2009 Ninth International Conference on Intelligent Systems Design and Applications

Structural Learning of Bayesian Networks by using Variable Neighbourhood Search based on the Space of Orderings

Juan I. Alonso-Barba, Luis delaOssa and Jose M. Puerta
Laboratory of Intelligent Systems and Data Mining
Albacete Research Institute of Informatics
Computing Systems Department
University of Castilla-La Mancha, 02071 Albacete, Spain
{jialonso,ldelaossa,jpuerta}@dsi.uclm.es

Abstract

Structural Learning of Bayesian networks (BNs) is an NP-hard problem generally addressed by means of heuristic search algorithms. Although these techniques do not guarantee an optimal result, they allow obtaining good solutions with a relatively low computational effort.

Many proposals are based on searching the space of Directed Acyclic Graphs. However, there are alternatives consisting of exploring the space of equivalence classes of BNs, which yields more complex and difficult to implement algorithms, or the space of the orderings among variables. In practice, ordering-based methods allow reaching good results, but, they are costly in terms of computation.

In this paper, we prove the correctness of the method used to evaluate each permutation when exploring the space of orderings, and we propose two simple and efficient learning algorithms based on this approach. The first one is a Hill climbing method which uses an improved neighbourhood definition, whereas the second algorithm is its natural extension based on the well-known Variable Neighbourhood Search metaheuristic. The algorithms have been tested over a set of different domains in order to study their behaviour in practice.

1. Introduction

Learning Bayesian networks from data is a complex task which receives a lot of attention from researchers in the area. This problem has been stated as an *NP-hard* one [6], therefore, it becomes necessary using heuristic and metaheuristic techniques to find quality solutions.

In most related works, the search of the network structure is carried out in the space of Directed Acyclic Graphs (DAGs). In this sense, there are several proposals based on greedy [3], local search [11], or population based algorithms [13, 2]. A different approach, which generally improves the aforementioned methods, consists of searching the space of equivalence classes. One of the algorithms based on that idea, the *Greedy Equivalence Search (GES)* [4], is nowadays the reference in Bayesian Network learning. This method is asymptotically correct because, under certain conditions, the final solution is guaranteed to be a perfect-map of the target distribution. However its use is limited due to the fact that it is hard to implement and understand.

There is an alternative to both approaches. Given a topological order among variables, it is possible to obtain a minimal I-map compatible with it [3, 8]. Based on that, some methods assign to each order the score of the best Bayesian network consistent with it, and then use search algorithms which traverse the permutation space. Searching the space of orderings presents two main advantages: (1) the dimension of the search space is reduced; (2) the need to perform acyclicity checks on candidate successors is avoided. Moreover, each step in the search makes a more global modification of the current state, thereby better avoiding local maxima. In practice, ordering-based methods reach good results, but are very costly in terms of computation.

With this work, we propose an efficient method to evaluate each permutation and we prove its correctness. Moreover, we propose an efficient method which explores the space of orderings by means of a *Variable Neighbourhood Search* (VNS) [14] algorithm. This method is based on the use of a particular neighbourhood operator which allows reducing the calculations significantly. Experiments show that the results obtained with this technique do not present statistical difference with those obtained with GES, and improve the Hill Climbing in the space of DAGs.

This paper is structured into four sections besides this introduction. Section 2 introduces some aspects of Bayesian

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.157

network learning through search in the space of orderings. Then, in section 3 both the proposed algorithm and the improvements used to increase its efficiency are explained with detail, and an experimental evaluation is carried out in section 4. Finally, conclusions and future works are briefly described in section 5.

2. Learning Bayesian Networks: Space of Orderings

Bayesian Networks are graphical models that can efficiently represent n-dimensional probability distributions [15]. This representation has two components that respectively codify qualitative and quantitative knowledge:

• A graphical structure, or more precisely, a *directed acyclic graph* (DAG), $\mathcal{G} = (\boldsymbol{V}, \mathbf{E})$, where the nodes in $\boldsymbol{V} = \{X_1, X_2, \ldots, X_n\}$ represent the random variables[1] from the problem being modeling, and the topology of the graph (the arcs in $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$) encodes conditional (in)dependence relationships among the variables (by means of the presence or absence of direct connections between pairs of variables).

• A set of numerical parameters ($\boldsymbol{\Theta}$), usually conditional probability distributions drawn from the graph structure: For each variable $X_i \in \boldsymbol{V}$ we have a conditional probability distribution $P(X_i|pa_{\mathcal{G}}(X_i))$, where $pa_{\mathcal{G}}(X_i)$ represents any combination of the values of the variables in $Pa_{\mathcal{G}}(X_i)$, and $Pa_{\mathcal{G}}(X_i)$ is the parent set of X_i in $\mathcal{G}$. From these conditional distributions we can recover the joint probability distribution over $\boldsymbol{V}$ thanks to the Markov Condition [12]:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|Pa_{\mathcal{G}}(X_i))$$

We denote that variables in $\mathbf{X}$ are conditionally independent (through d-separation) of variables in $\mathbf{Y}$ given the set $\mathbf{Z}$, in a DAG $\mathcal{G}$ by $\langle \mathbf{X}, \mathbf{Y}|\mathbf{Z}\rangle_{\mathcal{G}}$. The same sentence but in a probability distribution p is denoted by $I_p(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. A DAG $\mathcal{G}$ is an *I-map* of a probability distribution p if $\langle \mathbf{X}, \mathbf{Y}|\mathbf{Z}\rangle_{\mathcal{G}} \Rightarrow I_p(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$, and it is minimal if no arc can be removed.

Furthermore, a distribution p is faithful if there exists a graph, $\mathcal{G}$, to which it is faithful. In a faithful BN $\langle \mathbf{X}, \mathbf{Y}|\mathbf{Z}\rangle_{\mathcal{G}} \Leftrightarrow I_p(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. It is always possible to build a minimal I-map of any given probability distribution p, but some distributions do not admit an isomorphic (faithful) model [15].

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

$$\mathcal{G}^* = \arg\max_{\mathcal{G} \in \mathcal{G}^n} f(\mathcal{G} : D) \tag{1}$$

where $f(\mathcal{G} : D)$ is a scoring metric which evaluates the merit of any candidate DAG $\mathcal{G}$ with respect to the dataset D, and $\mathcal{G}^n$ is the set containing all the DAGs with n nodes.

Efficient evaluation of neighbors DAGs in Local Search algorithms, such as Hill-Climbing, is based on an important property of scoring metrics: *decomposability* in the presence of full data. In the case of BNs, decomposable metrics evaluate a given DAG as the sum of its node family score, i.e., the subgraphs formed by a node and its parents in $\mathcal{G}$.

2.1. Search Space of Orderings

Early algorithms for BN structure learning assumed that a given order was known and then they searched for a network consistent with such order. In this sense, the K2 [7] algorithm is the most usual choice.

Given an ordering $\prec$, we can define the possible parent sets for any given variable X_i: $\mathcal{U}_{i,\prec} = \{\boldsymbol{U} : \boldsymbol{U} \prec X_i\}$, where $\boldsymbol{U} \prec X_i$ is defined to hold when all nodes in $\boldsymbol{U}$ precede X_i in $\prec$. The optimal parent set for each node X_i is simply:

$$Pa^*_{\mathcal{G}}(X_i) = \arg\max_{\boldsymbol{U} \in \mathcal{U}_{i,\prec}} f(X_i|\boldsymbol{U} : D) \tag{2}$$

As the decisions for different nodes do not constrain each other, this set of selected families provides the optimal network $\mathcal{G}^*_{\prec}$ consistent with $\prec$. Without any restriction in the number of parents for the variables, we note that any acyclic (directed) graph is consistent with some ordering. Hence, the optimal network is simply the network $\mathcal{G}^*_{\prec}$ given by:

$$\prec^* = \arg\max_{\prec} f(\mathcal{G}^*_{\prec} : \mathcal{D}) \tag{3}$$

We can therefore find the optimal network by finding the optimal ordering, where the score of an ordering is the score of the best network consistent with it. However, we still have the problem of finding the best parent set for the node X_i. This problem is, in general, exponential. In this paper we propose to carry out this search with the algorithm K2 equipped with a delete operation apart from the usual adding operation. At each step, the modified algorithm K2, which we are going to name $K2M$, analyzes all the possible addition or deletion operations, and it can choose independently for each variable X_i, the one with the highest positive difference with respect to f. This last algorithm has been already used in other papers [2, 8], the first one with an Estimation of Distribution Algorithm (EDA) and the second with a local search algorithm. However, $K2M$ has not been

[1] We use standard notation, that is, bold font to denote sets and n-dimensional configurations, calligraphic font to denote mathematical structures, upper case for variables or sets of random variables, and lower case to denote states of variables or configurations of states (vectors).

proven yet to be able to find the optimal subset of parents for each variable. Next, the definition and results necessary to do that are described [4].

A probability distribution p is contained in a DAG $\mathcal{G}$ if there exists a set of parameter values $\mathbf{\Theta}$ such that the Bayesian network defined by $(\mathcal{G}, \mathbf{\Theta})$ represents p exactly. Of course, if two graphs are correct, then the sparser one should receive more merit. It is the basis to define a criterion score to be *consistent*. Locally Consistent scoring criterion is defined as: let $\mathcal{G}$ be any DAG, and $\mathcal{G}'$ the DAG obtained by adding edge $X_i \rightarrow X_j$ to $\mathcal{G}$. A scoring metric is *locally consistent* if in the limit as data grows large the following two conditions hold:

1. If $\neg I_p(X_i, X_j | Pa_{\mathcal{G}}(X_j))$, then $f(\mathcal{G} : D) < f(\mathcal{G}' : D)$
2. If $I_p(X_i, X_j | Pa_{\mathcal{G}}(X_j))$, then $f(\mathcal{G} : D) > f(\mathcal{G}' : D)$

Chickering [4] also proves that the BDe, BIC, MDL scoring criteria are score equivalent, consistent and locally consistent when we suppose that D constitutes a sample which is isomorphic to a graph. The following proposition gives us the way to evaluate correctly (asymptotically) a given order by using the $K2M$ algorithm:

Proposition 1 *Let D be a dataset containing m iid samples from some distribution p. Let $\hat{\mathcal{G}}$ be the DAG obtained by running the $K2M$ algorithm for each variable X_i, $\hat{\mathcal{G}} = K2M(\prec, D)$. If the metric f used to evaluate DAGs is consistent and locally consistent, then $\hat{\mathcal{G}}$ is a* **minimal I-map** *of p in the limit as m grows large.*

Proof. *First we prove that $\hat{\mathcal{G}}$ is an I-map of p. Let us suppose the contrary, i.e., $\hat{\mathcal{G}}$ is not an I-map of p. Then there is at least one pair of variables X_i and X_j such that $\langle X_i, X_j | Pa_{\hat{G}}(X_i) \rangle_{\hat{\mathcal{G}}}$ and $\neg I_p(X_i, X_j | Pa_{\hat{\mathcal{G}}}(X_i))$. Thus, $\hat{\mathcal{G}}$ cannot be a local optimum of f because the addition of arc $X_j \rightarrow X_i$ has a positive difference.*
Now we prove the minimal condition. Again let us suppose the contrary, that is, there exists $X_j \in Pa_{\hat{\mathcal{G}}}(X_i)$ such that $I_p(X_i, X_j | Pa_{\hat{\mathcal{G}}}(X_i) \setminus \{X_j\})$. If so, $\hat{\mathcal{G}}$ cannot be a local optimum because there is (at least) one deletion operation with a positive difference. ■

This is an important result since, given the correct order among the variables it guarantees, under faithful conditions as GES algorithm does, that $K2M$ would obtain the correct Bayesian network.

3. Local algorithms. HCbO and VNSbO

In this paper we are going to focus on two local methods which use the *K2M* algorithm to transform an ordering into a Bayesian Network. As mentioned, due to proposition 1, they obtain a minimal I-map.

In order to define a local method, a neighbourhood operation to transform one solution to another has to be addressed. In this case, we have selected the insertion. *Insert*$(\prec, i, j)$ is defined as follows: for different indexes i and j, the insert operator modifies the ordering $\prec$ generating a new ordering $\prec'$ by moving the variable in position i to the position j in the new ordering. If $j > i$ the variables in positions $(i, j]$ are moved one position backward in the ordering, else if $j < i$ the variables in positions $[j, i)$ are moved one position fordward in the ordering.

Once a local operator, as the previous one, is defined, a greedy Hill-Climbing algorithm based on Orderings (HCbO) can easily be implemented. On each iteration, the algorithm moves from the ordering $\prec_k$ to the ordering $\prec_{k+1}$ applying the neighbourhood operation with the highest positive difference with respect to $\prec_k$. The algorithm stops when there is not a neighbour ordering better than the current one.

The evaluation of all the possible insert operations in an iteration of the HCbO algorithm can be very expensive in terms of computation. We propose three improvements that speed up these operations significantly.

The first improvement, used in previous works, allow the reduction of the number of computations of parent sets in any insert operation, $\prec'=$ *Insert*$(\prec, i, j)$, from n to $|i - j| + 1$. Simply, the computation of the parent sets for those variables that are not located between variables at positions i and j can be omitted, as the set of variables preceding them is the same in $\prec$ and $\prec'$.

The second improvement speed up the computation of some insert operations, *Insert*$(\prec, i, j)$, if it has already been evaluated the insertion of the variable in position i at another position. Figure 1 shows the insertion of a variable at different positions. Let suppose that $\prec'$=*Insert*$(\prec_{k-1}, 2, 6)$ has already calculated and let us focus in the operation $\prec''$=*Insert*$(\prec_{k-1}, 2, 7)$. Taking into account the previous improvement, it is only needed to recompute the parent set of the variables between C and B. But, as we can see in the figure, the ordering of the variables between A and F in $\prec'$ and $\prec''$ is the same. So, it only needs to recompute two different parent sets in this operation (G and B).

These two last improvements allow an efficient implementation of the insert operation. It is possible to compute only two parent sets in each insert operation if they are computed in a correct order. That is, computing *Insert*$(\prec, i, j-1)$ before *Insert*$(\prec, i, j)$ if $j > i + 1$ and *Insert*$(\prec, i, j+1)$ before *Insert*$(\prec, i, j)$ if $j < i - 1$.

The third improvement uses the computations done in an iteration of the algorithm to skip some computations in the next iteration. In order to clarify the explanation, we explain this improvement individually, without considering the previous improvements. Figure 1 shows an example of the insert operation in two consecutive iterations. The al-

gorithm has moved from $\prec_{k-1}$ to $\prec_k$ using the operation *Insert*$(\prec_{k-1}, 3, 5)$. In iteration k, for instance, the algorithm needs to compute the operator *Insert*$(\prec_k, 2, 7)$ and note that *Insert*$(\prec_{k-1}, 2, 7)$ was already computed. If we observe the figure, we note that variable A and the subset of variables between F and H are the same in the operations in $\prec_{k-1}$ and $\prec$. So, it only needs to compute the new parent set for variables C, D and E, that is, the variables modified by the current insertion but also by the previous accepted insertion. As a direct consequence of this improvement, if the intersection between the previous accepted insertion and the current insertion is null the computation of the parent sets for all the variables can be omitted.

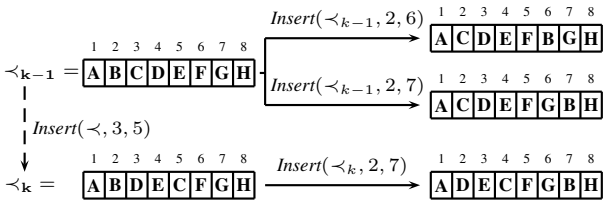

Figure 1. Example of the insert operation

Even considering the improvements presented, the computation of the insertions could be very expensive. For this reason, the algorithm uses a parameter (named maximum radius) that limits the possible operations. If a maximum radius r_m is set, for each variable X in position i, it only considers the insertion of this variable at positions $j \in [i - r_m, i + r_m]$. This approach limits the neighbourhood of an ordering, but it is still possible to reach a larger movement with two or more intermediate steps.

Given an ordering, the number of neighbour orderings is directly proportional to the maximum radius. So, for big values of the maximum radius, the number of neighbours that the HCbO needs to evaluate in each iteration can still be high. Therefore, it seems worth starting with a small value for the maximum radius and increase it during the execution of the algorithm.

The Variable Neighbour Search [14] algorithm works with different neighbourhood definitions. In our case, called VNSbO, each neighbourhood definition is determined by the value of the maximum radius. At each iteration of the algorithm, if none of the neighbours of the current ordering improves it, the maximum radius r is increased in one unit. Otherwise, the algorithm moves from the current ordering to the best neighbour ordering and set r to 1. The algorithm stops when r is bigger than the maximum radius previously established. If we set r_m equal to $n - 1$ (number of variables minus one) we say that we have a VNSbO without radius restriction.

4. Experimental evaluation

4.1. Experimental Setup

In this section, the described algorithms are compared with the standard Hill-Climbing in the space of DAGs (HC), and the Greedy Equivalence Search Algorithm (GES) [4].

The actual implementation of all the algorithms was coded in Java and interacts with the `WEKA` library [16] for dataset management. Our algorithms are implemented including all the improvements presented previously. The implementation of HC is done including the common optimizations suggested in the literature. The implementation of GES is based on the one included in `Tetrad 4.3.9` [2]. We added the optimizations suggested in [4] and modified the scoring function to be exactly the same that we use in the rest of the algorithms.

The score metric used in the algorithms is the Bayesian Dirichlet equivalent in their uniform prior version BDeu [11]. The equivalent sample size (N') used in the experiments is 10 and the network priors are calculated as in [5] where $k = 1/(N' + 1)$. For HCbO and VNSbO we report the execution of 30 independent runs of the algorithm with 30 random initial orderings respectively. For HC and GES the initial DAG is the empty one as usual.

We have selected 8 networks to test the algorithms with different sizes and characteristics and from different sources: 1. ALARM, 2. BARLEY, 3. CHILD, 4. HAILFINDER, 5. INSURANCE, 6. MILDEW, 7. MUNIN version1 and 8. PIGS, all of them commonly used in the literature. In these cases, we have easy domains with few variables like ALARM and CHILD and we have complex domains with hundred variables like BARLEY, MUNIN or PIGS. For each one of these networks we obtained a dataset by sampling the network with 5000 instances.

4.2. Empirical results

In order to compare the algorithms described in the previous section, we considered two kinds of factors as performance indicators: the quality of the network obtained by the algorithm, given by the value of the score metric (BDeu) for the resulting model; and the complexity of each algorithm, given by the number of computations of the score metric.

Table 1 shows the results of the algorithms HC, GES and the unconstrained versions (no radius restriction) of HCbO and VNSbO.

For a more rigorous analysis of the BDeu results in Table 1, we performed a Friedman rank test [10] as suggested by

[2] http://www.phil.cmu.edu/projects/tetrad/

	BDeu				Score computations			
	HC	GES	HCbO	VNSbO	HC	GES	HCbO	VNSbO
1.	-49313	-49163	-48706	**-48696**	3201	4888	14759	11155
2.	-290677	**-286309**	-287209	-287696	4744	8067	11117	9883
3.	-61896	**-61761**	-61775	-61823	901	1829	2361	1901
4.	-253169	**-253010**	-253137	-253127	6541	14011	18728	15582
5.	-68542	-68216	**-68136**	-68148	1894	3063	6541	5191
6.	-259433	**-259262**	-259418	-259438	2305	3310	4211	3793
7.	-236081	-231365	-223367	**-223019**	75766	127236	322563	235634
8.	-1684333	**-1681903**	-1682400	-1682441	527089	552732	2459935	1832388

Table 1. BDeu score and computations of the metric

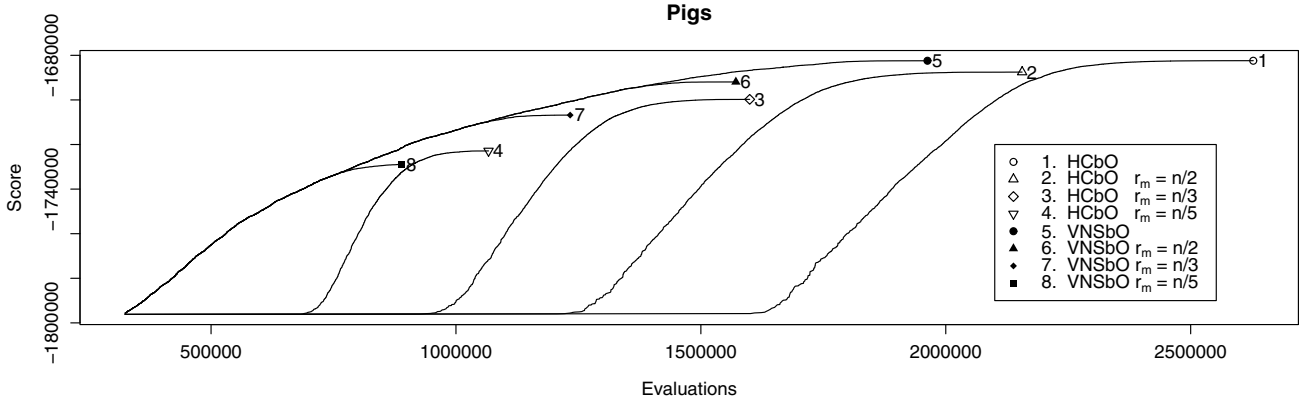

Figure 2. BDeu score versus score computations

Demšar [9] to compare the relative performances of multiple algorithms across multiple data sets. With a 95% confidence level, we can reject the hypothesis that all algorithms are equivalent for all eight scenarios. We also performed a post-hoc analysis using the Bergmann-Hommels procedure [1], and we can conclude that there are only statistical differences between HC and all other algorithms.

With respect to the number of computations of the metric, HC is the one that performs less computations followed by GES, then VNSbO and the one that performs more computations is HCbO.

We also performed an analysis of how the maximum radius affect both accuracy and speed of the algorithms. The maximum radius is set proportionally to the number of variables in the dataset using factors $1/5$, $1/3$ and $1/2$. The results are shown in Figure 2 where we plot the BDeu score versus the total number of score computations at each iteration of the HCbO and VNSbO algorithm for the PIGS dataset. We show the graph for the PIGS dataset as example, but it is important pointing out that, for all the datasets, the graph shows the same behaviour. From these experiments, we can reach the following conclusions:

- As expected, the final score increases as the maximum radius increases and the number of computations decreases as the maximum radius decreases.
- HCbO performs a large number of computations in the first iteration. Smaller values of the radius ease this situation. By contrast VNSbO does not have this problem because of the use of small radius in the initial steps.
- Constraining the radius in VNSbO determines when the algorithm finishes. However, the unconstrained version of VNSbO outperforms the constrained versions for any given number of score computations.
- Comparing any version of HCbO with the unconstrained version of VNSbO, we see that HCbO only reach the accuracy of VNSbO at the end of its execution, so if we want to use a good any-time algorithm the recommendation is clearly VNSbO algorithm. In fact, if we set a limited number of statistical computation we can observe in figure 2 that there is always a version of VNSbO algorithm that outperforms the best HCbO algorithm at this point.

5. Conclusions and future work

In this paper we have presented a correct way to evaluate a given ordering when using the space of orderings

for structural learning of Bayesian Networks, which consists of a modified version of the well-known K2 algorithm equipped with a deletion operation. Moreover, we have introduced a new neighbourhood operator based of insertion as an alternative of the classical swapping operation. This last choice allows improvements in the neighbourhood computation presented also in this paper, saving a lot of parent set computations. Finally we have presented two local methods based on the previous operator, one is the classical Hill-Climbing search and the second is a VNS method. This last method can be viewed as a natural extension of the use of the parameter radius in the search that automatically is able to adapt this parameter during the search.

From the experiments, we can conclude that HCbO and VNSbO improve the classical HC based on search the DAGs space in terms of accuracy, but HC is, in general, faster. Constraining the radius can be a good choice to speed up the algorithms in exchange of getting a little worse results. However, using VNSbO that automatically tunes the radius appears to be better option than using HCbO.

In relation with GES algorithm, considered the reference algorithm for BN learning, results obtained with the proposed algorithms do not present significant difference with those obtained with this algorithm. However, if comparing from a practitional point of view, they are significantly easier to implement.

In a future we plan to study how to save more computations in order to speed up the algorithms, this objective can be done by searching more restricted neighbourhoods, but assuring that the behaviour of the algorithms is the same. Also we can use the structure of the BN for a given ordering to characterize the minimal number of operations to be evaluated in the next step of the local algorithm.

References

[1] G. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In P. Bauer, G. Hommel, and E. Sonnemann, editors, *Proc. Symp. on Multiple Hypotheses Testing*, pages 110–115, Berlin, 1988. Springer.

[2] R. Blanco, I. Inza, and P. Larrañaga. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18(2):205–220, 2003.

[3] W. Buntime. Theory refinement on bayesian networks. In *Proceedings of the seventh conference (1991) on Uncertainty in Artificial Intelligence (UAI'91)*, pages 52–60, Los Angeles, California, United States, 1991. Morgan Kaufmann Publishers Inc.

[4] D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[5] D. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128, 1995.

[6] D. M. Chickering. Learning bayesian networks is np-complete learning from data. In *Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, fisher, d. & lenz, h. j edition, 1996.

[7] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[8] L. de Campos and J. Puerta. Stochastic local algorithms for learning belief networks: Searching in the space of the orderings. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, number 2143 in Lecture Notes in Artificial Intelligence, pages 228–239. Springer Verlag, 2001.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[10] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[11] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[12] F. Jensen and T. Nielsen. *Bayesian networks and decision graphs*. Springer, 2007.

[13] 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.

[14] N. Mladenović and P. Hansen. Variable neighborhood search. *Comps. in Opns. Res.*, 24:1097–1100, 1997.

[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, 1988.

[16] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.