly discuss some of these issues, and the implications for our current EPQ algorithm.

Castelo and Kočka [14] highlight the relevance of the *inclusion boundary* (IB) principle in graph learning. This principle stresses the need for the traverse set of operators to be able to produce *any* model in some "tight" neighbourhood of the current model. A number of algorithms (backed by some useful convergence results holding under reasonable sampling assumptions) have been proposed and tested with success [14–16]. As it turns out, the desired connectivity is given by the *ENR* (equivalence-no-reversals) neighbourhood of a DAG $\mathbf{G}$: the set of all DAGs that can be reached from any DAG $\mathbf{H} \in [\mathbf{G}]$ by (valid) single directed arc addition or deletion. This neighbourhood might appear to be relatively simpler to implement in b-space [14] than in e-space [15]. Indeed, our traverse set made up by the six operators proposed in [7] does not verify the IB principle since some DAGs in ENR may not be reachable (although many other DAGs not in ENR can be covered). Chickering [15] has subsequently introduced (more complex) `Insert` and `Delete` operators respecting the IB principle (covering ENR). Thus, a first line for future work would address the tradeoff between simplicity and ENR-coverage towards the design of more efficient traverse operators in e-space [16]. On the other hand, we have already begun the evaluation of EPQ with regard to other inclusion-driven EP algorithms in b-space.

Moving beyond this overview of the general learning scenario, we would also like to furnish some prospects for future evolutionary developments based on the present EPQ. First of all, there is a very interesting research line in connection with the Ω distribution used for selecting operators. We would like to tackle the design of strategies for dynamically adapting Ω. In this sense, we envision both adaptive approaches — in which some population-wide rules are used to modify Ω during the run —, and self-adaptive approaches — in which each individual l carries its own Ω_l distribution (whose population is also subject to evolution). Some lessons from the utilization of these mechanisms in the field of *evolution strategies* can be used for this purpose [17].

Recombination operators are also in perspective. We face here a difficult scenario for recombination, for not only acyclicity constraints apply (as in the case of DAGs), but also additional constraints due to the nature of CPDAGs. Of course, we could instantiate $\bar{\mathbf{G}}$ and $\bar{\mathbf{H}}$ (using the PDAG-to-DAG routine) to obtain DAGs $\mathbf{G}$ and $\mathbf{H}$, use standard operators over DAGs as in [3] to produce some DAG $\mathbf{K}$, and finally apply the DAG-to-CPDAG routine to obtain an offspring $\bar{\mathbf{K}}$ derived from $\bar{\mathbf{G}}$ and $\bar{\mathbf{H}}$. However, we could not guarantee that $\bar{\mathbf{K}} \neq \bar{\mathbf{G}}$ and $\bar{\mathbf{K}} \neq \bar{\mathbf{H}}$. To deal with this, we have in mind the decomposition of the recombination process in a sequence of basic operations assimilable to the mutation operators we have used, much in the line of what is done in path-relinking [18]. This and the basic EP approaches can be completed with the incorporation of *phenotypic* measures based on the availability of local scoring measures, e.g. [3].

We have used a variant of BDe or log $P(\mathbf{D}|\mathbf{G})$ as our basic scoring metric. An alternative is provided by the log posterior $\Psi(\mathbf{G}|\mathbf{D}, \boldsymbol{\alpha}) = \log \pi(\mathbf{G}|\mathbf{D}) = \log \pi(\mathbf{G}) + \log P(\mathbf{D}|\mathbf{G})$, where $\pi(\mathbf{G})$ is some prior on DAG structures [9]. For example, letting $g = \sum_i (r_i - 1) q_i$, the number of free parameters in $\boldsymbol{\theta}$, $\pi(\mathbf{G}) \propto N^{-g/2}$ penalizes complex structures (as in our previous work [3]). These ideas may be useful to counter-balance biases towards denser networks that could appear when working with BDeu [13].

Acknowledgement. We are grateful to D.M. Chickering and R. Castelo for some useful conversations on these ideas, and to Alicia Puerta and Quique López for valuable assistence. The authors are partially supported by the DMR Foundation's Decision Engineering Lab and by grants TIC2001-0175-C03-03 and TIC2002-04498-C05-02 from the Spanish MCyT.

References

1. Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R., Kuijpers, C.H.: Structure learning of bayesian networks by genetic algorithms: A performance analysis of con-

trol parameters. IEEE Transactions on Pattern Analysis and Machine Intelligence **10** (1996) 912–926

2. Wong, M., Lam, W., Leung, K.: Using evolutionary programming and minimum description length principle for data mining of bayesian networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999) 174–178
3. Cotta, C., Muruzábal, J.: Towards a more efficient evolutionary induction of bayesian networks. In Merelo, J., Adamidis, P., Beyer, H.G., Fernández-Villacañas, J.L., Schwefel, H.P., eds.: Parallel Problem Solving From Nature VII. Volume 2439 of Lecture Notes in Computer Science. Springer-Verlag, Berlin (2002) 730–739
4. Heckerman, D.: A tutorial on learning with bayesian networks. In Jordan, M., ed.: Learning in Graphical Models. Kluwer, Dordrecht (1998) 301–354
5. Andersson, S., Madigan, D., Perlman, M.: A characterization of markov equivalence classes for acyclic digraphs. Annals of Statistics **25** (1997) 505–541
6. Gillespie, S., Perlman, M.: Enumerating Markov equivalence classes of acyclic digraph models. In Goldszmidt, M., Breese, J., Koller, D., eds.: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, Seatle WA, Morgan Kaufmann (2001) 171–177
7. Chickering, D.: Learning equivalence classes of Bayesian-network structures. Journal of Machine Learning Research **2** (2002) 445–498
8. Fogel, L., Owens, A., Walsh, M.: Artificial Intelligence Through Simulated Evolution. Wiley, New York NY (1966)
9. Heckerman, D., Geiger, D., Chickering, D.: Learning bayesian networks: the combination of knowledge and statistical data. Machine Learning **20** (1995) 197–243
10. Acid, S., de Campos. L.M.: Searching for bayesian network structures in the space of restricted acyclic partially directed graphs. Journal of Artificial Intelligence Research **18** (2003) 445–490
11. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin Heidelberg (2003)
12. Beinlich, I., Suermondt, H., Chavez, R., Cooper, G.: The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In Hunter, J., Cookson, J., Wyatt, J., eds.: Proceedings of the Second European Conference on Artificial Intelligence and Medicine, Berlin, Springer-Verlag (1989) 247–256
13. Kayaalp, M., Cooper, G.: A bayesian network scoring metric that is based on globally uniform parameter priors. In Darwiche, A., Friedman, N., eds.: Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence, San Francisco CA, Morgan Kaufmann (2002) 251–258
14. Castelo, R., Kočka, T.: On inclusion-driven learning of bayesian networks. Journal of Machine Learning Research **4** (2003) 527–574
15. Chickering, D.: Optimal structure identification with greedy search. Journal of Machine Learning Research **3** (2002) 507–554
16. Nielsen, J., Kočka, T., Peña, J.: On local optima in learning bayesian networks. In Rulff, U., Meek, C., eds.: Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, Acapulco, Mexico, Morgan Kaufmann (2003) 435–442
17. Beyer, H.G.: Toward a theory of evolution strategies: Self adaptation. Evolutionary Computation **3** (1996) 311–347
18. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **39** (2000) 653–684