Learning an L1-Regularized Gaussian Bayesian Network in the Equivalence Class Space

Diego Vidaurre, Concha Bielza, and Pedro Larrañaga

***Abstract*—Learning the structure of a graphical model from data is a common task in a wide range of practical applications. In this paper, we focus on Gaussian Bayesian networks, i.e., on continuous data and directed acyclic graphs with a joint probability density of all variables given by a Gaussian. We propose to work in an equivalence class search space, specifically using the *k*-greedy equivalence search algorithm. This, combined with regularization techniques to guide the structure search, can learn sparse networks close to the one that generated the data. We provide results on some synthetic networks and on modeling the gene network of the two biological pathways regulating the biosynthesis of isoprenoids for the *Arabidopsis thaliana* plant.**

***Index Terms*—Equivalence class, gene networks, graphical Gaussian model, *k*-greedy equivalence search (GES), Lasso, microarrays, network induction, regularization.**

I. Introduction

A GAUSSIAN Bayesian network (GBN) [1] is a probabilistic graphical model that encodes a joint Gaussian density $[f(\boldsymbol{X})]$ on a p-dimensional random variable $\boldsymbol{X} = (X_1, \ldots, X_p)$

$$f(\boldsymbol{x}) \equiv \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{1}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^{\mathrm{T}}$ is the vector of means and $\boldsymbol{\Sigma}$ is the $p \times p$ covariance matrix. In a GBN, the density function of the joint distribution can be expressed as the product of p univariate normal densities defined as

$$f_i\left(x_i|\boldsymbol{pa}(x_i)\right) \equiv N\left(m_i + \sum_{x_j \in \boldsymbol{pa}(x_i)} \beta_{ji}(x_j - m_j), v_i\right). \tag{2}$$

The variance is v_i, and the mean is composed of subparameters m_i and $\boldsymbol{\beta_i} = (\beta_{1i}, \ldots, \beta_{li})^{\mathrm{T}}$, where l is the number of parents of variable X_i, denoted by $\boldsymbol{Pa}(X_i)$. β_{ji} is the linear regression coefficient of X_j in the regression of X_i on $\boldsymbol{Pa}(X_i)$. It represents how strong the relationship between X_i and X_j is; if $\beta_{ji} = 0$, then X_j will not be a parent of X_i. Multivariate normal density and the product of normal densities given in (2) are alternative and interchangeable representations [2].

There are two basic approaches to GBN structure learning from data: algorithms based on constrained methods and *score+search* methods. Constraint-based approaches build networks that fulfill the conditional independences estimated from data [3]. The conditional independences between variables are checked by means of statistical tests. A recent method of this kind was developed by Margaritis [4].

On the other hand, score+search algorithms are founded on a scoring function for network evaluation in an attempt to find the model that best fits the data. The methods suggested in [5] and [6] are two examples of score+search algorithms. A representation, a scoring function, and a search strategy have to be defined.

First, to represent the solutions and move in the search space, we typically choose between directed acyclic graphs (DAGs), partial DAGs (PDAGs), or variable orderings. Variable orderings are an intermediate representation that must be mapped to a graph to be meaningful. An equivalence class, modeled by a PDAG [2], is the set of graphs with the same conditional independences, encoding a unique probability density. Equivalence classes are often the preferred representation. They are the only representation that is capable of meeting the inclusion boundary (IB) requirement [7], [8] described hereinafter. In the PDAG, compelled arcs (arcs with the same orientation for all the members in the class) are modeled by directed arcs; the others are represented by undirected edges.

Second, assuming that we have a data set $\mathcal{D}$ of size n, to measure how well the model fits the data, the likelihood of the model given the data is defined as

$$L\left(\mathcal{D}; \boldsymbol{v}, \boldsymbol{m}\right) = \prod_{i=1}^{p}\prod_{r=1}^{n} \frac{1}{\sqrt{2\pi v_i}} e^{-\frac{1}{2v_i}\left(x_{ir} - m_i - \sum_{x_j \in \boldsymbol{pa}(x_i)} \beta_{ji}(x_{jr} - m_j)\right)^2}. \tag{3}$$

A penalized scoring criterion, made up of the likelihood function and a penalization term that favors simple models, is usually employed. An example of such a criterion is minimum description length (MDL), which is defined in Section III. A scoring criterion is score equivalent if it returns the same value for all the members inside the class. Sometimes, a unique orientation for all arcs is needed to provide an exact causal semantic to the network. When the density function itself, without causality implications, is of interest, to be score equivalent is a positive property.

Finally, the third element is the search strategy. Whether it is better to search in equivalence class spaces or, alternatively,

Manuscript received March 25, 2009; revised August 15, 2009 and October 26, 2009; accepted October 26, 2009. Date of publication January 15, 2010; date of current version September 15, 2010. This work was supported in part by the Spanish Ministry of Science and Innovation under Projects TIN2007-62626 and TIN2008-06815-C02-02, in part by Consolider Ingenio 2010-CSD2007-00018, and by the Cajal Blue Brain Project. This paper was recommended by Associate Editor R. Lynch.

The authors are with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 28660 Madrid, Spain.

Digital Object Identifier 10.1109/TSMCB.2009.2036593

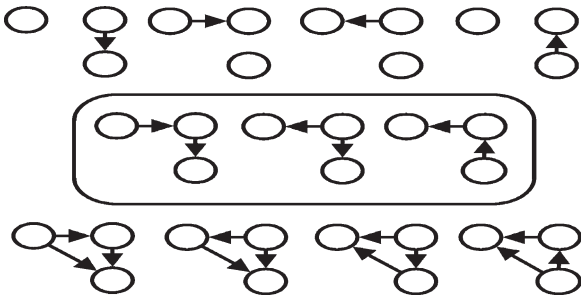


Fig. 1. Equivalence class for three nodes. Around it, its inclusion boundary, (top) dropping an arc and (bottom) adding an arc.

in DAG spaces is still an open question. Some researchers think that it is not always worthwhile to work with equivalence classes because it is more complex. Gillispie and Perlman [9] analyzed the expected number of DAGs inside an equivalence class. They found that, on average, this number is not big. This could be an argument in favor of DAG spaces. However, the variance is high, and potentially, there could be a huge number of DAGs inside some equivalence classes, even when we are trying to achieve sparse graphs. For example, a tree of p vertices gives rise to a class of size p, whereas the equivalence class contains $p!$ DAGs in the case of the complete DAG (worst case) [10]. For this reason, it is often a good idea to work with equivalence classes because they are a more efficient and robust representation. This representation is more able to deal with situations involving equivalence classes of high cardinality.

Three main problems arise when working in the DAG space rather than with equivalence classes. First, some operators defined to move between DAGs may operate between graphs in the same equivalence class. This is a waste of time, unless it is checked by the algorithm. Therefore, if we work with a score-equivalent criterion, these moves do not change the overall network fitness. Second, the move from the current equivalence class might not be the best if we ignore the equivalence class space and analyze only the neighborhood of just one of its DAGs. Given that all members of a class score the same value (assuming a score-equivalent criterion), there is no reason other than randomness to prefer a specific DAG member. The third problem is related to how likely the final DAG is to belong to a specific equivalence class. If all DAGs in the same class are interpreted as different models, it is reasonable to expect the final output to be an equivalence class covering many rather than just a few DAGs. This makes it unpredictable.

A model is defined as *inclusion optimal* with regard to a density $f(\boldsymbol{X})$ if it is able to represent the density with a minimum number of arcs. The IB concept [7] is defined as a neighborhood of a model where the search strategy selects new models. Intuitively, the IB can be defined as the union set of equivalence classes that can be reached from each DAG inside the current class by single arc addition or deletion (see Fig. 1). The *greedy equivalence search* (GES) algorithm for Bayesian network learning [11] uses the IB. It has appealing theoretical properties for finding inclusion optimal models (see [8] and [12] for more details). The k-GES (KES) algorithm [13] generalizes GES and respects the IB too. Hence, it retains the same theoretical properties. KES features a stochastic factor. This way, multiple runs can be made to extract common patterns in the solutions, whereby the final model includes just the arcs that showed up in most of the runs.

In this paper, we extend the KES algorithm to continuous (Gaussian) densities and learn sparser models based on regularization techniques. By definition, a GBN defines a regression for each variable X_i over its parents [see (2)]. Supposing standardized data (so that $m_i = 0$ and $v_i = 1$), it is defined as

$$x_i = \sum_{x_j \in \boldsymbol{pa}(x_i)} \beta_{ji} x_j + w_i \tag{4}$$

where w_i is the Gaussian noise term. Thus, it makes sense to apply a regularization technique to these regressions, linking the variable selection task to the neighborhood selection process in network structure learning. Here, we use the least absolute shrinkage and selection operator, commonly named Lasso [14]. Lasso has been widely used for simultaneous variable selection and regression because of its capacity to move many regression coefficients to zero.

The key idea is to use Lasso regression in the equivalence class space (prior to the learning stage) for each variable X_i on the remaining variables, discarding variables whose coefficients have been moved to zero as possible parents of X_i in the GBN. This produces simple models that properly fit the available data. We keep the set of parents that yields the best MDL score. As an additional contribution, to reduce the computational burden, we also take advantage of the "approximate" convexity of the MDL score for each separate variable against the remainder. This makes it possible to stop the Lasso algorithm before it ends.

Lasso has already been employed in the literature for some sorts of probabilistic graphical model learning, taking advantage of its ability for variable selection (neighborhood selection).

Li and Yang [15] performed neighborhood selection by using Lasso to estimate a DAG from a given ordering, and then transformed the DAG into an undirected graph, which is the final aim of their algorithm. Under a Bayesian perspective, they used a Wishart prior distribution for the precision matrix. The DAG prior is derived from this precision matrix prior, and such a DAG prior turns out to be equivalent to a Laplace prior. Because the objective is learning the undirected graph, they used an arbitrary ordering. Hence, this method cannot be used to estimate a final directed graph.

Meinshausen and Bühlmann [16] carried out neighborhood selection also in an undirected Gaussian graphical model setting. They used Lasso regressions individually with every variable against the rest, in such a way that an edge is created when the regression coefficient is not zero. With an appropriate selection of the penalization parameter, the method is proved to be consistent for sparse high-dimensional graphs under certain assumptions. However, the Lasso estimate is based only on individual regressions and ignores the overall likelihood of the network. This may entail some problems. For example, the regression coefficient of a given variable X_i, when X_j is the response, may (and probably will) be different from the regression coefficient of X_j, when X_i is the response; it means that it is possible that only one coefficient is shrunk to zero. In

such a case, it is not clear whether the undirected edge must be present or not in the learned graph.

Recently, Banerjee *et al.* [17] presented two new efficient algorithms to estimate the (sparse) covariance matrix that exactly maximizes the L1-penalized maximum likelihood. Friedman *et al.* [18] developed an even faster algorithm for this task, called graphical Lasso. Both papers state that the method of Meinshausen and Bühlmann [16] is an approximation, as, in general, it does not reach the maximum likelihood.

The use of Lasso directly to learn a graphical model entails an important drawback: It fails to recover the true sparsity pattern when variables are highly correlated, in particular when there exists high correlation between relevant and irrelevant variables [19]. This could be the case in some real-world scenarios. Thus, as proposed in our paper, it is reasonable to use additional heuristics instead of employing Lasso directly for network induction.

There has been less work focusing on (directed) GBNs, but we claim that the orientation is semantically useful in some problems like gene network analysis. Undirected graphs also imply a high complexity (NP-hard) in the estimation of parameters when the distribution is not Gaussian. Schmidt *et al.* [20] worked alternatively on the DAG space and the variable ordering space using Lasso to restrict both search spaces. The drawbacks of their method are those commented earlier for greedy searching in the DAG space. In this paper, instead of directly creating an arc when the corresponding regression coefficient is not zero, as in [16], we employ Lasso as a previous neighborhood selector working in the equivalence class space. The regression is also carried out for every variable against the rest, as in [20]. Therefore, Lasso is viewed as a variable-filtering first step and not as a direct model selection method. Afterward, we employ the greedy algorithm KES in the GBN training scenario.

The rest of this paper is organized as follows. Section II describes the KES algorithm and sets out basic notions on regularization. Section III introduces the proposed method, detailing how KES and regularization have been combined in a new algorithm. Section IV outlines the set of experiments used to test the algorithm to learn three synthetic data sets and a real genetic regulatory network for isoprenoid biosynthesis in *Arabidopsis thaliana*. Finally, in Section V, we outline conclusions and future work.

II. Background

A. KES Algorithm

The concept of equivalence for Bayesian networks [8] has been widely discussed in the literature: Two DAGs are *Markov equivalent* (just equivalent from now on) if they represent the same set of conditional independences.

A DAG G' is said to *include* a graph G, $G \subset G'$, if, for all the models $M(G,\theta)$ parameterized by θ whose structure is represented by G, there exists a second model $M'(G',\theta')$ parameterized by θ' with a structure G' that represents the same density function, i.e.,

$$G \subset G' \text{ iff } \forall\theta \quad \exists\theta' \quad | \quad f\left(x_{M(G,\theta)}\right) \equiv f\left(x_{M'(G',\theta')}\right). \quad (5)$$

Two DAGs G and G' are said to be *equivalent* if

$$G \subset G' \wedge G' \subset G. \quad (6)$$

For example, the DAG $A \rightarrow B \rightarrow C$ is equivalent to the DAG $A \leftarrow B \leftarrow C$ because all the joint density functions (or probability functions) that can be encoded by the first DAG can also be encoded by the second, and vice versa. The equivalence relationship is reflexive, symmetric, and transitive and hence gives rise to the concept of *equivalence class*. The equivalence class definition is the same for GBNs. A *v-structure*, also called immorality, is induced when two disconnected vertices are parents of a third vertex. G and G' are equivalent if they have the same skeletons and the same v-structures [21]. A *covered arc* is an arc that is not part of any v-structure. Thus, it is possible to move across all the individuals inside an equivalence class just by covered-arc reversing.

A formal definition of *inclusion boundary* can be presented on the basis of the DAG inclusion concept. Letting G and G' be two DAGs, we denote $G \prec G'$ to mean that $G \subset G'$, and there is no DAG G'' such that $G \subset G'' \subset G'$. The IB of G is then defined as

$$IB(G) = \{G' | G' \prec G\} \cup \{G'' | G \prec G''\}. \quad (7)$$

The set of DAGs defined by the first term of the union is called the lower IB, and the set of DAGs defined by the second term is called the upper IB. The set of operators defined by a neighborhood is said to satisfy the IB condition if, for a DAG G, the induced neighborhood includes $IB(G)$[7]. The two neighborhoods satisfying this condition are as follows:

1) Equivalence-class-based No-arc Reversals (ENR), which considers all simple edge additions and deletions from all the DAGs belonging to the equivalence class.
2) Equivalence-class-based Non-Covered-arc Reversals (ENCR), which considers all simple edge additions, deletions, and non-covered-arc reversals from all the DAGs belonging to the equivalence class.

The ENR neighborhood exactly matches, and the ENCR neighborhood includes the inclusion boundary, but they are computationally complex to calculate. As we will note hereinafter, these neighborhoods may be somehow approximated.

The so-called Meek conjecture [11] essentially claims that if a DAG G includes another DAG G', then G is reachable from G' through a finite sequence of edge additions and covered-arc reversals. An important conclusion of this premise is drawn: In the limit of large data sets, if the probability distribution (density function) has a perfect map in a DAG, a greedy search algorithm is suitable for finding the optimal solution after a finite set of edge additions and covered-arc reversals. Chickering [8] presented a proof of the Meek conjecture.

Basing on this, the GES algorithm [11] starts with the empty graph and greedily explores the equivalence class space, moving at each iteration to the state where score improvement is the greatest, and stopping the search when a locally optimal model is reached. It does so in two phases, considering different neighborhoods in each one. In the first phase, the neighborhood is based on edge addition, whereas in the second phase, the

neighborhood contains the networks obtained by deleting a single edge. Chickering [8] presented a version of GES where the entire IB is examined at each step, removing the separation into two phases. In [22], six operators were introduced to enrich the search space: insert undirected arc, delete undirected arc, insert directed arc, delete directed arc, reverse directed arc, and create v-structure.

Because of the high computational cost of working with neighborhoods (specifically ENR and ENCR) that satisfy the IB condition, mainly to enumerate all the DAGs of the class, Castelo and Koka [23] defined an approximate approach. The strategy is to somehow simulate the class by performing a sufficient number of covered-arc reversals. At each individual or set of covered-arc reversals, we get a DAG member of the class, and we explore some neighborhood of this member in a cheaper DAG space. It is shown that the number of covered-arc reversals does not need to be large because the average size of a class is bounded by a constant [9]. The number of covered-arc reversals should depend on the size of the true equivalence class if we have any idea of its cardinality [24].

The *stochastic equivalence search* (SES) algorithm introduces a modification in the GES search strategy. It does not select the best member of the IB at each step but randomly picks any of the models that improve the score.

In search of a tradeoff, GES and SES are generalized in the KES algorithm [13], controlling the degree of randomness versus greediness by a parameter $0 \leq k \leq 1$. In a nutshell, KES is an iterative algorithm that extracts an uninformed k proportion (at least one model) of the IB at each step, selecting the best model in this set. GES corresponds to the $k = 1$ case; SES is the $k = 0$ case. Nielsen *et al.* [13] presented a theoretical analysis of KES, supported on known results about GES. We will work here with the KES algorithm, performing the IB neighborhood approximation proposed in [13] and already suggested in [23].

B. Lasso Regularization

Regularization techniques (see a review in [25]) have been attracting the attention of many researchers lately and have been successfully employed in many fields related to statistics, machine learning, and signal processing. The key idea is to add a penalty term to the usual least squares linear regression with the aim of reducing the variance of the estimates, preventing overfitting, and improving the interpretation of the model. Although other regularization methods have been proposed, regularization with the ℓ_1-norm (called Lasso [26]) has been particularly popular due to its ability to move the regression coefficients of irrelevant variables to zero, thereby doing parameter estimation and variable selection simultaneously.

Let $y \in \mathbb{R}$ be the response variable to be predicted from a p-dimensional variable $\boldsymbol{X} = (X_1, \ldots, X_p)$. The linear regression model has the form

$$g(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{p} x_i \beta_i. \tag{8}$$

Parameters $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^{\mathrm{T}}$ are estimated from a set of training data, denoted by $\mathcal{D}' = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, by the least squares method. Each $\boldsymbol{x}_r = (x_{1r}, \ldots, x_{pr})^{\mathrm{T}}$ is a vector of measurements for the rth instance. The Lasso formulation shrinks the regression coefficients by imposing a penalty on their size. Thus, the Lasso estimate minimizes a penalized residual sum of squares

$$\hat{\boldsymbol{\beta}}^s = \arg\min_{\beta} \sum_{r=1}^{n} \left(y_r - \beta_0 - \sum_{i=1}^{p} x_{ir} \beta_i \right)^2 \tag{9}$$

subject to

$$\sum_{i=1}^{p} |\beta_i| \leq s. \tag{10}$$

We can omit the intercept β_0 from the model by standardizing the predictors. $\hat{\boldsymbol{\beta}}^s$ depends on the value of $s \geq 0$, the penalty parameter: The greater the value of s, the greater the number of $\hat{\beta}_i^s$ that are not zero. The *regularization path* is composed of the values of coefficients $\hat{\beta}_i^s$ across the range of s. In Lasso, as we increase the value of s, one $\hat{\beta}_i^s$ at a time is made different from zero. This allows us to pay attention only to the finite set of s values where a new $\hat{\beta}_i^s$ is driven to a nonzero value. Between two of these values, the changes in coefficients $\hat{\beta}_i^s$ are linear.

Computing the regularization path is a quadratic programming problem with linear inequality constraints. However, it can be efficiently solved by the least angle regression (LARS) algorithm in $O(p^3 + np^2)$ computations [27] (the same cost of a least squares fit on p variables).

Lasso has proven to have interesting theoretical properties for variable selection in the literature, showing its capacity to deal with high-dimensional data (very much present in real domains). Specifically, to hold these properties, Meinshausen and Bühlmann [28] showed that the data-set cardinality has to grow at no more than a logarithmic rate against the number of variables. This turns out to be a clear advantage when working with biological data, where the amount of training samples is usually small compared with the number of variables. The consistency of Lasso for variable selection (determination of the true model) is demonstrated for underlying models that fulfill certain conditions [16], [19]. Zhao and Yu [19] introduced the term "irrepresentable condition," meaning that the correlation between the relevant (nonzero coefficients) and irrelevant variables (zero coefficients) must be limited; otherwise, spurious variables may be included in the selected variable set.

III. KES Combined With Lasso

We propose a Lasso-based previous step to preselect a set of potential parents for each variable on which the KES algorithm will work greedily. Arcs corresponding to zero coefficients in the penalized regression are discarded. The sparse candidate algorithm [29] also performs a preselection phase. This algorithm needs the user to set the maximum number of parents per node in advance. The sparse candidate algorithm restricts the search space so that there is only one set of possible parents for each variable in the subsequent maximization step (typically a greedy algorithm). These sets are constructed by including the variables that are most closely associated with the target

variable; this is usually quantified with a pairwise measure like *mutual information*.

Alternatively, we employ Lasso with each variable on the rest, discarding as parents the variables whose regression coefficients become zero. Afterward, we launch the KES algorithm on the equivalence class search space. Thus, unlike the sparse candidate algorithm, we do not need to establish a maximum number of parents in advance.

The pseudocode of our proposal is shown in Algorithm 1, illustrating the two phases of the algorithm. Parent restriction appears in the *for* loop, calculating for each variable the set of coefficients that yields the best MDL score in the regularization path (represented by a matrix with p columns and a variable number of rows—one for each point in the regularization path where a coefficient vanishes or reappears into the model). The score+search KES algorithm, restricted by the previous step, is enclosed in the *repeat* loop. Notice that MDL is used in both phases.

Algorithm 1 Lasso embedded in KES
Input: data set $\mathcal{D}$ with p variables and n cases, $k \in [0, 1]$
Output: partially DAG G
for $i = 1$ **to** p **do**
  $Path(i) :=$ matrix containing the L1-regularization path of variable X_i
  $Beta(i) :=$ row of p coefficients with the best MDL score in $Path(i)$
  $PotentialParents(i) :=$ set of variables such that $Beta(i)(j) \neq 0$, for $j = 1 \dots p$
end for
Initialize $G :=$ empty or randomly generated model
$minimum := false$
repeat
  $K := \max(k \cdot size(IB(G, PotentialParents)), 1)$
  where $IB(G, PotentialParents)$ is the $IB(G)$ constrained by Lasso
  $S :=$ set of K models drawn from $IB(G, PotentialParents)$
  $G' :=$ the model from S with the best MDL score
 if $MDL(G') < MDL(G)$ **then**
  $G := G'$
 else
  $minimum := true$
 end if
until $minimum$ is $true$

Note that not all variables have to share the same value of s when carrying out their regression against the rest of variables, as some nodes in the true structure may have stronger connectivity than others. A stronger regularizer (smaller s) would be required for these nodes so that more regression coefficients $\hat{\beta}_i^s$ will be driven to zero. A common strategy for selecting s is to choose it by cross-validation from a grid of values. However, it would have to be applied individually for each variable and could hence introduce a heavy computational load in high-dimensional problems.

TABLE I
MEAN AND STANDARD DEVIATION (IN SECONDS) OF TEN RUNS MEASURING THE RUN TIME FOR SOME NETWORK SIZES OF THE *FACTOR* STRUCTURE (SEE SECTION IV). ALL DATA SETS HAVE 1000 INSTANCES

No. variables	KES	KES + Lasso	Lasso
20	$3.08(\pm 0.22)$	$1.75(\pm 0.06)$	$0.25(\pm 0.06)$
50	$84.68(\pm 13.28)$	$27.57(\pm 0.82)$	$1.77(\pm 0.05)$
100	$1145.3(\pm 28.11)$	$259.0(\pm 9.90)$	$16.3(\pm 0.84)$

Instead, Schmidt *et al.* [20] took advantage of the piecewise constant nature of the number of nonzero coefficients against parameter s and suggested to take for each variable the best set of "parent candidates" from such regularization path according to some criterion, which has a finite and reduced number of solutions. That is the approach that we use in this paper. Hence, we do not estimate an explicit value of s. We employ the (score-equivalent) MDL criterion, which is defined as

$$MDL(X_i) = m \log(n)/2 + NLL\left(X_i | \boldsymbol{Pa}(X_i)\right) \qquad (11)$$

where m is the number of parameters different from zero and $NLL(X_i|\boldsymbol{Pa}(X_i))$ is the negative log-likelihood of the network made up of this node and its parents.

In short, we evaluate all the points (all the sets of p coefficients) where a new coefficient vanishes (or appears) in each variable's regularization path, and choose the set of coefficients that minimizes the MDL score. The chosen set of penalized regression coefficients is not used again beyond the parent preselection stage, and the GBN parameters will be learned later, irrespective of these coefficients.

In the second phase, we must choose between two strategies. The first is that a variable X_j is included as a possible parent of X_i if X_j is selected by Lasso when the response variable is X_i or if X_i is selected by Lasso when the response variable is X_j. The second is if both conditions are fulfilled. We have tested the two strategies, which we will call OR-Lasso and AND-Lasso, respectively.

Even restricted to the aforementioned subset of potential parents, the KES algorithm does not lose the theoretical properties that we described earlier if we assume L1 regularization to be an ideal variable selector (specifically if it does not miss true relations). As noted previously, this applies under certain conditions. In this case, it will find a set of variables with all the parents, children, and coparents: the Markov blanket [20]. Hence, ideally, Lasso only discards false parents.

We use a DAG to represent the current equivalence class in our implementation. Therefore, we need a method to approximate the IB from a DAG representation. We do not calculate the complete IB neighborhood at each step; rather, we approximate its size from the total number of possible parents (taking the Lasso restriction into account). Then, we derive a k-proportional number of models of this estimated size and keep the best one. Thus, we are not actually drawing a k proportion from the set of models that is better than the current one, but an approximate k proportion from the total (approximate because the size of the IB has been approximated), keeping the best model from this set.

The worst case computational cost of the algorithm is equivalent to KES. This would be the unlikely situation where the

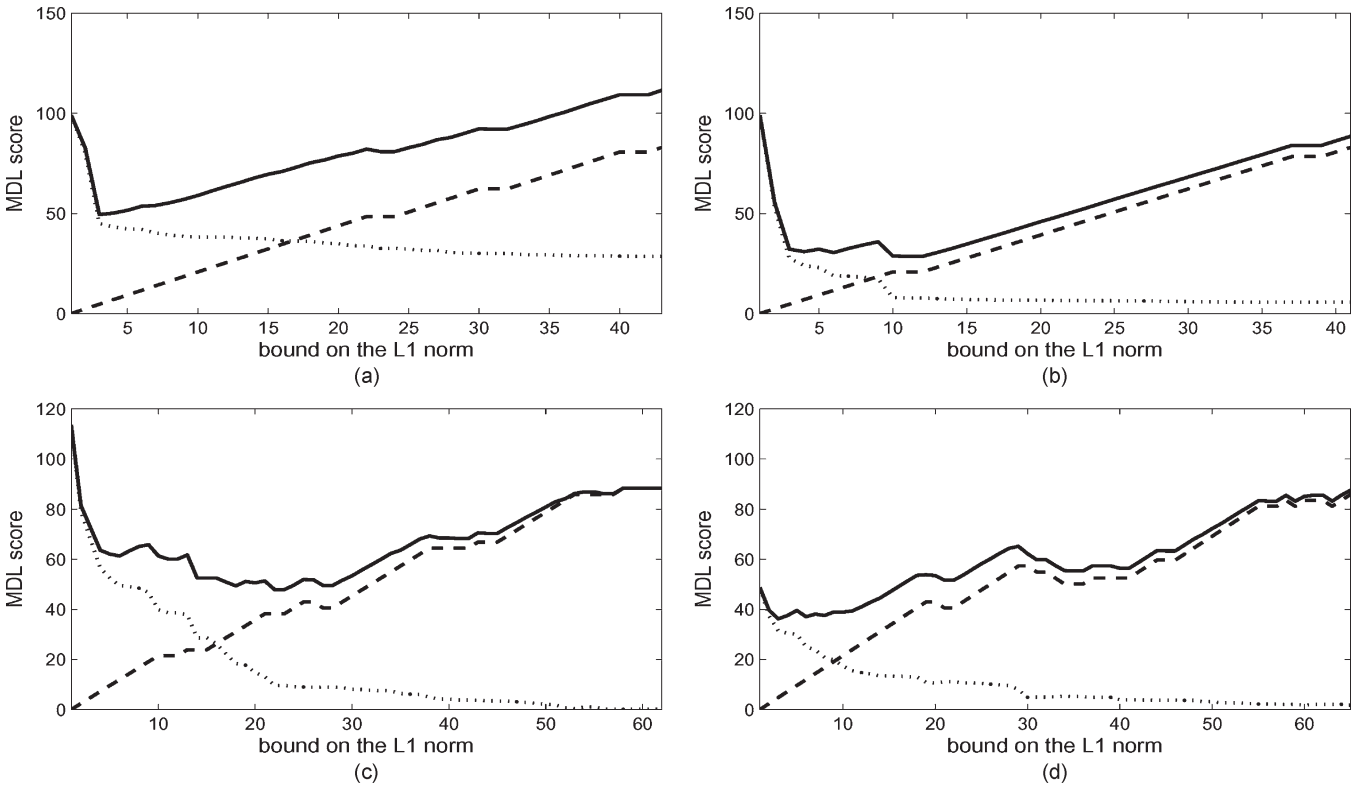


Fig. 2. At the top, the MDL curve along the regularization path for two variables of the *Alarm* network (see Section IV-A). At the bottom, two variables of isoprenoid biosynthesis for *Arabidopsis thaliana* (see Section IV-B). The solid line represents MDL, the dotted line is the negative log-likelihood, and the dashed line is the penalization term in MDL [the first term in (11)]. Therefore, the solid line is the sum of the other two lines.

best Lasso preselection yields that all variables can be parents of all variables. However, Lasso is proved to hold a parsimonious property [30]. Thus, the variable selector nature of Lasso usually restricts the search space quite a lot and makes the mean computational cost of the proposed approach definitely lower than that of the KES algorithm alone. Some run times are shown in Table I to illustrate this issue.

Moreover, this paper introduces a simple but useful heuristic on the LARS algorithm when used in the parent restriction phase. As noted before, we must search for the MDL-optimal $\hat{\boldsymbol{\beta}}^s$ for each and every variable in the regularization paths. The MDL score along the regularization path of each Lasso with each variable as the response (starting with all $\hat{\beta}_i^s = 0$ and ending with all $\hat{\beta}_i^s \neq 0$) often follows a convex curve with only one minimum [see Fig. 2(a)]. The reason is that the MDL score is the sum of two terms [see (11)]. The first term penalizes networks that contain many parameters. It strictly increases when parameters are added to the model. The second term is the negative log-likelihood, and it strictly decreases as more coefficients are made different from zero. The addition of a strictly increasing function and a strictly decreasing function is a convex function with only one global minimum. In some cases, however, the regularization path for Lasso may have more than p steps, i.e., at some steps, a variable may become zero as s is increased, and be added again to the model later. In such a situation, the MDL function along the regularization path is not convex anymore.

These irregularities are the source of our heuristic, as more than one local minimum is sometimes present [Fig. 2(b)–(d)]. For this reason, the whole curve should be inspected to be sure of getting the global minimum. However, the global minimum is usually reached at an early phase [an exception is shown in Fig. 2(c)]. Even more importantly, when the MDL curve has grown for long enough, it normally never decays significantly again. This leads us to stop LARS if the MDL curve has grown for a number of steps, e.g., equal to 20% of the full model variables, where we can be reasonably sure of having obtained the global minimum. In this way, we are saving a great deal of computational cost.

In some cases, some local minima are reached later on [Fig. 2(c) and (d), particularly Fig. 2(d)], but they are far from the global minimum and should be ignored. To make our stopping method robust to these minor slumps and preserve its efficiency, instead of checking punctual MDL values, we calculate the mean in a certain window of MDL values. This way, we only keep searching if the decay is relevant, i.e., we stop LARS when the MDL mean of this window has been growing for enough iterations. If the number of values to be included in this mean is too small (or is just one), we are being conservative and will probably advance along the regularization path further than necessary. On the other hand, if we take many values, we risk stopping early and missing the real global minimum. Empirically, we have found that taking the last five values leads to good results, although this point should be further researched.

In Table II, we show the iteration number where the algorithm stops using different window sizes for each variable against the rest in the *Arabidopsis thaliana* isoprenoid

TABLE II
LARS STOPPING ITERATION FOR DIFFERENT WINDOW SIZES, FOR THE *Arabidopsis thaliana* DATA SET. THE SECOND COLUMN REPRESENTS THE ITERATION WHEN THE GLOBAL MINIMUM IS REACHED

Variable	Optimal iteration	Window size						
		1	2	3	4	5	6	7
AACT1	10	18	18	18	18	17	18	19
AACT2	3	14	14	14	14	13	14	14
CMK	2	9	10	11	12	12	13	14
DPPS1	3	14	14	14	14	12	14	15
DPPS2	2	9	10	11	12	12	13	14
DPPS3	1	9	10	11	12	12	13	14
DXPS1	22	33	33	33	33	30	33	34
DXPS2	2	9	10	11	12	12	13	14
DXPS3	6	15	15	15	15	14	15	14
DXR	4	14	14	14	14	13	14	14
FPPS1	5	14	14	14	14	13	14	15
FPPS2	3	9	10	11	12	12	13	14
GGPPS1	3	17	17	17	17	16	17	15
GGPPS2	2	9	10	11	12	12	13	14
GGPPS3	1	9	10	11	12	12	13	14
GGPPS4	1	9	10	11	12	12	13	14
GGPPS5	1	9	10	11	12	12	13	14
GGPPS6	2	9	10	11	12	12	13	14
GGPPS8	2	9	10	11	12	12	13	14
GGPPS9	2	9	10	11	12	12	13	14
GGPPS10	2	9	10	11	12	12	13	14
GGPPS11	4	14	14	14	14	13	14	14
GGPPS12	3	14	14	14	14	13	14	14
GPPS	7	17	17	17	17	16	17	17
HDR	3	9	10	11	12	13	13	14
HDS	4	9	10	11	12	13	13	14
HMGR1	4	9	10	11	12	13	13	14
HMGR2	6	15	15	15	15	14	15	16
HMGS	4	14	14	14	14	13	14	15
IPPI1	6	9	10	11	12	21	22	21
IPPI2	4	9	10	11	12	21	22	21
MCT	3	9	10	11	12	21	22	21
MECPS	3	9	10	11	12	21	22	21
MK	3	9	10	11	12	21	22	21
MPDC1	3	9	10	11	12	21	22	21
MPDC2	4	9	10	11	12	21	22	21
PPDS1	2	9	10	11	12	21	22	21
PPDS2	2	9	10	11	12	21	22	21
UPPS1	3	14	14	14	14	13	14	15

TABLE III
LARS STOPPING ITERATION FOR DIFFERENT STOPPING RATIOS (SEE TEXT), FOR THE *Arabidopsis thaliana* DATA SET. THE SECOND COLUMN REPRESENTS THE ITERATION WHEN THE GLOBAL MINIMUM IS REACHED

Variable	Optimal iteration	Stopping ratio				
		0.05	0.1	0.15	0.2	0.3
AACT1	10	14	16	18	20	24
AACT2	3	10	12	14	16	20
CMK	2	8	10	12	14	18
DPPS1	3	10	12	14	16	20
DPPS2	2	8	10	12	14	18
DPPS3	1	8	10	12	14	18
DXPS1	22	25	31	33	35	39
DXPS2	2	8	10	12	14	18
DXPS3	6	11	13	15	17	21
DXR	4	10	12	14	16	20
FPPS1	5	10	12	14	16	20
FPPS2	3	8	10	12	14	18
GGPPS1	3	10	15	17	19	23
GGPPS2	2	8	10	12	14	18
GGPPS3	1	8	10	12	14	18
GGPPS4	1	8	10	12	14	18
GGPPS5	1	8	10	12	14	18
GGPPS6	2	8	10	12	14	18
GGPPS8	2	8	10	12	14	18
GGPPS9	2	8	10	12	14	18
GGPPS10	2	8	10	12	14	18
GGPPS11	4	10	12	14	16	20
GGPPS12	3	10	12	14	16	20
GPPS	7	13	15	17	19	23
HDR	3	8	10	12	14	18
HDS	4	8	10	12	14	18
HMGR1	4	8	10	12	14	18
HMGR2	6	11	13	15	17	21
HMGS	4	10	12	14	16	20
IPPI1	6	8	10	12	24	28
IPPI2	4	8	10	12	24	28
MCT	3	8	10	12	24	28
MECPS	3	8	10	12	24	28
MK	3	8	10	12	24	28
MPDC1	3	8	10	12	24	28
MPDC2	4	8	10	12	24	28
PPDS1	2	8	10	12	24	28
PPDS2	2	8	10	12	24	28
UPPS1	3	10	12	14	16	20

biosynthesis data set (see Section IV-B). The optimal iteration, where the global minimum is reached, is depicted in the second column; if the stopping iteration is lower than the optimal one, we are trapped in a local minimum. Note that the window size does not have a big impact on most variables. For example, for the first variable, the algorithm always stops around iteration 18, i.e., beyond the global minimum in iteration 10, regardless of the window size. More meaningful is the effect of the number of iterations that the MDL mean of the window needs to keep increasing to stop. In Table III, we show the ratio of this parameter against the number of variables. To simplify, we call such ratio "stopping ratio." Note that the difference between 0.05 and 0.3 is significant. Again, the iteration corresponding to the global miminum is depicted in the second column. Note that, in both experiments, we always advanced further than the optimal iteration and thus covered the global minimum. For all variables, the lowest size of the regularization path is over 60, and the mean size is 82. However, the global minimum is usually in the first third of the regularization path.

IV. EXPERIMENTS

A. Synthetic Networks

The *Alarm* network [31] contains 37 nodes and 46 arcs. The *Insurance* network [32] has 27 variables and 52 arcs. Both are commonly used to test Bayesian network learning algorithms. Finally, we have generated a synthetic network called *Factor*, with 100 variables and 382 arcs. This holds an arc from variable X_i to X_j if i is a divisor of j. Because we need a GBN, we will use the dependences of each network to simulate continuous Gaussian data sets of 100 samples for *Alarm*, 1000 samples for *Insurance*, and 50 samples for *Factor* (testing the $p > n$ case). Parameter β_i in (2) is generated at random from a standard normal distribution. The rest of the parameters of the density functions are fixed ($m_i = 0$ and $v_i = 1$). We run the KES algorithm for ten different values of $k : 0.1, 0.2, \ldots, 0.9, 1.0$, with and without a previous Lasso step. Here, we illustrate the results using the OR-Lasso strategy, as the AND-Lasso strategy turned out to be very restrictive in this case, producing networks with few arcs. For each k, we performed ten runs and calculated

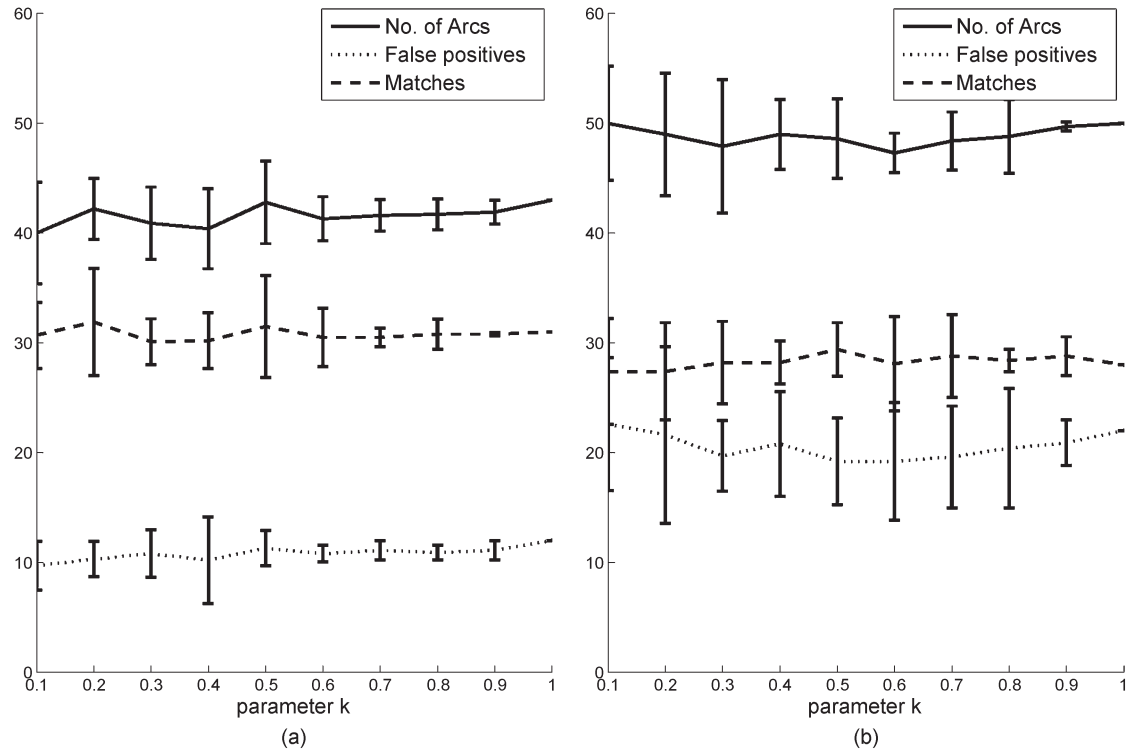


Fig. 3. Behavior of the KES algorithm for the *Alarm* network. (a) With Lasso preselection. (b) Without Lasso parent preselection. Total, true, and false arcs are shown.

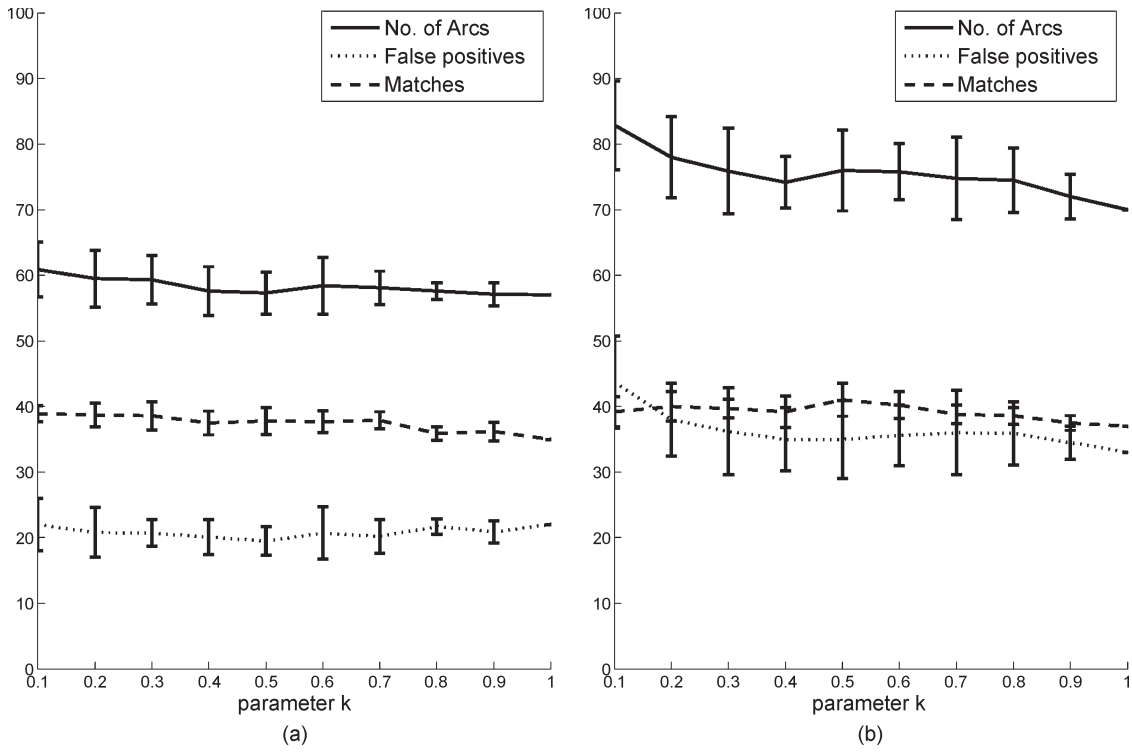


Fig. 4. Behavior of the KES algorithm for the *Insurance* network. Same setting as that in Fig. 3.

the mean and the standard deviation. Results are shown in Figs. 3–5. We also tried the approach proposed in [20], which works in the DAG space, although we do not show the results in the figures for clarity. Instead, we present means and standard deviations in Table IV.

Note that our approach (working in the equivalence class space) on the whole outperforms the networks obtained working in the DAG space. This is particularly notable in the *Alarm* network. For the *Factor* network, the number of correct arcs is equivalent in both spaces, whereas in the equivalence class

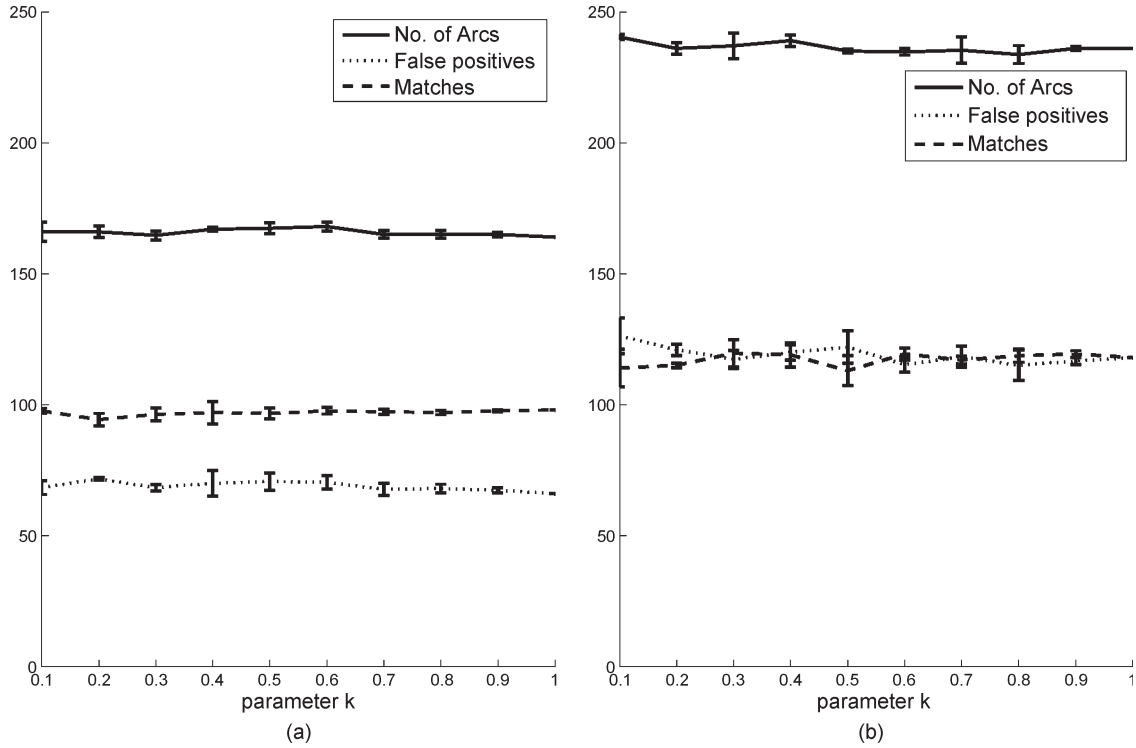


Fig. 5. Behavior of the KES algorithm for the *Factor* network. Same setting as that in Fig. 3.

TABLE IV
MEAN AND STANDARD DEVIATION OF SOME MEASURES FOR NETWORKS OBTAINED BY THE APPROACH PROPOSED IN [20]

Network	No. of arcs	False positives	Matches
Alarm	$33.50(\pm 1.58)$	$13.70(\pm 1.70)$	$19.80(\pm 0.42)$
Insurance	$63.80(\pm 4.10)$	$22.60(\pm 3.03)$	$41.20(\pm 2.44)$
Factor	$168.10(\pm 2.02)$	$70.50(\pm 2.76)$	$97.60(\pm 1.17)$

space, the number of false positives is slightly lower. On the other hand, for the *Insurance* network, where $n > p$, the results between the DAG space and the equivalence class space are quite similar. Note also that the random generation of the network parameters may affect the final output and could be the cause of some differences between the learned structures and the original network.

Regarding the comparisons, within the equivalence class space, between networks learned with Lasso preselection and networks learned without it, the main conclusion that we can draw is that the Lasso information makes the output network sparser and tends to include fewer false arcs and, encouragingly, even more correct arcs. Again, this is more obvious for the *Alarm* network. For the *Insurance* network, the number of correct arcs is not very different, but the Lasso preselection yields less false positives (and, therefore, fewer total arcs). The same happens for the *Factor* network, where the number of false positives and the number of correct arcs are surprisingly alike for the KES algorithm.

Without Lasso, we also observe that intermediate values of k (moderate randomness) result in slightly better networks. The *Factor* network, where k has no influence at all, is an exception. When using Lasso, the quality of the networks is roughly equivalent in all cases of k. Moreover, the variance of the results is lower. Consequently, we could say that, with the preselection phase, the algorithm becomes quite robust to k, i.e., there is not so much variation on parameter k, and this is no longer a cause for concern. Furthermore, as mentioned before, the computational cost is significantly lower when Lasso is employed.

B. Pathways of the Arabidopsis thaliana Plant

Probabilistic graphical models and, specifically, GBNs may be used to model genetic networks. In the GBN case, each variable represents a continuous gene expression level (see [33] for a review). Narrowing the field down to static GBNs (that do not model gene coregulation against time), Wu *et al.* [34] defined the GBN by previously determining the conditional independence relationships, and Imoto *et al.* [35] combined microarray data and known biological information to train the model.

Next, we will test our method with a real-world data set taken from a biological environment: a list of 118 gene expression patterns measured under different conditions for 40 genes that are found to be relevant in isoprenoid biosynthesis for *Arabidopsis thaliana*[36]. *Arabidopsis thaliana* is the first plant whose complete genome has been sequenced. Isoprenoids are the largest family of biological substances in nature and the oldest known biomolecules; over 30 000 known compounds help in a great variety of biochemical processes. Understanding the nature of their synthesis is a task with many practical pharmaceutical and food applications. It is known that such synthesis follows two different gene routes in high-order plants like *Arabidopsis thaliana*: the mevalonate (MVA) pathway and the methylerythritol 4-phosphate (MEP) pathway. Of the 40 measured genes, 16 come from the MVA pathway, 19 are

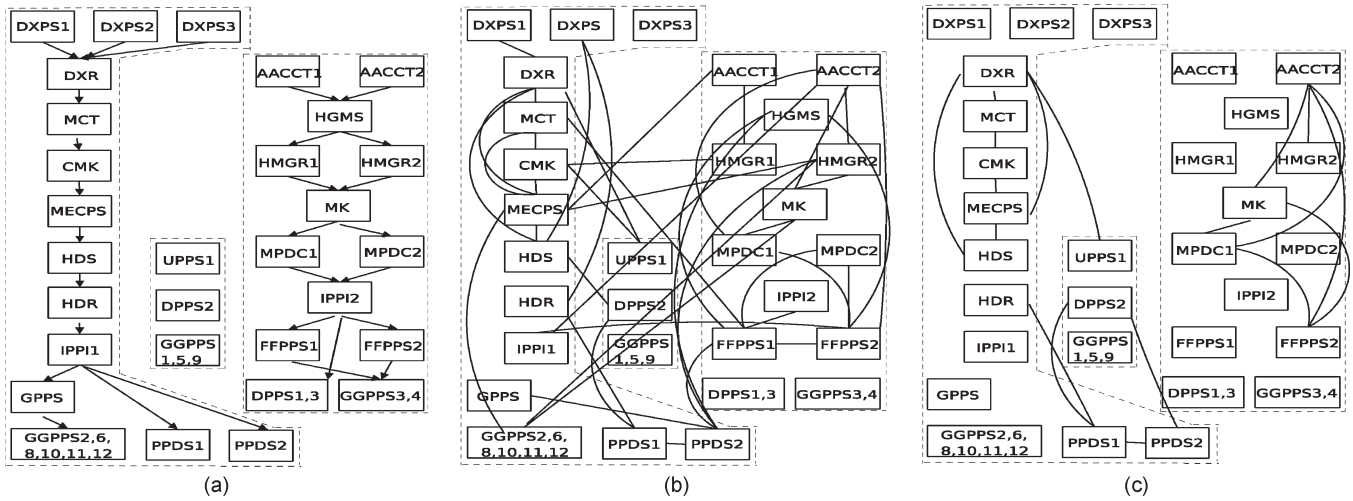


Fig. 6. (a) Real isoprenoid biosynthesis network. (b) Network shown in [36]. (c) Network shown in [37].

from the MEP pathway, and the remaining 5 are encoding proteins located in the mitochondrion. Fig. 6(a) shows the true pathways: The MEP pathway is the set of proteins on the left, and the MVA pathway is the set on the right; genes related to mitochondrial proteins are UPPS1, DPPS2, GGPPS1, GGPPS5, and GGPPS9.

Our aim here is to check if our method is capable of neatly separating the two pathways. We train ten networks for each k value ($k \in \{0.1, 0.2, \ldots, 0.9, 1.0\}$), and we only keep those arcs that have been repeated with a frequency of at least fr. In order to experiment with different degrees of sparseness, we examine some values for fr: 0.6, 0.7, 0.8, 0.9, and 1.0. The networks obtained in [36] and [37] are shown in Fig. 6(b) and (c), respectively.

It is also interesting to investigate the cross relationship between both pathways. One might expect a limited connectivity between the two pathways [37] because they are both developed in different parts of the cell [see Fig. 6(a)]. However, Wille *et al.* [36] cited some reports about these interactions, showing that cross-link connections do exist under certain circumstances.

Both the method suggested in [36] and the threshold gradient method (TGD) proposed in [37] identify a separation of both pathways. Note that undirected networks are used in both papers, whereas we train directed networks. The simpler method shown in [36] trained a dense network. Although this network, in essence, owns many true arcs and distinguishes the two pathways, it connects many genes that are independent in the true network, with relatively dense cross-link connections. On the other hand, TGD, using bootstrap and keeping only the arcs that appear in at least 50% of the obtained networks, reached a sparse network, and no cross-link between pathways was drawn. However, there are some connections between the MEP pathway and mitochondrial proteins (located at the center of the network) that do not exist in the true-pathway diagram. Furthermore, this network misses many true edges. For example, DXPS1, DXPS2, and DXPS3 appear to be completely independent, as do all GGPPS genes.

We have built different networks, depending on the parent preselection strategy used (no preselection, AND-Lasso, and OR-Lasso) and on the fr and k values. Except for the networks trained without Lasso preselection, parameter k has little influence. If we do not employ Lasso [Fig. 7(a)], the KES algorithm outputs very dense networks where we can barely distinguish the two pathways. Only when fr is set to a high value that sparser networks are reached, but it is not yet possible to appreciate the two pathways. For example, there are no arcs connecting genes inside the MVA pathway in Fig. 7(a). Note that this network has an equivalent number or even fewer arcs than the others; this is because of the higher variability of the networks trained without constraints: There are many arcs, but they differ from one run to another.

AND-Lasso networks [Fig. 7(b)] turn out to be the most interesting because they identify two interconnected modules in the network with relatively few arcs. The MVA pathway is specially well connected as compared with the real one in Fig. 6(a), and the DXPS, DXR, MCT, CMK, MECPS, and HDS set is also connected in the MEP pathway except for the missing CMK—MECPS arc. Moreover, most AND-Lasso networks discover a relation between IPPI1 and some MVA pathway genes, which, as noted in [36], could have an interesting biological interpretation.

When using an OR-Lasso [Fig. 7(c)], there are many arcs that appear in at least 80% of the trained networks. Although we find some cross-link connections, we also find a higher arc density in each pathway. For instance, in the MEP pathway, DXPS, DXR, MCT, CMK, MECPS, HDS, and HDR are closely connected. Again, we find an interaction of IPPI1 with elements of the MVA pathway.

Table V shows a quantitative comparison among the networks in Figs. 6(b) and (c) and 7(a)–(c). We score each pathway using a simplified version of the *structural Hamming distance* (SHD) [38]. This measure is defined by the number of operators (add or delete an undirected edge, and add, remove, or reverse a directed arc) that are different in the PDAG to be scored and in the real PDAG. Because we want to evaluate also the undirected networks shown in Fig. 6(b) and (c), we will ignore the arc orientation and will not count reversal operators.

We also measure the rate between the number of arcs within the pathways and the number of arcs crossing between

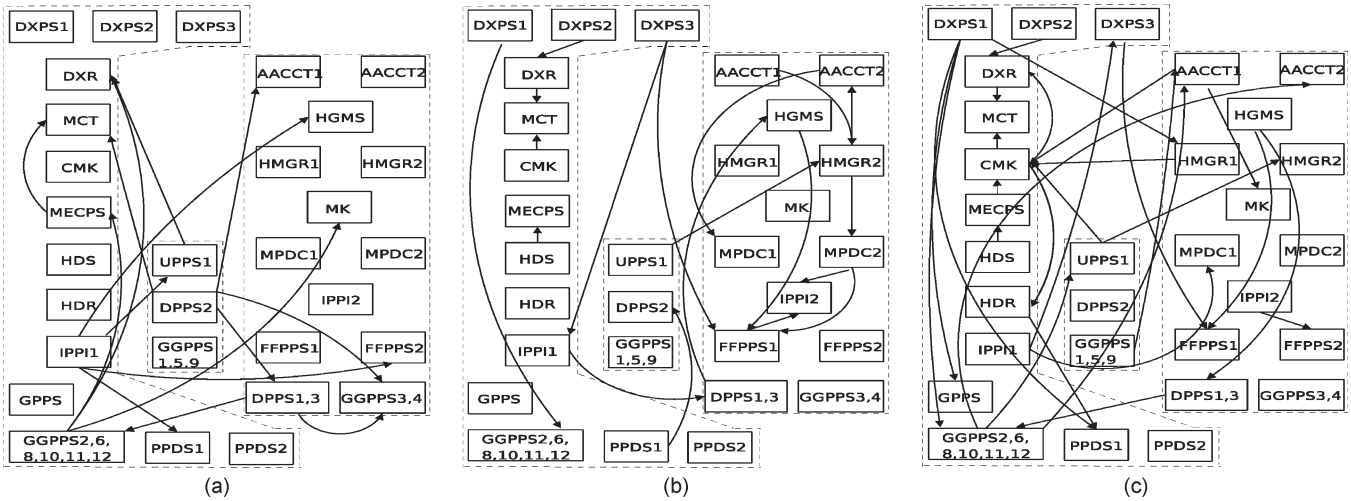


Fig. 7. (a) Network trained without Lasso preselection ($fr = 0.8, k = 0.2$). (b) Network trained with AND-Lasso preselection ($fr = 0.7, k = 0.7$). (c) Network trained with OR-Lasso preselection ($fr = 0.8, k = 0.8$).

TABLE V
SHD AND SEPARABILITY MEASURES FOR THE NETWORKS SHOWN IN FIGS. 6(b) AND (c) AND 7(a)–(c). SHD ASSESSES INDEPENDENTLY EACH PATHWAY

Network	SHD in pathway MEP	MVA	Total SHD	Separability
Wille *et al.*'s method [36]	16	22	38	1.52
TGD [37]	14	21	35	5.66
KES	15	16	31	0.50
AND-Lasso	11	20	31	3.75
OR-Lasso	15	17	32	1.60

pathways and linking with mitochondrial genes. This value might be considered as a measure of the ability to separate the pathways, and will be denoted as *separability*.

Roughly speaking, the networks of Figs. 6(c) and 7(b) (corresponding to TGD and AND-Lasso methods) appear to be the best. Whereas TGD presents the best separability, AND-Lasso gives good SHD results and could produce the most refined pathways at some extent.

In summary, we have built networks that can compete with state-of-the-art methods, showing a very good computational performance. This has been shown with some synthetic data sets and a real biological network.

V. CONCLUSION

We have presented a procedure for learning the structure of GBNs based on a well-known regularization method for parent filtering and on the KES algorithm, a greedy algorithm working on the equivalence class space. As discussed previously, there are theoretical properties that support both methods. To the best of our knowledge, this is the first time that regularization has been employed in equivalence class searching.

One advantage of our method is its computational efficiency. It is known that LARS solves Lasso with a reduced computational burden. Also, with parent restriction, the KES algorithm is significantly faster than ordinary KES while offering better results. This means that several executions or bootstrapping, trying to extract common patterns, can be run in situations where this would normally be infeasible. With the aim of improving efficiency, we have developed a simple but useful MDL-based method for stopping LARS in advance when it is used for parent preselection.

We have applied our approach with good results to three synthetic databases and a real biological data set, the isoprenoid biosynthesis pathways of *Arabidopsis thaliana*. Our results are successfully backed by previous domain knowledge, even though the available data offer just gene coexpression levels, not always directly related to regulatory patterns, i.e., it does not always exactly reflect regulatory dependences. Previous work on building graphical models for this data set used undirected Gaussian networks. We think that our link orientation could supply useful information on the underlying biological processes in some situations. In fact, the original biological network that we are trying to model is also directed.

For future research, it would be interesting to evaluate the different varieties of Lasso (fused, group, elastic net, etc.) for parent preselection, depending on the nature of the biological domain. We think that the combination of the KES algorithm with an appropriate Lasso extension may be a useful tool that deserves further investigation. We may also intend to refine our LARS stopping method to save computation resources and make sure that we will find the true global minimum of the MDL curve.

ACKNOWLEDGMENT

The authors would like to thank J. Nielsen for the valuable support with the KES implementation. The authors would also like to thank the referees for their valuable remarks that have definitely contributed to the improvement of this paper.

REFERENCES

[1] D. Geiger and D. Heckerman, "Learning Gaussian networks," in *Proc. 10th Conf. Uncertainty Artif. Intell.*, 1994, pp. 235–243.
[2] E. Castillo, J. M. Gutiérrez, and A. S. Hadi, *Expert Systems and Probabilistic Network Models*. New York: Springer-Verlag, 1997.
[3] P. W. Smith and J. Whittaker, "Edge exclusion tests for graphical Gaussian models," in *Learning in Graphical Models*, M. Jordan, Ed. Dordrecht, The Netherlands: Kluwer, 1998, pp. 555–574.

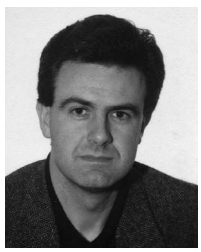
[4] D. Margaritis, "Distribution-free learning of Bayesian network structure in continuous domains," in *Proc. 20th Nat. Conf. Artif. Intell.*, 2005, pp. 825–830.
[5] F. Bach and M. I. Jordan, "Learning graphical models with Mercer kernels," in *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2003.
[6] S. Davies, "Fast factored density estimation and compression with Bayesian networks," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 2002.
[7] T. Kočka, R. R. Bouckaert, and M. Studený, "On characterizing inclusion of Bayesian networks," in *Proc. 17th Conf. Uncertainty Artif. Intell.*, 2001, pp. 261–268.
[8] D. M. Chickering, "Optimal structure identification with greedy search," *J. Mach. Learn. Res.*, vol. 3, pp. 507–554, 2002.
[9] S. B. Gillispie and M. D. Perlman, "Enumerating Markov equivalence classes of acyclic digraph models," in *Proc. 17th Conf. Uncertainty Artif. Intell.*, 2001, pp. 171–177.
[10] S. B. Gillispie, "Formulas for counting acyclic digraph Markov equivalence classes," *J. Stat. Plan. Inference*, vol. 136, no. 4, pp. 1410–1432, Apr. 2006.
[11] C. Meek, "Graphical models: Selecting causal and statistical models," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1997.
[12] D. M. Chickering and C. Meek, "Finding optimal Bayesian networks," in *Proc. 18th Conf. Uncertainty Artif. Intell.*, 2002, vol. 3, pp. 94–102.
[13] J. Nielsen, T. Koka, and J. Pea, "On local optima in learning Bayesian networks," in *Proc. 19th Conf. Uncertainty Artif. Intell.*, 2003, pp. 435–442.
[14] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. R. Stat. Soc., B*, vol. 58, no. 1, pp. 267–288, 1996.
[15] F. Li and Y. Yang, "Using modified Lasso regression to learn large undirected graphs in a probabilistic framework," in *Proc. AAAI*, 2005, pp. 801–806.
[16] N. Meinshausen and P. Bühlmann, "High dimensional graphs and variable selection with the Lasso," *Ann. Statist.*, vol. 34, no. 3, pp. 1436–1462, 2006.
[17] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data," *J. Mach. Learn. Res.*, vol. 9, pp. 485–516, 2008.
[18] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical Lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, Jul. 2008.
[19] P. Zhao and B. Yu, "On model selection consistency of Lasso," *J. Mach. Learn. Res.*, vol. 7, pp. 2541–2567, 2006.
[20] M. W. Schmidt, A. Niculescu-Mizil, and K. P. Murphy, "Learning graphical model structure using L1-regularization paths," in *Proc. AAAI*, 2007, pp. 1278–1283.
[21] T. Verma and J. Pearl, "Equivalence and synthesis of causal models," in *Proc. 6th Conf. Uncertainty Artif. Intell.*, 1990, pp. 1287–1330.
[22] D. M. Chickering, "Learning equivalence classes of Bayesian-network structures," *J. Mach. Learn. Res.*, vol. 2, pp. 445–498, 2002.
[23] R. Castelo and T. Koka, "On inclusion-driven learning of Bayesian networks," *J. Mach. Learn. Res.*, vol. 4, pp. 527–574, 2003.
[24] J. Muruzábal and C. Cotta, "A study on the evolution of Bayesian network graph structure," in *Advances in Probabilistic Graphical Models*, vol. 214. New York: Springer-Verlag, 2007, pp. 193–213.
[25] J. Fan and R. Li, "Statistical challenges with high dimensionality: Feature selection in knowledge discovery," in *Proc. Int. Congr. Mathematicians, Madrid*, vol. 3, J. V. M. Sanz-Sole, J. Soria, and J. Verdera, Eds. Zürich, Switzerland: Eur. Math. Soc., 2006, pp. 595–622.
[26] T. Hesterberg, N. M. Choi, L. Meier, and C. Fraley, "Least angle and l_1 penalized regression: A review," *Statist. Surv.*, vol. 2, pp. 61–93, 2008.
[27] B. Efron, I. Johnstone, T. Hastie, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.
[28] N. Meinshausen and P. Bühlmann, "Consistent neighbourhood selection for sparse high-dimensional graphs with the Lasso," *Statist. Surv.*, vol. 2, pp. 61–93, 2004.
[29] N. Friedman, I. Nachman, and D. Pe'er, "Learning Bayesian network structure from massive datasets: The sparse candidate algorithm," in *Proc. Int. Conf. Artif. Intell.*, 1999, pp. 206–215.
[30] K. Knight and W. Fu, "Asymptotics for Lasso-type estimators," *Ann. Statist.*, vol. 28, no. 5, pp. 1356–1378, 2000.
[31] I. Beinlich, G. Suermondt, R. Chávez, and G. F. Cooper, "The ALARM monitoring system," in *Proc. 2nd Eur. Conf. Artif. Intell. Med.*, 1989, pp. 247–256.
[32] J. Binder, D. Koller, S. Russell, and K. Kanazawa, "Adaptive probabilistic networks with hidden variables," *Mach. Learn.*, vol. 29, no. 2/3, pp. 213–244, Nov. 1997.
[33] N. Friedman, "Inferring cellular networks using probabilistic graphical models," *Science*, vol. 303, no. 5659, pp. 799–805, Feb. 2004.
[34] X. Wu, Y. Ye, and K. R. Subramanian, "Interative analysis of gene interactions using graphical Gaussian models," in *Proc. 3rd ACM SIGKDD Workshop Data Mining Bioinformatics, BIOKDD*, 2003, pp. 63–69.
[35] S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano, "Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks," *J. Bioinformatics Comput. Biol.*, vol. 2, no. 1, pp. 77–98, 2004.
[36] A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelić, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem, and P. Bühlmann, "Sparse graphical Gaussian modeling of the isoprenoid gene network in *Arabidopsis thaliana*," *Genome Biol.*, vol. 5, no. 11, pp. R92.1–R92.13, 2004.
[37] H. Li and J. Gui, "Gradient directed regularization for sparse Gaussian concentration graphs, with applications to inference of genetic networks," *Biostatistics*, vol. 7, no. 2, pp. 302–317, 2006.
[38] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max–min hill-climbing Bayesian network structure learning algorithm," *Mach. Learn.*, vol. 65, no. 1, pp. 31–78, Oct. 2006.

Diego Vidaurre received the M.S. degree in computer engineering from Rey Juan Carlos University, Madrid, Spain, in 2005. He is currently working toward the Ph.D. degree at the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, under the supervision of Pedro Larrañaga and Concha Bielza.

His first publication appeared in the IEEE TRANSACTIONS ON NEURAL NETWORKS in 2007. His areas of interest are probabilistic graphical models, neural networks, machine learning, and neuroscience.

Concha Bielza received the M.S. degree in mathematics from Complutense University of Madrid, Madrid, Spain, in 1989 and the Ph.D. degree in computer science from the Universidad Politécnica de Madrid, Madrid, in 1996.

She is currently an Associate Professor of statistics and operations research with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid. Her research interests are primarily in the areas of probabilistic graphical models, decision analysis, metaheuristics for optimization, data mining, classification models, and real applications. Her research has appeared in journals like *Management Science*, *Computers and Operations Research*, *Statistics and Computing*, the *European Journal of Operational Research*, *Decision Support Systems*, *Naval Research Logistics*, the *Journal of the Operational Research Society*, *Bioinformatics*, *Medical Decision Making*, *Methods of Information in Medicine*, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and *Expert Systems with Applications* and as chapters of many books.



Pedro Larrañaga received the M.S. degree in mathematics from the University of Valladolid, Valladolid, Spain, in 1981 and the Ph.D. degree in computer science from the University of the Basque Country, San Sebastian, Spain, in 1995.

He is currently a Professor of computer science and artificial intelligence with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain. He has published three edited books and over 50 refereed journal papers. His main interests are in the areas of probabilistic graphical models, machine learning, evolutionary computation, bioinformatics, and neuroscience.