# An Iterated Local Search Algorithm for Learning Bayesian Networks with Restarts Based on Conditional Independence Tests

Luis M. de Campos,[1,*] Juan M. Fernández-Luna,[2,†] J. Miguel Puerta[3,‡]
[1]*University of Granada, Department of Ciencias de la Computación e I.A., 18071—Granada, Spain*
[2]*University of Jaén, Department of Informática, 23071—Jaén, Spain*
[3]*University of Castilla-La Mancha, Department of Informática, 02071—Albacete, Spain*

A common approach for learning Bayesian networks (BNs) from data is based on the use of a scoring metric to evaluate the fitness of any given candidate network to the data and a method to explore the search space, which usually is the set of directed acyclic graphs (DAGs). The most efficient search methods used in this context are greedy hill climbing, either deterministic or stochastic. One of these methods that has been applied with some success is hill climbing with random restart. In this article we study a new algorithm of this type to restart a local search when it is trapped at a local optimum. It uses problem-specific knowledge about BNs and the information provided by the database itself (by testing the conditional independencies, which are true in the current solution of the search process). We also study a new definition of neighborhood for the space of DAGs by using the classical operators of arc addition and arc deletion together with a new operator for arc reversal. The proposed methods are empirically tested using two different domains: ALARM and INSURANCE. © 2003 Wiley Periodicals, Inc.

## 1. INTRODUCTION

Bayesian networks (BNs) are knowledge representation tools able to manage efficiently the dependence/independence relationships among the random variables that compose the problem domain we want to model. This representation has two components: (a) a graphical structure, more precisely a directed acylic graph (DAG) and (b) a set of parameters, which together specify a joint probability distribution over the random variables.[1,2] In BNs, the graphical structure represents

*e-mail: lci@decsai.ugr.es
†e-mail: jmfluna@ujaen.es
‡Author to whom all correspondence should be addressed. e-mail: jpuerta@info-ab.uclm.es

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 18, 221–235 (2003)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.10085

dependence and independence relationships. The numerical component is a collection of conditional probability measures, which shape the relationships.

Once we have the BN specified, it constitutes an efficient device to perform inference tasks. However, the previous problem of building such a network still remains. Therefore, an interesting task is to develop automatic methods capable of learning the network directly from data, as an alternative or a complement to the method of eliciting opinions from experts. Algorithms for learning (the structure of) BNs have been studied, basically, from two points of view: methods based on conditional independence test[3–6] and methods based on a scoring metric optimization,[7–9] although there also exist algorithms that use a combination of these two methods.[10–12] In this study we consider learning methods based on a scoring metric that also use conditional independence tests, so they can be classified as hybrid methods.

Because learning BNs are, in general, an NP-hard problem,[13] we have to solve it with heuristic methods. Most existing scoring-based learning algorithms apply standard heuristic search techniques such as greedy hill climbing (HCST), simulated annealing, genetic algorithms, etc. In this article, we focus on local search algorithms, more precisely, hill climbing with random restart (HCSTRR). These methods examine only possible local changes at each step and apply the one that leads to the greatest improvement in the scoring metric. When the search process is blocked at a local optimum, it is restarted on the basis of a random modification of the current optimum. Our proposals try to improve these methods in two different ways: first, by designing a methodology based on the use of conditional independence tests to guide the restarts and, second, by modifying the neighborhood structure, i.e., changing the classical operators used to move in the space of DAGs.

The article is structured as follows: We begin in Section 2 with the preliminaries. In Section 3, we explain how to use information about the conditional independencies, which are true in the current solution of the search process to restart it, and describe the proposed hybrid learning algorithm called I-map restart (IMAPR). The experimental results obtained by this algorithm on two different domains (ALARM[14] and INSURANCE[15]) are also described and compared with the ones obtained by HCST and HCSTRR. In Section 4 we define the modified neighborhood structure of the DAG space and show the corresponding experimental results when IMAPR and HCST are equipped with the new operators. Finally, Section 5 contains the concluding remarks and some proposals for future research.

## 2. PRELIMINARIES

In this section we briefly review BNs and procedures to learn them. A BN is a graphical representation of a joint probability distribution. It comprises two components: (1) a DAG $G = (\mathbf{V}, E)$, where $\mathbf{V} = \{x_1, x_2, \ldots, x_n\}$, the set of nodes, represents the system variables and $E$, the set of arcs, represents the dependence relationships among the variables; (2) a set of parameters is also stored for each variable in $\mathbf{V}$, usually conditional probability distributions. For each

variable $x_i \in \mathbf{V}$ we have a family of conditional distributions $P(x_i|\mathrm{Pa}(x_i))$, where $\mathrm{Pa}(x_i)$ represents the parent set of the variable $x_i$ in $G$.

The conditional independence relationships represented in a DAG $G$, denoted as $\langle X, Y|Z\rangle_G$, where $X$, $Y$, and $Z$ are disjoint subsets of variables in $\mathbf{V}$, are graphically encoded through the $d$-separation criterion,[2] by means of the presence or absence of direct connections between pairs of variables. Given a joint probability distribution $P$ over $\mathbf{V}$, $X$, and $Y$ are probabilistically independent given $Z$ and denoted as $I(X, Y|Z)$, if and only if the following statement holds: $P(x|yz) = P(\mathbf{x}|\mathbf{z})$, $\exists \mathbf{x}, \mathbf{y}, \mathbf{z}$ such that $P(\mathbf{yz}) > 0$, where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ denote a configuration for the subsets of variables $X$, $Y$, and $Z$, respectively. A DAG $G$ is an I-map of a probability distribution $P$ if $\langle X, Y|Z\rangle_G \Rightarrow I(X, Y|Z)$, and is *minimal* if no arc can be eliminated from $G$ without violating the I-map condition. $G$ is a D-map of $P$ if $\langle X, Y|Z\rangle_G \Leftarrow I(X, Y|Z)$. When a DAG $G$ is both an I-map and a D-map of $P$, it is said that $G$ and $P$ are *isomorphic* models. It is always possible to build a minimal I-map of any given probability distribution $P$, but some distributions do not admit an isomorphic model.[2]

The problem of learning a BN can be stated as follows: given a *training set* $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ of instances of $\mathbf{V}$, find the BN that, in some sense, best matches $D$. The common approach to this problem is to introduce a scoring function $f$ that evaluates each network with respect to the training data and then to search for the best network according to this score. Different Bayesian and non-Bayesian scoring metrics can be used.[7–10] The alternative approach, constraint-based, is to search for the network satisfying as many independencies present in the data as possible.[4–6] Obviously, the decision about which conditional independencies are either true or false is made by means of statistical tests.

A desirable and important property of a scoring metric is it decomposability in the presence of full data, i.e., the scoring function can be decomposed in the following way:

$$f(G : D) = \sum_{i=1}^{n} f(x_i|\mathrm{Pa}(x_i) : N_{x_i,\mathrm{Pa}(x_i)}) \tag{1}$$

where $N_{x_i,\mathrm{Pa}(x_i)}$ are the statistics of the variables $x_i$ and $\mathrm{Pa}(x_i)$ in $D$, i.e., the number of instances in $D$ that match each possible instantiation of $x_i$ and $\mathrm{Pa}(x_i)$. The decomposition of the metric is very important for the learning task: a local search procedure that changes one arc at each move can efficiently evaluate the improvement obtained by this change. Such a procedure can reuse the computations made at previous stages (only the statistics corresponding to the variables in which its parent set has been modified need to be recomputed). An example is a greedy hill-climbing method that at each step performs the local change that yields in the maximal gain until it reaches a local maximum. Because this procedure is trapped in the first local maximum it reaches, several methods for avoiding this situation have been used, such as stochastic hill climbing, simulated annealing, tabu search, etc. The main representative of stochastic hill climbing is HCSTRR, which has been used by several authors[8] with relative success. This fact has motivated us to

try a new search method based on the same principles as the previous one but with a more systemic restart if the current local search does not improve the best current maximum.

## 3. LOCAL SEARCH WITH RESTARTS IN THE SPACE OF DAGs

Let us assume that we have a database of cases $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$, and a decomposable scoring metric for measuring the fitness of any candidate DAG $G$ to the data $f(G : D)$. Let $\mathcal{G}_n$ be the family of all the DAGs with $n$ nodes. Then, the problem that we want to solve is

$$\text{find} \quad G^* = \arg\max_{G \in \mathcal{G}_n} f(G : D) \tag{2}$$

The efficiency of the algorithms that search in the space of DAGs using local methods is caused by mainly the property of decomposition that many metrics exhibit, as we have mentioned previously. The standard choices for local changes in the space of DAGs (which can be evaluated locally) are arc addition, arc deletion, and arc reversal. In an HCSTRR, once we have reached a local maximum, it is randomly modified by applying a number of feasible local transformations (avoiding the inclusion of directed cycles in the graph) before firing a new hill climbing. If the number of transformations performed is high, then the rate of computations reused from previous stages will be poor, thus wasting the advantage provided by the decomposability property. Moreover, probably the perturbed network will be quite complex (in the sense that the average number of parents per variable is high, and even that some node has a very high number of parents) and then the new statistics to be computed will be more complex, thus rendering the new hill climbing even more expensive. On the contrary, if the number of transformations performed is low, then we will have the risk of being unable to escape from the current local maximum. For these reasons this number of transformations is a parameter of difficult adjustment.

### 3.1. A Hybrid Learning Algorithm with Restarts Based on Conditional Independence Tests

We will study an algorithm that restarts a local search for a new promising starting point by taking advantage of problem-specific knowledge about BNs and using the information provided by the database itself. The specific transformation will be based on conditional independence tests. Acid and de Campos[16] studied a way to obtain the minimum $d$-separating subset for two nodes in a DAG in an efficient way. In the previous section we introduced the concepts of I-map and minimal I-map. These are the elements necessary to describe our hybrid algorithm. The result stated in the next proposition, in which the proof follows immediately from the definition of I-map, is the basis of the proposed transformation.

PROPOSITION 1. *Let G be a DAG that is an I-map of a joint probability distribution P. For each pair of nodes $x_i$ and $x_j$ nonadjacent in G, let $S_G(x_i, x_j)$ be the minimum subset d-separating $x_i$ and $x_j$ in G. Then, $I(x_i, x_j | S_G(x_i, x_j))$ holds.*

Our idea is to check whether the DAG obtained as the new local maximum is an I-map. To do this, we will perform all the possible conditional independence tests between pairs of nonadjacent nodes in the DAG using the minimum $d$-separating subset; if we find that some pairs of variables are not conditionally independent, then the DAG will not be an I-map; therefore, we would have a reason to continue looking for another (better) local maximum.

The method comprises the following steps: for each pair of variables $x_i$ and $x_j$ in $G$, if they are not adjacent and the independence test $I(x_i, x_j|S_G(x_i, x_j))$? fails, then we could add an arc linking these variables to $G$; on the other hand, if $x_i$ and $x_j$ are adjacent and the independence test $I(x_i, x_j|S_{G\setminus\{x_i-x_j\}}(x_i, x_j))$? does not fail, then we could remove the arc linking $x_i$ and $x_j$ from $G$.

Note that none of these two actions guarantees that the transformed DAG is a minimal I-map; however, the addition of an arc linking nodes that are dependent is necessary if we want to obtain an I-map, and the removal of an arc linking independent nodes may be useful to get a minimal I-map. We must also note that the insertion/deletion of an arc in $G$ at some step, as a consequence of performing a test, may have influence on the results of the remaining tests (because the $d$-separating sets may change). Therefore, the ordering for the conditional independence tests will be important for the transformation. Moreover, it is necessary to perform all the possible tests ($2n^2$), including the inverse tests $I(x_i, x_j|S_G(x_i, x_j))$? and $I(x_j, x_i|S_G(x_j, x_i))$?.[a] We will clarify these questions using the following example.

*Example 1.* Let us assume that the true network to be learned is $x_1 \rightarrow x_2 \rightarrow x_3$ and the current local maximum DAG is the empty graph; we also suppose that the order to perform the independence tests is the lexicographical order among the variables. First, the test $I(x_1, x_2|\emptyset)$? is performed and it fails; then we add a link, e.g., $x_1 \rightarrow x_2$; next, the test $I(x_1, x_3|\emptyset)$? is performed and it also fails; then, we add a link, e.g., $x_1 \rightarrow x_3$. For $x_2$, first, we perform the test $I(x_2, x_1|\emptyset)$?, and the result is negative; the following test is $I(x_2, x_3|x_1)$? and again it is negative; so we add a link, e.g., $x_2 \rightarrow x_3$. For $x_3$, the first test is $I(x_3, x_1|x_2)$? and it is positive; so we delete the link $x_1 \rightarrow x_3$; finally, the test $I(x_3, x_2|\emptyset)$? is performed and it is negative. In this case we get the target network. If the order to perform the tests was $x_3x_1x_2$, then we would obtain the complete graph, which is also an I-map of the true model although with a higher number of arcs.

We must note that not only the ordering of the independence tests is a free parameter in the transformation scheme, but also the direction of each new added link is undetermined. These two parameters are selected randomly, thus preventing a cyclic behavior of the algorithm. The pseudocode of the complete learning algorithm, which we call IMAPR, is displayed in Figure 1.

[a]This is necessary if the minimum $d$-separating subsets $S_G(x_i, x_j)$ and $S_G(x_j, x_i)$ are not equal.

1. *Initialization: Select an initial solution, G.*
2. *Repeat until performing a maximum number of iterations:*
   (a) $G' = Hill\text{-}Climbing(G)$; $G := G'$.
   (b) *Select a random ordering* $\sigma$ *for the variables in* $\mathbf{V}$.
   (c) *Repeat until all the conditional independence tests have been carried out:*
      i. *Select* $x_i$ *and* $x_j$ *according to the previous ordering* $\sigma$.
      ii. *If these variables are not adjacent in G then*
         A. *Compute the subset* $S = S_G(x_i, x_j)$.
         B. *If* $\neg I(x_i, x_j|S)$, *then adds to G an arc linking* $x_i$ *and* $x_j$ *with random direction (without forming directed cycles).*
      iii. *If these variables are adjacent in G then*
         A. *Compute the subset* $S = S_{G\setminus\{x_i - x_j\}}(x_i, x_j)$.
         B. *If* $I(x_i, x_j|S)$, *then delete the arc linking* $x_i$ *and* $x_j$ *from G.*

**Figure 1.** Pseudocode of the IMAPR algorithm.

### 3.2. Experimentation

In this section, we will evaluate experimentally the proposed algorithm. We have selected two test domains: ALARM[14] and INSURANCE.[15] For ALARM we have executed our algorithm for the first 3,000 and 10,000 cases of the ALARM database (which contains 20,000 cases). For INSURANCE, we have generated three databases with 10,000 cases each by means of probabilistic logic sampling (in this case we show the average of the results obtained for the three databases). In all the cases we have executed the IMAPR algorithm with a fixed number of iterations equal to 15; we have used the K2 metric (logarithmic version),[7] which is decomposable, and an empty network as the initial solution.

The conditional independence tests have been performed using an hypothesis test based on the $\chi^2$ distribution. It is important to remark that two different significance levels have been used in the tests: For the cases in which there is an arc (either $x_i \rightarrow x_j$ or $x_j \rightarrow x_i$) in the DAG, we perform the test $I(x_i, x_j|S_{G\setminus\{x_i - x_j\}}(x_i, x_j))$? using a fixed level of significance of 0.01; in these cases we are demanding a strong empirical confirmation of the dependence between $x_i$ and $x_j$ displayed by the DAG (thus facilitating the elimination of the arc otherwise). On the other hand, for the cases in which the arc does not exist, we perform the test $I(x_i, x_j|S_G(x_i, x_j))$? with a fixed level of significance equal to 0.25; so, we do not demand a strong empirical evidence about the dependence between $x_i$ and $x_j$, facilitating in this way the insertion of the arc. In both cases the idea is to favor the modification of the current structure, provided that this modification is not clearly in contradiction with the available data. These two parameters have not been fitted by preliminary experimentation.

The following performance measures have been computed:

(1) Measures of quality (effectiveness) of the learned network: (K2) value of the metric K2(log); Kullback-Leibler (KL) distance between the probability distributions associ-

**Table I.** Results for HCST.

| HCST | ALARM-3000 | ALARM-10000 | INSURANCE |
|---|---|---|---|
| K2 | −14,425.62 | −47,267.11 | −57,998.10 |
| KL | 9.220 | 9.266 | 8.423 |
| A | 6 | 11 | 10.33 |
| D | 4 | 4 | 11.67 |
| I | 3 | 6 | 7.67 |
| It | 1 | 1 | 1 |
| EstEv | 3,375 | 3,280 | 2,050.33 |
| TEst | 1.54E + 05 | 1.47E + 05 | 7.66E + 0.4 |
| NVars | 2.99 | 2.99 | 3.09 |

ated with the database and to the learned network;[b] and the number of arcs added (A), deleted (D), and inverted (I) when we compare the result of the algorithm with the true network.

(2) Measures of efficiency of the algorithm: the number of iterations (It) performed by the algorithm until obtaining the best network, number of different statistics evaluated (EstEv) during the execution of the algorithm, total number of statistics (TEst) used by the algorithm,[c] and average number of variables (NVars) that intervene in the computed statistics.

To compare the behavior of the IMAPR algorithm, we have also performed experiments with an HCST and an HCSTRR algorithm. For HCSTRR, the number of random perturbations has been set to 125 and the number of iterations is also 15. We have limited the maximum number of possible parents for each node to eight.[d]

The results of the experiments are shown in Tables I (HCST), II (IMAPR), and III (HCSTRR).[e] For the nondeterministic algorithms IMAPR and HCSTRR, the displayed values represent the average $\bar{f}$ and the standard deviation $s_{\bar{f}}$ of 10 executions, although we also give information about the best individual found in all the executions.

With respect to the efficiency of the algorithms, we are going to focus on the EstEv and NVars. Most of the running time of the scoring-based learning algorithms is spent in the evaluation of statistics, and this time increases exponentially with the number of variables involved. Therefore, an approximate measure of the time complexity of an algorithm is EstEv $* 2^{\text{NVars}}$ (for IMAPR we have to add to EstEv the number of different statistics computed that are necessary to perform the

[b]Actually, we have calculated a monotonic transformation of this measure, which is much easier to compute.[7] Its interpretation is the higher the KL value, the better the network.

[c]Note that this number can be considerably greater than EstEv. By using hashing techniques, we can store and efficiently retrieve any statistics previously calculated so that it is not necessary to recompute them. This avoids many accesses to the database and improves the efficiency.

[d]Without imposing such a limit the algorithm is extremely slow because the computation of the statistics associated to nodes with a high number of parents is very costly.

[e]In this case, the experiments with ALARM have been performed only with the database of 3,000 cases.

**Table II.** Results for IMAPR.

| | ALARM-3000 | | | ALARM-10000 | | | INSURANCE | | |
|---|---|---|---|---|---|---|---|---|---|
| IMAPR | $\bar{f}$ | $s_{\bar{f}}$ | Best | $\bar{f}$ | $s_{\bar{f}}$ | Best | $\bar{f}$ | $s_{\bar{f}}$ | Best |
| K2 | −14,416.37 | 7.39 | −14,405.12 | −47,246.15 | 48.73 | −47,111.03 | −57,912.43 | 40.31 | −57,842.76 |
| KL | 9.225 | 0.003 | 9.231 | 9.266 | 0.002 | 9.268 | 8.439 | 0.006 | 8.465 |
| A | 5.00 | 2.16 | 3 | 9.70 | 2.58 | 3 | 9.73 | 0.88 | 3 |
| D | 2.70 | 1.16 | 1 | 3.60 | 0.70 | 2 | 11.20 | 0.35 | 8 |
| I | 2.60 | 0.52 | 2 | 4.90 | 1.52 | 2 | 8.93 | 1.79 | 3 |
| It | 10.60 | 4.88 | | 5.10 | 5.24 | | 8.80 | 4.04 | |
| EstEv | 12,485.70 | 452.62 | | 14,570.60 | 391.98 | | 5,940.00 | 291.01 | |
| TEst | 1.07E + 06 | 4.35E + 04 | | 1.24E + 06 | 3.07E + 04 | | 3.97E + 05 | 2.14E + 04 | |
| NVars | 4.24 | 0.09 | | 4.46 | 0.04 | | 4.41 | 0.07 | |

Fifteen iterations.

**Table III.** Results for HCSTRR.

| | ALARM-3000 | | | INSURANCE | | |
|---|---|---|---|---|---|---|
| HCSTRR | $\bar{f}$ | $s_{\bar{f}}$ | Best | $\bar{f}$ | $s_{\bar{f}}$ | Best |
| K2 | −14,413.20 | 6.64 | −14,403.28 | −57,918.90 | 36.95 | −57,872.20 |
| KL | 9.230 | 0.006 | 9.230 | 8.435 | 0.036 | 8.400 |
| A | 5.80 | 2.25 | 2 | 8.78 | 3.11 | 7 |
| D | 2.10 | 0.99 | 1 | 11.44 | 2.19 | 9 |
| I | 2.30 | 2.67 | 0 | 6.44 | 3.54 | 3 |
| It | 7.80 | 5.18 | | 8.78 | 4.94 | |
| EstEv | 44,381.60 | 1,116.15 | | 27,610.56 | 841.26 | |
| TEst | 3.12E + 06 | 9.79E + 04 | | 1.41E + 06 | 3.31E + 04 | |
| NVars | 5.40 | 0.06 | | 6.06 | 0.10 | |

Fifteen iterations.

independence tests. Assuming that all these statistics are different, an upper bound for this number is 18,648 for ALARM and 9,828 for INSURANCE).

We can observe that IMAPR improves considerably the results obtained by HCST in all the cases, especially with respect to the value of the metric [note that we are using log(K2) as the metric, so that the differences are much greater in a nonlogarithmic scale] but also with respect to KL and the number of erroneous arcs (A + D + I). The K2 values of the true networks for the corresponding databases, which may serve as a kind of scale, are −14,412.69 for ALARM-3000, −47,086.57 for ALARM-10000, and −58,120.95 for INSURANCE.

It can also be observed that IMAPR and HCSTRR obtain very similar results with respect to the quality of the learned networks. In fact, a Mann-Whitney test did not find significant differences in any case. However, IMAPR is significantly more efficient than HCSTRR in all the cases (using the same test). In our experiments, IMAPR is roughly twice as fast as HCSTRR, taking into account the independence tests. Obviously, both IMAPR and HCSTRR are much slower than HCST (around 20 and 40 times, respectively).

## 4. REDEFINING THE NEIGHBORHOOD FOR LOCAL SEARCH IN THE SPACE OF DAGs

It is somewhat surprising for us that the IMAPR algorithm only improves HCSTRR in terms of efficiency but not in terms of effectiveness, despite the fact that IMAPR performs informed restarts instead of completely random restarts. By monitoring the algorithm as it progresses, we have identified some situations in which it gets into trouble. Let us explain these situations by means of the following example.

*Example 2.* Consider the network with four variables displayed in Figure 2(a). From this network we have generated a database containing 1,000 instances of these variables. The value of the K2 metric for this database and this network is also shown in the figure. The IMAPR algorithm could obtain the network in Figure

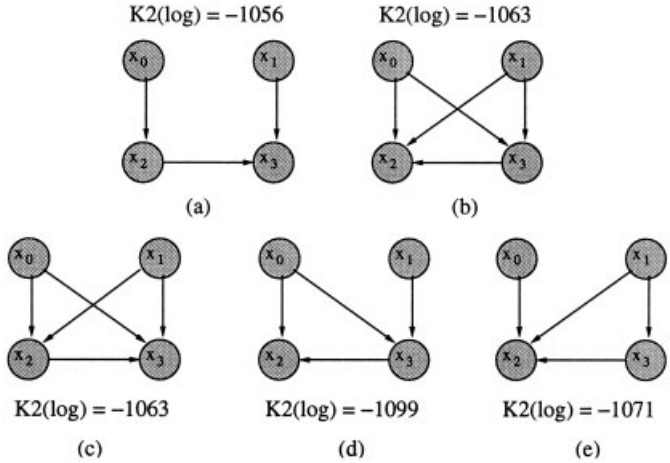


**Figure 2.** Problems with the classical neighborhood when an arc is wrongly directed.

2(b) (e.g., with the ordering $\{x_3, x_2, x_1, x_0\}$). None of the possible transformations of this network, using the classical operators of addition, deletion, and reversal of an arc, improves the K2 value [the best neighbors are displayed in Figure 2(c)–(e)] so that it is a local maximum. In fact, the DAG in Figure 2(b) is a minimal I-map of the model in Figure 2(a).

This behavior is also common for many learning algorithms. When we mistake the direction of some arc connecting two nodes (and this situation may be quite frequent), then the algorithms tend to cross the parents of these nodes to compensate the wrong orientation; the resultant configuration is quite stable, in the sense that no local transformation produces a better network and, therefore, it may be difficult to escape from this local maximum. In the previous example, the IMAPR algorithm is doing its work properly, but the wrong orientation of the arc linking the variables $x_2$ and $x_3$ prevents it from finding a better minimal I-map.

A possible solution for this problem could be to perform the search process in a space different from the space of DAGs. For example, we could use the space of equivalence classes of DAGs,[5] by defining specific operators for this space. The problem is that these operators do not take advantage of the decomposability property of the metrics, and the corresponding algorithms lose efficiency.

Recently, the possibility of searching in the DAG space but including some characteristics relative to equivalence classes of DAGs has been considered.[17] This method uses an operator called the repeated covered arc reversal (RCAR), which iteratively inverts a prefixed (random) number of *covered* arcs.[f] Then, an HCST algorithm is fired to obtain a new local maximum, and the whole process is iterated a fixed number of times.

[f]An arc $x_i \rightarrow x_j$ in a DAG $G$ is covered if $\mathrm{Pa}(x_j) = \mathrm{Pa}(x_i) \cup \{x_i\}$. A covered arc may be inverted and the resultant DAG is equivalent to $G$.

### 4.1. A New Operator for Arc Reversal

We are going to introduce a new operator for searching in the space of DAGs, which has some similarities with the idea behind RCAR. The classical notion of neighborhood of a DAG $G = (V, E)$ used for searching in the space of DAGs, can be formalized as follows:

$$\mathcal{N}_A(G) = \{(V, E') | E' = E \cup \{x_j \rightarrow x_i\}, x_j \rightarrow x_i \notin E \quad \text{and} \quad (V, E') \text{ is a DAG}\}$$

$$\mathcal{N}_D(G) = \{(V, E') | E' = E \backslash \{x_j \rightarrow x_i\}, x_j \rightarrow x_i \in E\}$$

$$\mathcal{N}_I(G) = \{(V, E') | E'$$

$$= (E \cup \{x_j \rightarrow x_i\}) \backslash \{x_i \rightarrow x_j\}, x_i \rightarrow x_j \in E \quad \text{and} \quad (V, E') \text{ is a DAG}\} \quad (3)$$

The classical neighborhood of $G$ is then $\mathcal{N}(G) = \mathcal{N}_A(G) \cup \mathcal{N}_D(G) \cup \mathcal{N}_I(G)$. Our proposal implies modification of the reversal operator and, therefore, the redefinition of the neighborhood $\mathcal{N}_I(G)$, as follows:

$$\begin{aligned} \mathcal{N}_I(G) = \{(V, E') | E' \\ = (((E \cup \{x_j \rightarrow x_i\}) \backslash (\{x_i \rightarrow x_j\} \cup P_{\text{old}}^{x_j, x_i})) \cup P_{A,B}^{x_j, x_i}) x_i \rightarrow x_j \in E, \\ A, B \subseteq (\text{Pa}(x_i) \cup \text{Pa}(x_j)) \backslash \{x_i\}, \quad \text{and} \quad (V, E') \text{ is a DAG} \end{aligned} \quad (4)$$

where

$$P_{\text{old}}^{x_j, x_i} = \begin{cases} \{x_p \rightarrow x_j | x_p \in \text{Pa}(x_j)\} \cup \{x_q \rightarrow x_i | x_q \in \text{Pa}(x_i)\} & \text{if } \text{Pa}(x_i) \cap \text{Pa}(x_j) \neq \varnothing \\ \varnothing & \text{if } \text{Pa}(x_i) \cap \text{Pa}(x_j) = \varnothing \end{cases} \quad (5)$$

and

$$P_{A,B}^{x_j, x_i} = \begin{cases} \{x_p \rightarrow x_j | x_p \in A\} \cup \{x_q \rightarrow x_i | x_q \in B\} & \text{if } \text{Pa}(x_i) \cap \text{Pa}(x_j) \neq \varnothing \\ \varnothing & \text{if } \text{Pa}(x_i) \cap \text{Pa}(x_j) = \varnothing \end{cases} \quad (6)$$

This new definition of $\mathcal{N}_I(G)$ states that for any given arc in $E$, if its extreme nodes share some parent, then we delete the current parents of both nodes, invert the arc, and add as the new parents for each node any subset of the old parents of either node. The idea is to give the algorithm the opportunity of uncrossing the parents of two nodes that have been connected in the wrong direction, without being limited to move only one arc every time.

Note that for any arc $x_i \rightarrow x_j \in E$, the number of possible reversals is now $O(2^{2k})$, with $k = |\text{Pa}(x_i) \cup \text{Pa}(x_j)| - 1$, instead of only one. Therefore, the number of neighbors of a DAG may increase exponentially. Nevertheless, in practice, the number of parents of a node is not usually high; therefore, we expect that the computational cost will not be excessive.[g] In any case, we could limit the

[g]Our experiments in the next section support this assertion.

**Table IV.** Results for HCST2.

| HCST2 | ALARM-3000 | ALARM-10000 | INSURANCE |
|---|---|---|---|
| K2 | −14,414.55 | −47,111.03 | −57,934.61 |
| KL | 9.225 | 9.268 | 8.427 |
| A | 4 | 3 | 7.67 |
| D | 2 | 2 | 11.00 |
| I | 2 | 2 | 7.67 |
| It | 1 | 1 | 1 |
| EstEv | 3,336 | 3,327 | 2,169.34 |
| TEst | 1.56E + 05 | 1.53E + 05 | 8.32E + 04 |
| NVars | 2.93 | 2.94 | 3.08 |

cardinality of the new parent sets (which is a common practice for other algorithms[7,18]).

### 4.2. Experimentation

To test the merits of the new definition of neighborhood, we have implemented a hill-climbing algorithm (HCST2) using the new operator for arc reversal (as well as the operators of arc addition and arc removal) and a version of IMAPR also using this operator (IMAPR2). In the light of the superior behavior of IMAPR with respect to HCSTRR, in terms of efficiency and without losing effectiveness, we decided not to experiment with HCSTRR2. We have used the same domains and databases considered in Section 3.2.

The results obtained by HCST2 are shown in Table IV. We can see that the improvement with respect to the results in Table I is quite considerable in terms of effectiveness. On the other hand, with respect to efficiency, the values of EstEv and NVars are quite similar for HCST and HCST2, so that the running times of these two algorithms are comparable. The results offered by IMAPR2, displayed in Table V, can be qualified as excellent; IMAPR2 outperforms all the other algorithms in effectiveness (in all the cases, a Mann-Whitney test found significant differences). The low standard deviations obtained are noteworthy, so that this algorithm seems to be quite stable. Note also that IMAPR2 is not much slower than IMAPR. In fact, it is faster than IMAPR for INSURANCE and slower for ALARM (in both cases there are significant differences).

## 5. CONCLUSIONS

In this article we have proposed a new algorithm for learning BNs that uses a hybrid methodology: The algorithm tries to optimize a scoring metric, but there is another stage in which it carries out conditional independence tests. Our algorithm can also be included within the methods that use a hill-climbing search with multiple restarts, which is our case are partially guided by problem-specific knowledge. In this way, these restarts do not depend on parameters of difficult adjustment (as the number of random transformations or the maximum number of

**Table V.** Results for IMAPR2.

| | ALARM-3000 | | | ALARM-10000 | | | INSURANCE | | |
|---|---|---|---|---|---|---|---|---|---|
| IMAPR2 | $\bar{f}$ | $s_{\bar{f}}$ | Best | $\bar{f}$ | $s_{\bar{f}}$ | Best | $\bar{f}$ | $s_{\bar{f}}$ | Best |
| K2 | −14,402.41 | 1.20 | −14,401.29 | −47,078.51 | 0.78 | −47,078.26 | −57,850.96 | 10.14 | −57,842.76 |
| KL | 9.231 | 0.001 | 9.231 | 9.274 | 0.000 | 9.274 | 8.447 | 0.033 | 8.465 |
| A | 1.70 | 0.48 | 2 | 0.30 | 0.48 | 1 | 7.17 | 3.89 | 3 |
| D | 1.00 | 0.00 | 1 | 1.10 | 0.32 | 1 | 10.10 | 1.83 | 8 |
| I | 0.40 | 0.84 | 0 | 0.10 | 0.32 | 0 | 6.83 | 3.39 | 3 |
| It | 8.00 | 3.40 | | 5.60 | 4.50 | | 4.70 | 3.07 | |
| EstEv | 16,489.80 | 1,401.51 | | 19.175.70 | 1,823.15 | | 4,992.80 | 447.41 | |
| Test | 2.46E + 06 | 8.80E + 05 | | 3.44E + 06 | 9.21E + 05 | | 5.83E + 05 | 1.16E + 05 | |
| NVars | 4.45 | 0.12 | | 4.58 | 0.10 | | 3.94 | 0.08 | |

Fifteen iterations.

parents per variable). We have also proposed a new definition of neighborhood for the space of DAGs that copes with the problems that local search algorithms encounter when some arcs are wrongly oriented.

The experimentation performed with the ALARM and INSURANCE domains supports the conclusion that our methods, particularly the combination of the IMAPR algorithm with the modified neighborhood, improve considerably the results offered by some state-of-the-art learning algorithms.

We want to perform a more systematic experimentation and comparative analysis using other, real-world and synthetic domains. For future research, we also plan to apply our hybrid method to search for the best BN using the space of orderings of the variables[19] instead of the space of DAGs. Several variants of the new operator for arc reversal defined in this work (e.g., using the intersection of the parent sets in place of their union) could be studied too.

## Acknowledgments

This work has been supported by the Spanish Ministerio de Ciencia y Tecnología (MCYT) under projects TIC2001-2973-CO5-01 and TIC2001-2973-CO5-05.

## References

1. Jensen, FV. An introduction to Bayesian networks. London: UCL Press, 1996.
2. Pearl J. Probabilistic reasoning in intelligent systems: Networks of plausible inference. San Mateo, CA: Morgan Kaufmann; 1988.
3. de Campos LM. Independency relationships and learning algorithms for singly connected networks. J Exp Theor Artif Intell 1998;10(4):511–549.
4. de Campos LM, Huete JF. A new approach for learning belief networks using independence criteria. Int J Approx Reason 2000;24:11–37.
5. Pearl J, Verma TS. Equivalence and synthesis of causal models. In: Proc 6th Conf on Uncertainty in Artificial Intelligence. 1990. pp 220–227.
6. Spirtes P, Glymour C, Scheines R. Causation, prediction, and search. Lecture Notes in Statistics 81. Berlin: Springer Verlag; 1993.
7. Cooper GF, Herskovits E. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 1992;9(4):309–348.
8. Heckerman D, Geiger D, Chickering DM. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 1995;20:197–244.
9. Lam W, Bacchus F. Learning Bayesian belief networks. An approach based on the MDL principle. Computational Intell 1994;10(4):269–293.
10. Acid S, de Campos LM. A hybrid methodology for learning belief networks: Benedict. Int J Approx Reason 2001;27(3):235–262.
11. Dash D, Druzdel M. A hybrid anytime algorithm for the construction of causal models from sparse data. In: Proc 15th Conf on Uncertainty in Artificial Intelligence. San Mateo, CA: Morgan Kaufmann; 1999. pp 142–149.
12. Singh M, Valtorta M. Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. Int J Approx Reason 1995;12:111–131.
13. Chickering DM, Geiger D, Heckerman D. Learning Bayesian networks is NP-Complete. In: Fisher D, Lenz H, editors. Learning from data: Artificial intelligence and statistic V. Berlin: Springer-Verlag; 1996. pp 121–130.

14. Beinlich IA, Suermondt HJ, Chavez RM, Cooper GF. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief network. In: Proc 2nd European Conf on Artificial Intelligence in Medicine. Berlin: Springer-Verlag; 1989. pp 247–256.
15. Binder J, Koller D, Russel S, Kanazawa K. Adaptive probabilistic networks with hidden variables. Machine Learning 1997;29(2):213–244.
16. Acid S, de Campos LM. An algorithm for finding minimum d-separating sets in belief networks, In: Proc 12th Conf Uncertainty in Artificial Intelligence. San Mateo, CA: Morgan Kaufmann; 1996. pp 3–10.
17. Kocka T, Castelo R. Improved learning of Bayesian networks. In: Proc 17th Conf on Uncertainty in Artificial Intelligence. San Mateo, CA: Morgan Kaufmann; 2001. pp 269–276.
18. Larrañaga P, Poza M, Yurramendi Y, Murga R, Kuijpers C. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. IEEE Trans Pattern Anal Mach Intell 1996;18(9):912–926.
19. de Campos LM, Puerta JM. Stochastic local search algorithms for learning belief networks: Searching in the space of orderings. Lecture Notes in Artificial Intelligence, vol 2143, Berlin: Springer-Verlag; 2001. pp 228–239.