Local Search Methods for Learning Bayesian Networks Using a Modified Neighborhood in the Space of DAGs

Luis Miguel de Campos[1], Juan Manuel Fernández-Luna[2], and Jose Miguel Puerta[3]

[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 Jaén, 23071 – Jaén, Spain
jmfluna@ujaen.es
[3] Dpto. de Informática
Universidad de Castilla-La Mancha, 02071 - Albacete, Spain
jpuerta@info-ab.uclm.es

Abstract. The dominant approach for learning Bayesian networks from data is based on the use of a scoring metric, that evaluates the fitness of any given candidate network to the data, and a search procedure, that explores the space of possible solutions. The most efficient methods used in this context are (Iterated) Local Search algorithms. These methods use a predefined neighborhood structure that defines the feasible elementary modifications (local changes) that can be applied to a given solution in order to get another, potentially better solution. If the search space is the set of directed acyclic graphs (dags), the usual choices for local changes are arc addition, arc deletion and arc reversal. In this paper we propose a new definition of neighborhood in the dag space, which uses a modified operator for arc reversal. The motivation for this new operator is the observation that local search algorithms experience problems when some arcs are wrongly oriented. We exemplify the general usefulness of our proposal by means of a set of experiments with different metrics and different local search methods, including Hill-Climbing and Greedy Randomized Adaptive Search Procedure (GRASP), as well as using several domain problems.

1 Introduction

Bayesian Networks (BNs) are graphical models able to represent and manipulate efficiently n-dimensional probability distributions [18]. A Bayesian network uses two components to codify qualitative and quantitative knowledge: (a) A directed acyclic graph (dag), $G = (\boldsymbol{V}, E)$, where the nodes in $\boldsymbol{V} = \{X_1, X_2, \ldots, X_n\}$ represent the random variables from the problem we want to solve, and the topology of the graph (the arcs in E) encodes conditional (in)dependence relationships

F.J. Garijo, J.C. Riquelme, and M. Toro (Eds.): IBERAMIA 2002, LNAI 2527, pp. 182–192, 2002.

among the variables (by means of the presence or absence of direct connections between pairs of variables); (b) a set of conditional probability distributions drawn from the graph structure: For each variable $X_i \in \boldsymbol{V}$ we have a family of conditional probability distributions $P(X_i|pa_G(X_i))$, where $pa_G(X_i)$ represents any combination of the values of the variables in $Pa_G(X_i)$, and $Pa_G(X_i)$ is the parent set of X_i in G. 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 decomposition of the joint distribution gives rise to important savings in storage requirements. It also allows, in many cases, to efficiently perform probabilistic inference (propagation), i.e., to compute the posterior probability for any variable given some evidence about the values of other variables in the graph [14,18]: The independences represented in the graph reduce changes in the state of knowledge to local computations.

Although in the last years the problem of learning or estimating Bayesian networks from data has received considerable attention, within the community of researchers into uncertainty in artificial intelligence, it is still an active research area. The fact that finding optimal BNs from data is, in general, a NP-Hard problem [7], has motivated the use of heuristic search methods to solve it. The common approach 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 [1,6,8,13,17]. The alternative approach, constraint-based, is to search for the network satisfying as much independences present in the data as possible [10,21,19]. Obviously, the decision about which conditional independences are either true or false is made by means of statistical tests. There also exist hybrid algorithms that use a combination of these two methods [1,9,22].

In this paper we focus on Local Search methods, the most efficient methods, and that rely on a neighborhood structure that defines the local rules used to move within the search space. The standard neighborhood in the space of dags uses the operators of arc addition, arc deletion and arc reversal. The main contribution of this paper is the proposal of an alternative definition of neighborhood, which may alleviate some problems of premature convergence to a local optimum due to the difficulty of (locally) improving dags where some arcs are wrongly oriented. We also propose a new algorithm for learning Bayesian network structures, which uses the GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic [12].

The paper is structured as follows: We begin in Section 2 with some preliminaries. In Section 3 we define the proposed neighborhood structure for searching in the space of dags. Section 4 describes GRASP-BN, a new iterated local search-based learning algorithm that uses Hill-Climbing and a probabilistic version of the algorithm B [5]. In Section 5, we analyze the experimental results obtained by both Hill-Climbing and GRASP-BN (on two different domains, ALARM [3]

and INSURANCE [2], and using two different scoring metrics, K2 [8] and BDeu [13]), when these algorithms use the standard neighborhood and the proposed alternative. Finally, Section 6 contains the concluding remarks and some proposals for future research.

2 Local Search Methods for Learning BNs

The problem of learning the structure of a Bayesian network can be stated as follows: Given a training set $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ of instances of $\boldsymbol{V}$, find the dag G^* such that

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

where $f(G : D)$ is a scoring metric measuring the fitness of any candidate dag G to the dataset D and $\mathcal{G}_n$ is the family of all the dags with n nodes.

Local Search (or Hill-Climbing) methods traverse the search space, starting from an initial solution, by examining only possible local changes at each step, and applying the one that leads to the greatest improvement in the scoring metric. The search process terminates when it is blocked at a local optimum (no local change improves the current solution), although it may be restarted on the basis of either a random modification of the current optimum (by applying a number of local transformations), or a new (random) initial solution. The set of feasible local changes that can be applied to any given solution is determined by the choice of the neighborhood structure. The effectiveness and efficiency of a local search procedure depends on several aspects, such as the neighborhood structure, the fast evaluation of the scoring metric of the neighbors, and the starting solution itself.

As we have already commented, the usual choices for local changes in the space of dags are arc addition, arc deletion and arc reversal, avoiding (in the first and the third case) the inclusion of directed cycles in the graph. Thus, there are $O(n^2)$ possible changes, where n is the number of variables.

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

$$f(G : D) = \sum_{i=1}^{n} f_D(X_i, Pa_G(X_i)) \tag{3}$$

$$f_D(X_i, Pa_G(X_i)) = f_D(X_i, Pa_G(X_i) : N_{x_i, pa_G(X_i)}) \tag{4}$$

where $N_{x_i, pa_G(X_i)}$ are the statistics of the variables 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 $Pa(X_i)$.

The efficiency of the algorithms that search in the space of dags using local methods is mainly due to the property of decomposition that many metrics exhibit: a procedure that changes one arc at each move can efficiently evaluate the

improvement obtained by this change. Such a procedure can reuse the computations carried out at previous stages, and only the statistics corresponding to the variables whose parent set has been modified need to be recomputed.

3 A Modified Neighborhood Structure in the Space of Dags

By monitoring a typical local search algorithm for learning BNs as it progresses, we have observed some situations where it gets into troubles. Let us explain these situations by means of the following example:

Example: Consider the network with four variables displayed in Figure 1(a). From this network we have generated a database containing 1000 instances of these variables using logic sampling. The value of the K2 metric for this database and this network is also shown in the figure. A Hill-Climbing algorithm could obtain the network in Figure 1(b). 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 Figures 1(c)-(e)), so that it is a local maximum. ♢

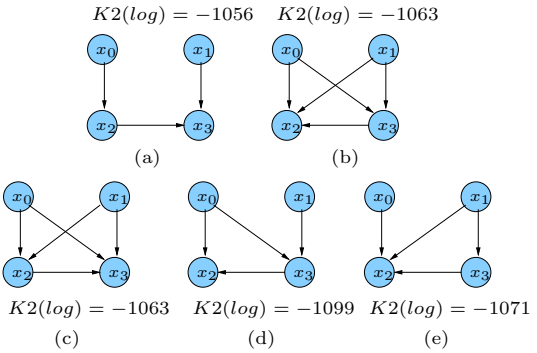

Fig. 1. Problems with the classical neighborhood when an arc is wrongly directed.

This behavior is common for many local search-based learning algorithms: When they mistake the direction of some arc connecting two nodes[1], 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 scape from this local maximum. In the previous example, the wrong orientation of the arc linking the variables X_2 and X_3 is the cause of the problem.

A possible solution for this problem could be to carry out the search process in a different space. For example, we could use the space of equivalence classes

[1] This situation may be quite frequent at early stages of the search process.

of dags, to postpone the decisions about the direction of the links until we have more information. Recently, the possibility of searching in the dag space, but including some characteristics relative to equivalence classes of dags, has been considered [15]. This method uses an operator, called RCAR (Repeated Covered Arc Reversal), which iteratively inverts a prefixed (random) number of *covered* arcs[2]. Then, a Hill-Climbing algorithm is fired to obtain a new local maximum, and the whole process is iterated a fixed number of times. We are going to follow a different approach, by modifying the neighborhood structure, i.e., changing the classical operators used to move in the space of dags.

The classical neighborhood of a dag G is $\mathcal{N}(G) = \mathcal{N}_A(G) \cup \mathcal{N}_D(G) \cup \mathcal{N}_R(G)$, being $\mathcal{N}_A(G)$, $\mathcal{N}_D(G)$ and $\mathcal{N}_R(G)$ the links subsets for added, deleted and inverted respectively. Our proposal implies to modify the reversal operator and, therefore, the redefinition of the neighborhood $\mathcal{N}_R(G)$

The new definition of $\mathcal{N}_R(G)$ states that, for any given arc $X_j \rightarrow X_i$ in G, 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 which have been connected in the wrong direction, without being limited to move only one arc every time, for example the dags (a) in the figure 1 could be a dag neighbor of the graph (b).

Note that, for any arc $X_j \rightarrow X_i \in G$, the number of possible 'reversals' is now $O(2^{2p})$, with $p = |Pa_G(X_i) \cup Pa_G(X_j)| - 1$, 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, so we expect that the computational cost will not be excessive[3]. In any case we could limit the cardinality of the new parent sets (which is a common practice for other algorithms [8,16]). Note also that, although the new reversal operator may change more than one arc, the number of parent sets that are modified is still two, hence we only have to recompute two statistics to evaluate the corresponding neighbor.

4 A Grasp-Based Learning Algorithm

GRASP [20] is a multi-start or iterative metaheuristic in which each iteration consists basically of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated until a local optimum is found during the local search phase. The best overall solution is kept as the result. In this section we develop an algorithm for learning Bayesian network structures using GRASP, which will be called GRASP-BN.

A general GRASP algorithm works as follows: At each iteration of the construction phase, a solution is built using a greedy randomized process: it incorporates elements to the partial solution under construction from a restricted

[2] An arc $X_j \rightarrow X_i$ in a dag G is covered if $Pa_G(X_i) = Pa_G(X_j) \cup \{X_j\}$. A covered arc may be inverted and the resultant dag is equivalent to G.

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

candidate list (RCL). The elements in this list are selected from all the feasible candidate elements according to a greedy evaluation function. The element to be incorporated into the partial solution is randomly selected from those in the RCL. Once the selected element is incorporated to the partial solution, the candidate list is updated and their elements are reevaluated. When a solution is finally obtained, the second phase fires a local search algorithm starting from this solution. These two phases are repeated a given number of iterations.

Our GRASP-BN algorithm will use, in its second phase, a Hill-Climbing (HC) search in the space of dags, for some neighborhood structure and some metric f. For the first phase, we are going to create a randomized version of the algorithm B [5].

Algorithm B is a greedy search heuristics. It starts with an empty dag G and at each step it adds the arc with the maximum increase in the (decomposable) scoring metric f, avoiding the inclusion of directed cycles in the graph. The algorithm stops when adding any valid arc does not increase the value of the metric. The gain obtained by inserting a feasible arc $X_j \to X_i$ in G can be evaluated efficiently by means of the difference $f_D(X_i, Pa_G(X_i) \cup \{X_j\}) - f_D(X_i, Pa_G(X_i))$. At each step, after inserting in G the best valid arc, $X_j \to X_i$, the algorithm identifies and discards the new unfeasible arcs by searching for the ancestors and descendants of X_i. After that, as the value $f_D(X_i, Pa_G(X_i))$ has been modified, the algorithm recomputes the new values of $f_D(X_i, Pa_G(X_i) \cup \{X_k\}) - f_D(X_i, Pa_G(X_i))$ for any valid arc $X_k \to X_i$.

The probabilistic version of this algorithm that we propose is the following: instead of always selecting the best arc, we will use a stochastic decision rule that selects the best arc with probability p_0, and with probability $1 - p_0$ each arc $X_j \to X_i$ in the restricted candidate list RCL (that will contain all the feasible candidate arcs which produce an improvement) will be selected with a probability $p(X_j \to X_i)$ proportional to its merit:

$$\begin{cases} \arg\max_{X_j \to X_i \in RCL} \{f_D(X_i, Pa_G(X_i) \cup \{X_j\}) - f_D(X_i, Pa_G(X_i))\} & u \leq p_0 \\ X_l \to X_r, & u > p_0 \end{cases} \tag{5}$$

where u is a random number uniformly distributed in $[0, 1]$, p_0 is the parameter that determines the relative importance of *exploitation* versus *exploration*, and $X_l \to X_r$ is an arc in RCL randomly selected according to the following probabilities:

$$p(X_l \to X_r) = \frac{f_D(X_r, Pa_G(X_r) \cup \{X_l\}) - f_D(X_r, Pa_G(X_r))}{\sum_{X_j \to X_i} f_D(X_i, Pa_G(X_i) \cup \{X_j\}) - f_D(X_i, Pa_G(X_i))} \tag{6}$$

We have to remark that this probabilistic version of the algorithm B could also be used at the initialization stages of other stochastic search algorithms (as Genetic Algorithms, Estimation of Distribution Algorithms and Ant Colonies).

5 Experimental Results

In this section we will evaluate experimentally the usefulness of the proposed neighborhood, as well as the GRASP-BN algorithm. We have selected two test domains: ALARM [3] and INSURANCE [2]. For ALARM we have used the first 3,000 cases of the classical 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 our experiments we have used two decomposable metrics: K2 [8] and BDeu [13,5] (both in logarithmic version). For BDeu, we use an equivalent sample size equal to 1. We have implemented two versions of a Hill-Climbing algorithm, using the classical definition of neighborhood (HCc) and the proposed modification (HCm). The initial solution of the search process is the empty network in all the cases. Experiments with other two forms of initialization (the networks obtained by the algorithms K2SN [11] and PC [21]) were also carried out, obtaining results similar to the ones displayed for the empty network. The GRASP-BN algorithm has also been implemented using the two neighborhood structures (GRASP-BNc and GRASP-BNm). The parameter determining the number of iterations has been fixed to 15, and the one to trade-off between exploration and exploitation is $p_0 = 0.8$.

The following performance measures have been computed: (1) Measures of quality (effectiveness) of the learned network: (K2) and (BDeu) the value of the corresponding metric for the best network found by the algorithm; the number of arcs added (A), deleted (D) and inverted (I) when we compare the learned network with the true network. (2) Measures of efficiency of the algorithm: (EstEv) the number of different statistics evaluated during the execution of the algorithm; (TEst) the total number of statistics used by the algorithm. 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; (NVars) the average number of variables that intervene in the computed statistics.

The results obtained by HCc and HCm using the K2 and BDeu metrics are shown in Table 1.

The results of the experiments with the algorithms GRASP-BNc and GRASP-BNm is shown in Table 2 for the ALARM domain, and in Table 3 for the INSURANCE domain. In all the cases, the displayed values represent the averages and the standard deviations of 10 executions of each algorithm. We also give information about the best individuals found in all the executions.

We can see that, in all the cases, the results obtained by an algorithm when it uses the new definition of neighborhood are considerably better than the ones offered by the same algorithm using the classical neighborhood[4], in terms of

[4] Except in one case, where we obtain the same results.

Table 1. Results of HCc and HCm using the K2 and BDeu metrics.

	ALARM		INSURANCE	
	HCc	HCm	HCc	HCm
K2	-14425.62	-14414.55	-57998.10	-57934.61
A	6	4	10.33	7.67
D	4	2	11.67	11.00
I	3	2	7.67	7.67
EstEv	3375	3336	2050	2169
TEst	1.54E+05	1.56E+05	7.66E+04	8.32E+04
NVars	2.99	2.93	3.09	3.08
	HCc	HCm	HCc	HCm
BDeu	-33109.47	-33109.47	-133393.03	-133326.27
A	3	3	7.67	7.00
D	2	2	10.00	10.00
I	2	2	10.00	10.67
EstEv	3300	3284	1995	2100
TEst	1.47E+05	1.49E+05	7.15E+04	8.80E+04
NVars	2.88	2.87	2.99	3.04

Table 2. Results of GRASP-BNc and GRASP-BNm for ALARM, using the K2 and BDeu metrics.

	GRASP-BNc			GRASP-BNm		
	μ	σ	Best	μ	σ	Best
K2	-14429.92	18.51	-14414.55	-14404.23	2.52	-14401.91
A	7.20	2.44	4	2.80	1.55	1
D	2.70	0.95	2	1.10	0.32	1
I	4.30	2.54	2	1.20	1.03	0
EstEv	13306	1228		13148	1061	
TEst	5.83E+05	8.85E+03		9.09E+05	9.12E+04	
NVars	4.01	0.05		3.93	0.08	
	μ	σ	Best	μ	σ	Best
BDeu	-33190.14	10.17	-33165.90	-33105.05	4.24	-33101.14
A	5.50	1.18	6	1.40	0.97	1
D	2.00	0.00	2	1.30	0.48	1
I	4.50	0.85	3	1.00	1.41	0
EstEv	12317	759		12769	735	
TEst	5.56E+05	5.23E+04		8.80E+05	6.00E+04	
NVars	3.87	0.07		3.85	0.03	

effectiveness (with respect to both the value of the metric[5] and the number of erroneous arcs). For comparative purposes, the respective K2 and BDeu values of the true networks for the corresponding databases are -14412.69 and -33113.83 for ALARM, and -58120.95 and -133160.47 for INSURANCE. The corresponding values for the empty networks, which may serve as a kind of scale, are -26008.08 and -59889.80 for ALARM, and -93593.76 and -215514.96 for INSURANCE.

With respect to the efficiency of the algorithms, we are going to focus in the number of different statistics evaluated (EstEv) and the number of variables involved (Nvars): Most of the running time of a scoring-based learning algorithm

[5] Note that we are using log versions of the metrics, so that the differences are much greater in a non-logarithmic scale.

Table 3. Results of GRASP-BNc and GRASP-BNm for INSURANCE, using the K2 and BDeu metrics.

	GRASP-BNc			GRASP-BNm		
	μ	σ	Best	μ	σ	Best
K2	-57950.53	53.29	-57857.90	-57835.23	30.30	-57779.41
A	10.63	2.11	7	4.57	2.36	4
D	12.27	1.01	10	9.40	1.19	9
I	8.90	3.53	2	3.13	2.34	2
EstEv	7702	626		9135	840	
TEst	2.41E+05	2.74E+04		6.45E+05	9.34E+04	
NVars	3.98	0.06		3.97	0.06	

	μ	σ	Best	μ	σ	Best
BDeu	-133143.92	165.52	-132814.06	-132763.92	100.64	-132592,25
A	6.60	1.92	3	2.23	1.28	0
D	9.97	1.79	7	8.07	0.91	8
I	7.43	2.61	7	2.87	1.96	1
EstEv	7768	923		9011	1095	
TEst	2.54E+05	3.85+04		6.62E+05	1.61E+05	
NVars	3.87	0.09		3.88	0.08	

is spent in the evaluation of statistics from the database, and this time increases exponentially with the number of variables. So, an approximate measure of the time complexity of an algorithm is $\text{EstEv} * 2^{\text{NVars}}$. We can observe that the values of NVars are almost identical and the values of EstEv are not sensibly different for the two versions of each algorithm: The values of EstEv in the ALARM domain are even lower when we use the new neighborhood, whereas the opposite is true for INSURANCE. The total number of statistics used is systematically larger with the new neighborhood, even considerably larger in some cases, but as we have already commented, using hashing techniques we can access very quickly to previously computed statistics, so that the time required to compute *new* statistics dominates completely. Therefore, the running times of the algorithms using the two neighborhoods are comparable.

These results increase our confidence in the usefulness and applicability of the proposed neighborhood for any learning algorithm that searches locally in the space of dags, no matter which metric or which type of local search is used.

The comparison between HC and GRASP is favorable to the later (except in the case of using the classical neighborhood with ALARM), at the expense of increasing the running times. Particularly, GRASP-BN equipped with the new neighborhood structure offers excellent results. Moreover, the low standard deviations obtained in this case are noteworthy, indicating that this algorithm seems to be quite stable.

6 Concluding Remarks

In this paper we have proposed a new definition of neighborhood for the space of directed acyclic graphs, which copes with the problems that local search-based learning algorithms encounter when some arcs are wrongly oriented. Our experiments, carried out with different metrics, databases and local search techniques,

support the conclusion that the new neighborhood structure improves the performance of the algorithms systematically, without significantly increasing their complexity.

We have also developed another learning algorithm, based on the GRASP metaheuristic, that uses a probabilistic version of the algorithm B in the construction phase, to initialize the local search phase. This algorithm, in conjunction with the new neighborhood structure, obtained excellent results in our experiments.

For future research, in addition to carry out a more systematic experimentation and comparative analysis, we also plan to study several variants of the new operator for arc reversal (e.g., using the intersection of the parent sets in place of their union or applying the operator only to covered arcs).

Acknowledgements. 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. S. Acid and L.M. de Campos. A hybrid methodology for learning belief networks: Benedict. *International Journal of Approximate Reasoning*, 27(3):235–262, 2001.
2. J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2):213-244, 1997.
3. I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, 247–256, 1989.
4. R.R. Bouckaert. Bayesian Belief Networks: From Construction to Inference. PhD. Thesis, University of Utrecht, 1995.
5. W. Buntine. Theory refinement of Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, 52–60, 1991.
6. W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210, 1996.
7. D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-Complete. In D. Fisher and H. Lenz, Eds., *Learning from Data: Artificial Intelligence and Statistics V*, Springer-Verlag, 121–130, 1996.
8. G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–348, 1992.
9. D. Dash and M. Druzdel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 142–149, 1999.
10. L.M. de Campos and J.F. Huete. A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, 24:11–37, 2000.
11. L.M. de Campos and J.M. Puerta. Stochastic local search algorithms for learning belief networks: Searching in the space of orderings. *Lecture Notes in Artificial Intelligence*, 2143:228–239, 2001.

12. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
13. D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–244, 1995.
14. F.V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, 1996.
15. T. Kocka and R. Castelo. Improved learning of Bayesian networks. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 269–276, 2001.
16. P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.
17. W. Lam and F. Bacchus. Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293, 1994.
18. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.
19. J. Pearl and T.S. Verma. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 220–227, 1990.
20. M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, Eds., *State of the Art Handbook in Metaheuristics*, Kluwer. To appear.
21. P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Lecture Notes in Statistics 81, Springer Verlag, 1993.
22. M. Singh and M. Valtorta. Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12:111–131, 1995.