Stochastic Local Algorithms for Learning Belief Networks: Searching in the Space of the Orderings

Luis M. de Campos[1] and J. Miguel Puerta[2]

[1] Dpto de Ciencias de la Computación e I.A.
Universidad de Granada
18071 - Granada, Spain
lci@decsai.ugr.es
[2] Dpto de Informática
Universidad de Castilla-La Mancha
02071 - Albacete, Spain
jpuerta@info-ab.uclm.es

Abstract. An important type of methods for learning belief networks from data are those based on the use of a scoring metric, to evaluate the fitness of any given candidate network to the data base, and a search procedure to explore the set of candidate networks. In this paper we propose a new method that carries out the search not in the space of directed acyclic graphs but in the space of the orderings of the variables that compose the graphs. Moreover, we use a new stochastic search method to be applied to this problem, Variable Neighborhood Search. We also experimentally compare our methods with some other search procedures commonly used in the literature.

Keywords: Belief Networks, Causal Orderings, Learning, Variable Neighborhood Search, Stochastic Hill-Climbing Search.

1 Introduction

Belief Networks (BNs), also known as Bayesian Networks or Causal Networks, are knowledge representation tools able to efficiently manage the dependence and 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 acyclic graph (dag), and b) a set of parameters, which together specify a joint probability distribution over the random variables [20]. In belief networks, the graphical structure represents dependence and independence relationships. The numerical component is a collection of conditional probability measures, which shape the relationships.

Once we have the belief network specified, it constitutes an efficient device to perform inference tasks. However, there still remains the previous problem of building such a network. So, 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.

S. Benferhat and P. Besnard (Eds.): ECSQARU 2001, LNAI 2143, pp. 228–239, 2001.

Nowadays, the problem of learning or estimating a belief network from data is receiving increasing attention within the community of researchers into uncertainty in artificial intelligence. Algorithms for learning (the structure of) BNs have been studied, basically from two points of view: Methods based on conditional independence tests [5,6,7,22,23] and methods based on a scoring metric optimization [12,16,17]. This classification is not exhaustive and/or strict, there also exist algorithms that use a combination of these two methods [1,2,13,21]. In this paper we only consider learning methods based on a scoring metric.

As learning belief networks is, in general, a NP-Hard problem [11], we have to solve it with heuristic methods. Most existing scoring-based learning algorithms apply standard heuristic search techniques, such as greedy hill-climbing, simulated annealing (local search), genetic algorithms, etc. In this paper we focus on local search methods, more precisely stochastic hill-climbing methods. These methods examine only possible local changes at each step, and apply the one that leads to the greatest improvement in the scoring function. When the search process is carried out in the space of dags, the usual choices for local changes are arc addition, arc deletion and arc reversal. Thus, there are $O(n^2)$ possible changes, where n is the number of variables.

However, several authors [18,14,8] have shown that the space of orderings of the variables is much 'smoother' than the space of dags. Moreover, it is also known that, by providing a good ordering of the variables, the learning algorithms become more efficient and accurate. In fact, there is a number of algorithms that need to use such an ordering [1,2,7,10,12]. Therefore, our proposal is to develop learning methods that carry out the search process in the space of the orderings instead of the space of dags.

The search method that we are going to adapt to our problem is, in addition to classical hill-climbing, the recently developed Variable Neighborhood Search (VNS) [15,19], which is a metaheuristic that uses a systematic change of neighborhood within a randomized local search algorithm.

The paper is structured as follows: we begin in Section 2 with the preliminaries. In Section 3 we formalize our proposal of learning belief networks by searching in the space of the orderings: we define our search space, the admisible local changes to move within this space and how to efficiently carry out the evaluation of the different orderings. In Section 4 we introduce the Variable Neighborhood Search. In section 5 we propose two learning algorithms based on orderings: one uses a hill-climbing search and the other uses VNS. In Section 6 we present the experimental evaluation. Finally, Section 7 contains the concluding remarks.

2 Preliminaries

In this section we briefly review BNs and how to learn them. A BN is a directed acyclic graph $G = (\boldsymbol{V}, E)$, where $\boldsymbol{V}$, a set of nodes, represents the system variables and E, a set of arcs, represents the dependence relationships among the variables. A set of parameters is also stored for each variable in $\boldsymbol{V}$, usually con-

ditional probability distributions. For each variable $x_i \in \boldsymbol{V}$ we have a family of conditional distributions $P(x_i|Pa_G(x_i))$, where $Pa_G(x_i)$ represents the parent set of the variable x_i. From these conditional distributions we can recover the joint distribution over $\boldsymbol{V}$:

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|Pa_G(x_i)) \tag{1}$$

This expression represents a decomposition of the joint distribution. The dependence/independence relationships which make possible this decomposition are graphically encoded (through the d-separation criterion [20]) by means of the presence or absence of direct connections between pairs of variables.

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 $\boldsymbol{V}$, find the BN that 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 [1,4,12,16,17].

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

$$f(G : D) = \sum_{i=1}^{n} f(x_i|Pa_G(x_i) : N_{x_i,Pa_G(x_i)}) \tag{2}$$

where $N_{x_i,Pa_G(x_i)}$ are the statistics of the variable x_i and $Pa_G(x_i)$ in D, i.e., the number of instances in D that match each possible instantiation of x_i and $Pa_G(x_i)$. 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 in previous stages. An example is a greedy hill-climbing method that at each step performs the local change that yields the maximal gain, until it reaches a local maximum. As 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 hill-climbing with random restart, which has been used by several authors with relative success (see [16] for more details). This fact has motivated us to try a new search method based on the same principles that the previous one, but with a systematic and reasonable random search in a larger neighborhood at each step if the current local search does not improve the best current maximum. This method, VNS [19], has been applied to solve optimization problems with successful results.

For a dag G, given a causal ordering θ (i.e., an ordering compatible with the topology of the dag[1]), the following independence relationships are true: x_i is conditionally independent of all the variables that precede it in the ordering,

[1] if there is an arc $x_i \to x_j$, then $\theta(x_i) < \theta(x_j)$.

given its parent set $Pa_G(x_i)$, for all x_i. This fact provides a systematic method to build belief networks: for each node x_i, the parents of x_i in the dag are the minimal subset of predecessors of x_i (in the ordering θ) which makes x_i conditionally independent of the rest of its predecessors.

However, different orderings may produce different networks. We would prefer those networks that are able to represent as much true independence relationships as possible (i.e., having as few arcs as possible). For that reason it makes sense to search for the best ordering.

3 Searching in the Space of the Orderings

Let us assume that we want to find a belief network for a problem having n variables, $\boldsymbol{V} = \{x_1, x_2, \ldots, x_n\}$, and we have a database of cases D. Although we are looking for a network, we are going to perform a main search process in the space of the orderings for the variables in $\boldsymbol{V}$, and this search will be guided by a scoring function, that evaluates the network obtained from the given ordering by means of a secondary search process (in the space of the dags compatible with this ordering).

So, our search space is the set of $n!$ orderings, θ, of the variables in $\boldsymbol{V}$ (i.e., the set of permutations of n elements). Now, we have to define the operator to move from one configuration to another neighboring configuration in this space. We propose to use the interchange between two positions i and j in the sequence defining an ordering. More precisely, if θ is the current configuration, then the $n(n-1)/2$ neighboring configurations of θ are those orderings θ_{ij}, where $i < j$, defined as follows:

Let x_u and x_v such that $\theta(x_u) = i$ and $\theta(x_v) = j$. Then

$$\theta_{ij}(x_k) = \begin{cases} \theta(x_k) & \text{if } x_k \neq x_u \text{ and } x_k \neq x_v \\ j & \text{if } x_k = x_u \\ i & \text{if } x_k = x_v \end{cases}$$

Now, we have to decide how to evaluate the quality of an ordering θ. Our proposal is to use a scoring metric, f, defined for dags and to perform a search process in the space of dags compatible with θ. The scoring value of the obtained dag G_θ, $f(G_\theta : D)$, will be the value of θ ($f(\theta : D) = f(G_\theta : D)$). For example, we can use a (deterministic) hill-climbing algorithm with operators of arc addition and arc removal (arc reversal has no sense because the ordering is fixed). In other words, we have to find the best parent set of each variable x_k among the variables that precede x_k in the ordering θ. The search of the parent set of a variable x_k can be done independently of the parent sets of the other variables.

However, if the metric f being used is decomposable, we should try to take advantage of this fact to reduce the complexity of evaluating an ordering θ_{ij}, by using as much information about the evaluation of θ as possible. As θ has been already evaluated, we know that

$$f(\theta : D) = \sum_{k=1}^{n} f(x_k | Pa_{G_\theta}(x_k) : N_{x_k, Pa_{G_\theta}(x_k)})$$

Therefore, for the nodes x_k such that $\theta(x_k) < i$ or $\theta(x_k) > j$, the set of predecessors of x_k will be the same for θ and for θ_{ij}, so that we can be sure that

$$Pa_{G_{\theta_{ij}}}(x_k) = Pa_{G_\theta}(x_k)$$

and we do not need to calculate $f(x_k|Pa_{G_{\theta_{ij}}}(x_k) : N_{x_k,Pa_{G_{\theta_{ij}}}(x_k)})$.

For the nodes x_k such that $i \leq \theta(x_k) \leq j$, their sets of predecessors change, so that we are forced to search again for their parent sets. For each one of these nodes x_k, we start from an empty parent set and, by applying the operators of arc addition and arc removal, perform a hill-climbing search.

With the aim of improving the efficiency of the search process, we are going to restrict, by means of a parameter, r (radius), the number of neighboring configurations. The only admissible neighbors of a given ordering θ are those orderings θ_{ij} such that the 'distance' between the variables to be interchanged is not greater than r, i.e., $|j-i| \leq r$. This will allow to speed up the search process, because each configuration has less neighbors (exactly $r(n-(r+1)/2)$), and the discarded neighbors are precisely the ones whose evaluation is more complex. A radius $r = n-1$ is equivalent to no restriction. If the starting point of the search process is a good ordering, we believe that a drastic change in this ordering (i.e., to interchange two very distant nodes) is not expected to produce an important improvement in the score. In any case, an interchange betwen two distant nodes x_u and x_v can also be obtained by performing successive interchanges involving x_u, x_v and some intermediate nodes (for example, an interchange of lenght $|j-i|$ may be obtained by means of three interchanges of lenght $|j-i|/2$).

4 Variable Neighborhood Search

In this section we review the rules of the basic VNS and apply them for learning belief networks.

Let us denote a finite set of pre-selected neighborhood structures with $\mathcal{N}_k$ ($k = 1, \ldots, k_{max}$), and let $\mathcal{N}_k(x)$ be the set of solutions in the k^{th} neighborhood of x (heuristic local search usually uses one neighborhood structure, i.e., $k_{max} = 1$). The basic VNS heuristic comprises the following steps:

Initialization. Select the set of neighborhood structures $\mathcal{N}_k, k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping criterion.
Repeat the following until the stopping criterion is met:
(a) Set $k = 1$; Until $k = k_{max}$, repeat the following steps:
  (a.1) *Shaking.* Generate a solution x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$).
  (a.2) *Local search.* Apply some local search method with x' as the initial solution; denote with x'' the solution obtained as local optimum.
  (a.3) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1(k = 1)$; otherwise, set $k = k + 1$.

The stopping condition may be based, for example, on maximum running time, maximum number of iterations (of step (a)), or maximum number of iterations (of step (a)) between two improvements. Note that point x' is generated at random in order to avoid cycling, which might occur if any deterministic rule were used.

Once we have an appropriate local search method for an optimization problem, it is easy to program steps (a.1) and (a.3) of the basic VNS. For example, if $\mathcal{N}_k$ is obtained by k-interchanges of solution attributes (as will be our case), only a few lines have to be added to an existing code for a local search method.

The basic VNS is a descent (ascent) first improvement method. Without much additional effort it could be transformed into a descent-ascent method (in step (a.3) set also $x \leftarrow x''$ with some probability even if the solution is worse than the incumbent). Of course, this variant is reminiscent of simulated annealing. Other variants of the basic VNS include:

- Introduce k_{min} and k_{step}, two parameters that control the movement between neighborhood structures, i.e., in the previous algorithm, instead of $k = 1$, set $k = k_{min}$ and instead of $k = k + 1$, set $k = k + k_{step}$. These parameters guide the intensification and diversification of the search.
- Remove the local search. This variant, which is denoted as Reduced VNS, is useful for very large problems for which local search is costly. This works in similar way to the Monte-Carlo method but in a more systematic way. Its relationship with the Monte-Carlo method is the same as that of VNS to multi-start methods.

When using more than one neighborhood structure in the search, as it is done in VNS, the following problem specific questions have to be answered:

- What $\mathcal{N}_k$ should be used and how many of them?
- What should be their order in the search?
- What search strategy should be used in changing neighborhoods?

Furthermore, we must decide what local search routine will be used in the local search step.

5 Learning Algorithms Based on Orderings

We have a database $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$, containing m instances of $\boldsymbol{V}$. We assume a given decomposable scoring metric $f(G : D)$ for dags. Let Θ_n be the set of all the orderings of n elements and $\mathcal{G}_\theta$ be the family of all dags G whose set of vertices is $\boldsymbol{V}$ and whose arcs are compatible with the ordering θ. The problem considered is then:

$$\text{Find} \quad \theta^* = \arg \max_{\theta \in \Theta_n} f(\theta : D) \tag{3}$$

where

$$f(\theta : D) = f(G_\theta : D) = \max_{G \in \mathcal{G}_\theta} f(G : D) \tag{4}$$

Thus, we first find the best dag, G_θ (according to the selected metric f), compatible with an ordering θ, and next select the ordering θ^* that has produced the best dag. The dag G_{θ^*} is the desired solution of our learning problem. We tackle this problem from a heuristic point of view: the two optimization processes are solved using search methods.

The (approximate) solution to the problem in Equation (4) will be obtained by using the search process of dags described in Section 3 (i.e., a hill-climbing search of the best parent set of each node in $\boldsymbol{V}$).

To solve the problem in Equation (3), we propose two alternatives. The first one is to also use a hill-climbing search in the space of the orderings (with the operator of interchange of two positions described in Section 3, and a fixed radius). We call this algorithm HCSO (*H*ill-*C*limbing *S*earch based on *O*rderings).

The second alternative is to use a VNS. This algorithm will be called VNSO (*V*ariable *N*eighborhood *S*earch based on *O*rderings). To do this, we need to define the neighborhood structures $\mathcal{N}_k$. $\mathcal{N}_1$ will be the neighborhood defined by the operator of interchange of any two positions, i.e.,

$$\theta' \in \mathcal{N}_1(\theta) \Longleftrightarrow \theta'(x_u) = \theta(x_v),\ \theta'(x_v) = \theta(x_u) \text{ and } \theta'(x_k) = \theta(x_k)\ \forall x_k \neq x_u, x_v$$

$\mathcal{N}_2$ will be defined by the interchange of two pairs of positions, i.e.,

$$\theta'' \in \mathcal{N}_2(\theta) \Longleftrightarrow \theta'' \in \mathcal{N}_1(\theta') \text{ and } \theta' \in \mathcal{N}_1(\theta)$$

Similarly,

$$\theta'' \in \mathcal{N}_k(\theta) \Longleftrightarrow \theta'' \in \mathcal{N}_1(\theta') \text{ and } \theta' \in \mathcal{N}_{k-1}(\theta)$$

The search strategy between neighborhoods that we are going to use is the one used in the basic VNS ($k = 1$ and at each step $k = k + 1$). As stopping criterion we use the maximum number of iterations between two improvements, together with a maximum number of total iterations.

The local search chosen for the step (a.2) of VNSO is just HCSO. However, instead of using HCSO with a fixed radius r and, in accordance with the search strategy used by VNS, we propose an updating scheme for this parameter: when we move from $\mathcal{N}_k$ to a greater neighborhood $\mathcal{N}_{k+1}$, we also increase the radius (from r to $r + 1$), and when we move to $\mathcal{N}_1$, the radius is set to its initial value.

6 Experimental Evaluation of the Algorithms

In order to test the behavior of the methods proposed in the paper, we have selected the ALARM network [3]. This network has 37 nodes and 46 arcs and is used for diagnosis in a medical domain. It has been considered as a benchmark for evaluating learning algorithms. All the experiments have been carried out on the first 10000 cases of the ALARM database (and comparing the results with the true ALARM network). The scoring function used in all the experiments is the K2 metric [12].

We have run the HCSO with two different radii: $r = 36$ (the maximum radius in this case) and $r = 7$. For VNSO, we have used $k_{max} = 7$ and the initial radius is $r = 7$. We have also used three different options to obtain the initial solution:

- S-PC: Learning a network using the PC algorithm [22] and extracting a topological ordering.
- S-∅: To start from an empty dag and an arbitrary ordering (in our case we used the ordering of the variables in the database).
- S-K2SN: To initialize the search with the result of the algorithm K2SN [9]. This algorithm is an extension of the algorithm K2 that does not require a given ordering: Starting from an empty graph, K2SN iteratively determines the best node to add, until all the nodes have been included in the graph. At each step, the best parent set for each node not previously introduced in the structure is selected (among the nodes already included in the graph, as the K2 algorithm does) and the node producing the best score is added to the graph, linking it to its corresponding parent set. In this way K2SN returns an ordering and a graph compatible with this ordering.

As stopping criterion, we have chosen a maximal number of two iterations without improvement, combined with a maximal number of three total iterations.

In order to compare our algorithms with the classical local search methods, we also use the classical hill-climbing in the space of dags (HCST), with operators of arc addition, arc removal and arc reversal, and the same three initialization methods.

6.1 Experimental Results

The information we have collected from each experiment is the following: the value of the K2 metric (log) of the best individual evaluated; the number of arcs added (A), deleted (D) and inverted (I), compared with the true ALARM network; the number of iterations carried out by the algorithm, i.e., the total number of hill-climbing searches carried out (nS); the total number of individuals evaluated during the search (nE); the total number of statistics (local scores) used (tS); the total number of different statistics calculated (tSC) (using a hashing method, we do not need to recalculate a local score already computed); the mean number of variables involved in the statistics (mV); finally, we also display the value (KL) of the best individual evaluated, which is defined as follows:

$$KL(G:D) = \sum_{\substack{i=1 \\ Pa(x_i) \neq \emptyset}}^{n} Dep(x_i, Pa(x_i)) \tag{5}$$

where:

$$Dep(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{\mathtt{x}_i, \mathtt{y}_j} P(\mathtt{x}_i, \mathtt{y}_j) \frac{P(\mathtt{x}_i, \mathtt{y}_j)}{P(\mathtt{x}_i) P(\mathtt{y}_j)} \tag{6}$$

Note that $KL(G:D)$ is a decreasing monotonic transformation of the Kullback distance between the probability distribution associated to the database and the probability distribution associated to the network G [5] (we use this

transformation because it can be calculated very efficiently, whereas the computation of the Kullback distance has an exponential complexity). The interpretation of $KL(G:D)$ is: the higher this parameter the better is the network.

So, we have collected five measures of the quality of the learned networks (K2, KL, A, D and I) and five measures of the complexity of the search methods (nS, nE, tS, tSC and mV). nE represents the number of dags evaluated in the case of HCST, and the number of orderings evaluated for HCSO and VNSO. tSC is an interesting measure because computing a new local score (not previously stored) requires accessing to the database and it can be a time-consuming process. The complexity of the calculus of these local scores increases exponentially with the value of mV. Although the cost of accessing to the value of a stored local score is much smaller, it is also interesting to know the value tS, because all these local scores have been actually used to compute the (global) scoring values. The measures nE, tS and tSC do not include the cost of the initialization step in the S-PC case (which is quite high).

The results of the experiments are displayed in Tables 1, 2 and 3. For VNSO, the experiments have been carried out ten times. Table 2 displays the average value μ and the standard desviation σ of each item.

Table 1. Results for HCSO.

	Radius = 36				Radius = 7		
	Empty	K2SN	PC		Empty	K2SN	PC
K2	-47080.22	-47076.20	-47109.89	K2	-47513.60	-47079.60	-47117.67
KL	9.2740	9.2740	9.2655	KL	9.2687	9.2762	9.2639
A	1	1	1	A	23	4	2
D	1	1	2	D	3	1	3
I	0	0	0	I	11	0	0
nS	1	1	1	nS	1	1	1
nE	17316	3330	3996	nE	7280	1300	3120
tSC	51609	20160	26263	tSC	18333	11384	17266
tS	1.39E7	2.52E6	2.96E6	tS	2.88E6	4.27E5	9.73E5
mV	4.83	4.35	4.39	mV	4.67	4.15	4.36

The best result found by the search algorithms is a network with a value of the K2 metric equal to -47076.20 (VNSO-S-$\emptyset$, VNSO-S-K2SN and HCSO-S-K2SN are all able to obtain this network). Note that the K2 and KL values of the true ALARM network for the database being used are equal to -47086.57 and 9.2744, respectively.

First, we have to note that the initialization method used is quite relevant from the point of view of the quality of the obtained result (K2, A, D, and I values) and the efficiency of the search process (nS, nE, tSC and tS values) for all the methods (Table 4 displays the K2 and KL values for the different initial networks). The best initialization is always produced by the K2SN algorithm (particularly, a simple HCSO initialized with K2SN produces the best result). This is not surprising for VNSO and HCSO because K2SN is explicitly designed to work with orderings, but is somewhat surprising for HCST. Another interest-

ing result is that the more informed initialization S-PC is not better than the 'vacuous' initialization S-$\emptyset$ when searching in the space of the orderings (the exception is HCSO with a small radius). It seems to us that PC directs the process towards a suboptimal local maximum, which is difficult to offset by the search process.

Table 2. Results for VNSO.

	Empty		K2SN		PC	
	μ	σ	μ	σ	μ	σ
K2	-47087.39	12.26	-47076.49	0.90	-47110.08	0.41
KL	9.2731	0,004	9.2740	0.000	9.2655	0.000
A	4.5	3.03	1.8	0.42	1.9	0.32
D	1.6	0.52	1.0	0.0	2.0	0.0
I	0.8	1.03	0.0	0.0	0.0	0.0
nS	181.0	99.3	96.1	29.02	68.6	17.3
nE	327335	204091	179865	55924	99061	30844
tSC	68087	11680	49902	5664	54670	7832
tS	5.98E7	2.73E7	4.16E7	1.12E7	2.43E7	6.53E6
mV	4,96	0,06	4.79	0,08	4.85	0.1

The results obtained support the conclusion that searching in the space of the orderings is a good idea: Both VNSO and HCSO outperform HCST in all the cases (except HCSO-S-$\emptyset$[2]).

Table 3. Results for HCST.

	Empty	K2SN	PC
K2	-47267.11	-47081.66	-47133.41
KL	9.2657	9.2761	9.2708
A	11	3	3
D	4	1	1
I	6	0	2
nS	1	1	1
nE	72335	15820	17901
tSC	3280	5384	1804
tS	1.47E5	5.56E4	3.65E4
mV	2.98	3.65	3.21

Focusing on the methods that search in the space of the orderings, restricting the search process by using a small radius produces results slighty worse than the unrestricted search, from the point of view of the solution quality, but with an important gain in efficiency. We conjecture that a radius of about a half of the maximum radius would be an optimal choice. On the other hand, VNSO is better than HCSO if we use the same radius, as we could expect. However, HCSO with maximum radius ($r = 36$) behaves even a bit better than VNSO

[2] remember that this initialization uses the ordering of the variables in the database, which is a particularly bad ordering.

with small radius ($r = 7$). We also conjecture that VNSO equipped with a radius of about one third of the maximum radius would produce excellent results.

Table 4. K2 and KL values for the three initial networks.

	Empty	K2SN	PC
K2	-86822	-47515	-51083
KL	0.0	9.2205	8.3170

With respect to the complexity of the search methods, although HCSO evaluates less individuals than HCST[3], the cost of evaluating each individual for HCSO is greater than for HCST. Overall, although HCSO gives results better than HCST, the latter is more efficient than the former. Obviously, the complexity of VNSO increases considerably. Nevertheless, we have observed that VNSO always finds the best individual in the first iteration (of step (a)), and due to the stopping criterion selected, it needs to perform another complete iteration to halt. So, the complexity of VNSO could be considerably reduced without compromising the quality of the result by performing only one iteration.

7 Concluding Remarks

In this work we have proposed a new strategy for learning belief networks based on searching for good orderings and searching for good networks compatible with a given ordering. Moreover, a new search method has been adapted to this problem. The proposed methods have improved the results obtained by other classical search methods that explore the space of dags. Nevertheless, a more systematic experimentation has to be done in order to confirm this conclusion. We also plan to study in the future other variants of VNS, as well as other operators for defining local changes in the space of the orderings.

Acknowledgements. This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT), under project TIC 2000-1351.

References

1. Acid, S., Campos, L.M. de: BENEDICT: An algorithm for learning probabilistic belief networks. In: Proceedings of the Sixth International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (1996) 979–984
2. Acid, S., Campos, L.M. de: A hybrid methodology for learning belief networks: BENEDICT. Int. J. Approx. Reason., to appear
3. Beinlich,I.A., Suermondt, H.J., Chavez, R., Cooper, G.: The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medicine (1989) 247–256

[3] This suggests that the space of the orderings is more 'smoother' than the space of dags.

4. Buntine, W.: A guide to the literature on learning probabilistic networks from data. IEEE T. Knowl. Data En. **8** (1996) 195–210
5. Campos, L.M. de: Independency relationships and learning algorithms for singly connected networks. J. Exp. Theor. Artif. In. **10** (1998) 511–549
6. Campos, L.M. de, Huete, J.F.: On the use of independence relationships for learning simplified belief networks. Int. J. Intell. Syst. **12** (1997) 495–522
7. Campos, L.M. de, Huete, J.F.: A new approach for learning belief networks using independence criteria. Int. J. Approx. Reason. **24** (2000) 11–37
8. Campos, L.M. de, Huete, J.F.: Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing. In: Proceedings of the Eighth International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems, Vol. I (2000) 333–340
9. Campos, L.M. de, Puerta, J.M.: Stochastic local and distributed search algorithms for learning belief networks. In: Proceedings of the Third International Symposium on Adaptive Systems (ISAS): Evolutionary Computation and Probabilistic Graphical Models (2001) 109–115
10. Cheng, J., Bell, D.A., Liu, W.: An algorithm for Bayesian belief network construction from data. In: Proceedings of AI and STAT'97 (1997) 83–90
11. Chickering, D.M.: Learning Bayesian networks is NP-complete. In: Fisher, D., Lenz, H.J. (eds): Learning from Data. Lectures Notes in Statistics, Vol. 112. Springer-Verlag (1996) 121–130
12. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Mach. Learn. **9** (1992) 309–347
13. Dash, D., Druzdzel, M.: A hybrid anytime algorithm for the construction of causal models from sparse data. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (1999) 142–149
14. Friedman, N., Koller, D.: Being Bayesian about network structure. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (2000) 201–210
15. Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. Eur. J. Oper. Res. **130** (2001) 449–467
16. Heckemann, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. Mach. Learn. **20** (1995) 197–244
17. Lam, W., Bacchus, F.: Learning Bayesian belief networks. An approach based on the MDL principle. Comput. Intell. **10** (1994) 269–293
18. Larrañaga, P., Kuijpers, C.M., Murga, R.H., Yurramendi. Y.: Learning Bayesian network structure by searching for the best ordering with genetic algorithms. IEEE T. Syst. Man Cy. A **26** (1996) 487–493
19. Mladenović, N., Hansen, P.: Variable neighborhood search. Computer Oper. Res. **24** (1997) 1097–1100
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan and Kaufman (1988)
21. Singh, M., Valtorta, M.: Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. Int. J. Approx. Reason. **12** (1995) 111–131
22. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction and Search. Lectures Notes in Statistics, Vol. 81. Springer-Verlag (1993)
23. Verma, T., Pearl, J.: Equivalence and synthesis of causal models. In: Uncertainty in Artificial Intelligence, Vol. 6. North-Holland (1991) 255–268