Towards consistency in general dependency networks

José A. Gámez and Juan L. Mateo and José M. Puerta
Computing Systems Department
Intelligent Systems and Data Mining Group – $i^3\mathcal{A}$
University of Castilla-La Mancha
Albacete, 02071, Spain

Abstract

Dependency networks are a probabilistic graphical model that claim several advantages from other models like Bayesian networks and Markov networks, for instance. One of these advantages in general dependency networks, which are the object of study in this work, is the ease of learning from data. Nonetheless this easiness is also the cause of its main drawback: inconsistency. A dependency network cannot encode the probability distribution underlay in the data but an approximation. This approximation can be enough good for some applications but not in other cases. In this work we make a study of this inconsistency and propose a method to reduce it. From the conclusions we have taken from this analysis we have developed an algorithm that has to be run after the standard learning algorithm yields its solution. Our method is an heuristic approach so we cannot assure that the resulting model is fully consistent, however we have carried out some experiments which make us to think that it produces high quality models and therefore is advisable its use.

1 Introduction

Probabilistic graphical models (PGM) (Lauritzen, 1996; Jensen and Nielsen, 2007) have been deeply under research and have been used in many applications in the last two decades because of their capabilities. There are several kinds of PGMs like decision graphs or Markov networks (MN), but probably the most famous and used are Bayesian networks (BN).

Dependency networks (DN) are a probabilistic graphical model proposed by (Heckerman et al., 2000) as an alternative to BN. The main difference between them is that the graph in DN does not have to be acyclic. The parametric component is the same, i.e. every variable has a conditional probability distribution given its parents. Another difference is that in DNs the parents for each variable is its *Markov Blanket* (MB) in the Bayesian network encoding the same domain. This is the reason why the graph of a dependency network can be cyclic.

In (Heckerman et al., 2000) are presented some tasks in which DNs can be worthwhile like probabilistic inference, collaborative filtering and visualization of relationships. Nonetheless, from the automatic learning point of view DNs have a drawback because its not easy to learn a set of conditional probability distributions (CPD) consistent with the joint probability distribution (JPD). That is the reason the authors relaxed the definition of DNs and they defined *general dependency networks*, however now we cannot expect that with the set of CPDs we were able to recover the JPD but an approximation. Heckerman et al. (2000) argue that this approximation can be better as the amount of data used in the learning process increases, however it still is an approximation.

In this work we want to analyze how this approximation can deteriorate the performance of a DN model and we propose a way to improve the whole model with a minimun computational cost or even speeding up the learning process.

In Section 2 we present a more formal and detailed definition of DNs. In Section 3 we make an analysis of the inconsistencies in general DNs. In Section 4 we explain our proposal to reduce those inconsistencies. In Section 5 we describe some experiments we have carried out in order to validate our proposal and show the results, and in Section 6 we conclude.

2 Dependency Networks

2.1 Consistent Dependency Networks

Given a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ with a positive JPD $P(\mathbf{X})$, a *consistent dependency network* for this domain consists of a pair $(\mathcal{G}, \mathcal{P})$ where $\mathcal{G}$ is a directed graph (not necessarily acyclic), in which every node represents a variable, and $\mathcal{P}$ is a set of CPDs. In $\mathcal{G}$ the set of parents for each variable X_i, denoted by $\mathbf{Pa}_i$, is formed by all those variables such that verify

$$P(X_i|\mathbf{Pa}_i) = P(X_i|X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n). \quad (1)$$

So if $P(\mathbf{X})$ is faithful to a graph, what is a common assumption in machine learning algorithms, then the parents for a given variable in a DN are its MB. Other way to say so is that a DN has the same adjacencies than a MN.

A DN is consistent in the sense that all the CPDs in $\mathcal{P}$ can be obtained from the JPD $P(\mathbf{x})$, i.e. we can obtain $P(\mathbf{X})$ from $\mathcal{P}$ in a similar way than in an BN or MN by the product of local probability distributions:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i)$$

In (Heckerman et al., 2000) is shown the equivalence between DNs and MNs. The only difference is that in MNs the quantitative component is provided by potential functions whereas in DNs is provided by CPDs. Given this equivalence one approach to learn DNs from data can be to learn a MN in order to obtain the structure and then compute the set of CPDs from the MN via probabilistic inference. Other possibility suggested also in that paper is to learn another probabilistic model (a BN for instance) and translate it into a DN. Nonetheless the problem with these approaches is that the conversion can be computational expensive and inefficient in many cases. That is the reason why the authors presented another definition for DNs more relaxed in order to ease the automatic learning from data. This new definition is covered in next section.

2.2 General Dependency Networks

A consistent DN is not attractive from an machine learning point of view because of the difficulties related with obtaining the set of CPDs, specially with the restriction that this set has to be consistent with the JPD for the variables in the domain. So *general* DNs, also described in (Heckerman et al., 2000), are based on the idea of removing those restriction about consistency. Thus every single CPD $P(X_i|\mathbf{Pa}_i)$ can be estimated independently from the others by any probabilistic classification method, as a probabilistic decision tree (PDT) (Buntine, 1991). Once we have all the CPDs we can build the structure of the dependency network from the (in)dependencies that are appeared during the learning process.

This way of learning a DN can be more efficient than learning from a MN in many cases, and other advantage is that its parallelization is straightforward, what can report a great benefit if we are dealing with a domain with a large number of variables. Nonetheless this heuristic approach has a disadvantage, due mainly to the independent search over the variables and poor estimations in small datasets, the learned CPDs may not be consistent with the JPD, this can be called *parametrical inconsistency.* But also *structural inconsistencies* can appear because after learning the CPD we can see that, for instance, X_i can be parent of X_j but not the opposite, i.e. the CPD for X_i would not contain X_j but the CPD for X_j would contain X_i. In (Heckerman et al., 2000), authors argue that this inconsistencies can be reduced as the amount of data used for leaning increases.

A formal definition for this new model is as follows. Given a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, consider the set of CPDs $\mathcal{P} = \{P_1(X_1|\mathbf{X}\backslash X_1), P_2(X_2|\mathbf{X}\backslash X_2), \ldots, P_n(X_n|\mathbf{X}\backslash X_n))\}$. It is not required that these distributions are consistent with $P(\mathbf{X})$, i.e. it is not required that this set can be obtained via inference from the JPD. Under these conditions a *dependency net-*

work for $\mathbf{X}$ and $\mathcal{P}$ is the pair $(\mathcal{G}, \mathcal{P}')$, where $\mathcal{G}$ is a directed graph usually cyclic and $\mathcal{P}'$ is a set of CPDs such that

$$P_i(X_i|\mathbf{Pa}_i) = P_i(X_i|\mathbf{X}\backslash X_i) \qquad (2)$$

for every $P_i \in \mathcal{P}$.

2.3 Inference

In any case, with a consistent or general dependency network, given the likely existence of cycles in the graph we cannot use exact inference algorithms used in BNs and some of the approximate. In the case of consistent DNs it can be converted to a MN and use standard techniques for probabilistic inference over MNs. Nonetheless a more general option is suggested in (Heckerman et al., 2000) for both models, Gibbs sampling (Geman and Geman, 1984). Basically this method works by repeatedly cycling through each variable in a fixed order during all the process, and sampling each X_i according to $P(X_i|\mathbf{Pa}_i)$. This procedure is called *ordered Gibbs sampler* but in the case of a general DN, given that the CPDs may not be consistent with the JPD, is called *ordered pseudo-Gibbs sampler.* Besides in (Heckerman et al., 2000) it is developed a more efficient method which can avoid some sampling and it is called *modified ordered (pseudo-)Gibbs sampler.* This method, in order to get $P(\mathbf{Y}|\mathbf{Z})$ and the value of $\mathbf{Z}$ is $\mathbf{z}$ for a DN in a domain with a set of variables $\mathbf{X}$, is shown in Figure 1.

The key point is in line 6, since if the values for all the parents for a given variable are known we can avoid the sampling for that variable and just take its value from its CPD. This algorithm is justified by Equations (1) and (2).

However, given the modified ordered Gibbs sampler, there are some situations in which we can avoid completely the sampling. For instance if we use a DN as a classifier and we assume that we always know the values for all predictive variables (Gámez et al., 2006). Other case is when is needed to obtain the probability for a full configuration in a DN, i.e. $P(\mathbf{X})$ when we have fixed the value for every single variable. We can see this computation in other way

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{X}\backslash X_i).$$

```
1 $\mathbf{U} = \mathbf{Y}$ (* the unprocessed variables *)
2 $\mathbf{P} = \mathbf{Z}$ (* the processed and conditioning
      variables *)
3 $\mathbf{p} = \mathbf{z}$ (* the values of $\mathbf{P}$ *)
4 While $\mathbf{U} \neq \emptyset$
5   Pick $X_i \in \mathbf{U}$ s.t. $X_i$ has no more
        parents in $\mathbf{U}$ than any variable
        in $\mathbf{U}$
6   If all parents of $X_i$ are in $\mathbf{P}$
7     $P(X_i|\mathbf{p}) = p(X_i|\mathbf{Pa}_i)$
8   Else
9     Use modified ordered Gibbs
          sampling to get $P(X_i|\mathbf{p})$
10  $\mathbf{U} = \mathbf{U} - X_i$
11  $\mathbf{P} = \mathbf{P} + X_i$
12  $\mathbf{p} = \mathbf{p} + x_i$
13 Return the product of the conditionals
      $P(X_i|\mathbf{p})$
```

Figure 1: Modified ordered Gibbs sampler

If we take the right part of the equation and use the modified ordered Gibbs sampler we have that $\mathbf{Y} = \{X_i\}$ and $\mathbf{Z} = \{\mathbf{X}\backslash X_i\}$, in line 5 we have only one choice and in line 6 the condition is true so the sampling is avoided. Therefore we can compute statistics such as the likelyhood of a DN for a dataset, and we assume that the dataset does not contain missing data, without any sampling.

3 Analysis of Parametrical Inconsistency

In this section we want to analyze some issues regarding with inconsistency in DNs. From now on we consider only general DNs.

Example 1. Consider the case in which we have two variables, X and Y, and they are dependent, then the DN for this domain, $\mathcal{DN}$, should have a graph with two links $X \to Y$ and $X \leftarrow Y$.

Hence $\mathcal{P}' = \{P(X|Y), P(Y|X)\}$ for $\mathcal{DN}$. However is clear that $P(X,Y) \neq P(X|Y) \cdot P(Y|X)$. In fact

$$\hat{P}(X,Y) = \frac{P(X,Y) \cdot P(X,Y)}{P(X) \cdot P(Y)} = f(X,Y) \cdot P(X,Y) \quad (3)$$

where

$$f(X,Y) = \frac{P(X,Y)}{P(X) \cdot P(Y)}$$

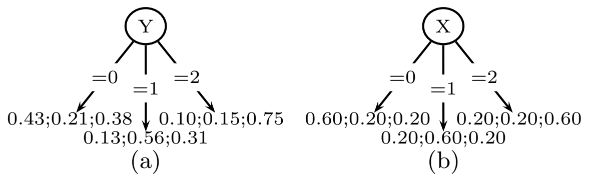
Therefore, even in a situation so simple like this one, we cannot expect to have a DN without inconsistencies, when it is learned from data of course. Moreover, looking at Equation (3) we can say that the inconsistency is smaller as the dependency between X and Y is weaker and $f(X,Y)$ tends to 1.

In (Heckerman et al., 2000) they propose learn DNs by means of probabilistic decision trees. This model are very good to encode contextual dependencies. Encoding the CPDs by probabilistic decision trees can help to reduce the inconsistencies because a decision tree tries to represent a more general probability distribution by pruning some branches which are similar. Then the dependence between the variables can be smoothed and thus $f(X,Y)$ is closer to 1. However if this happens we have a poorer estimation for the JPD even though is less inconsistent.

In order to illustrate that we can consider both variables in Example 1 are discrete with three states and their JPD is defined by this table:

	Y=0	Y=1	Y=2
X=0	0.12	0.04	0.04
X=1	0.06	0.18	0.06
X=2	0.10	0.10	0.30

When we compute $P(X|Y)$ and $P(Y|X)$ from $P(X,Y)$ in the way of a probability table or a full expanded probabilistic decision trees (see Figure 2 (a) and (b)) we obtain an estimation $\hat{P}_1(X,Y)$ which is shown in Figure 2(c):

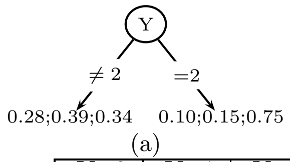


	Y=0	Y=1	Y=2
X=0	0.257	0.025	0.020
X=1	0.043	0.338	0.030
X=2	0.075	0.063	0.450

(c)

Figure 2: Joint probability distribution $\hat{P}_1(X,Y)$ (c) obtained using full decision trees for CPDs $P(X|Y)$ (a) and $P(Y|X)$ (b).

We can see large differences between the true figures and the estimation. Besides we can check that $\hat{P}_1(X,Y)$ is not a probability distribution because $\sum_{x,y}\hat{P}_1(x,y) = 1.301 \neq 1$. If, for instance, the learning procedure decides to change the representation of $P(X|Y)$ for a probabilistic decision tree in which branches for values 0 and 1 are merged (Figure 3(a)), because they are the most similar, then we obtain a new estimation $\hat{P}_2(X,Y)$ which is shown in Figure 3(b). $\hat{P}_2(X,Y)$ still differs from $P(X,Y)$ but is closer to it than $\hat{P}_1(X,Y)$ in average, and also is closer to be a probability distribution because $\sum_{x,y}\hat{P}_2(x,y) = 1.170$.



	Y=0	Y=1	Y=2
X=0	0.166	0.055	0.020
X=1	0.078	0.233	0.030
X=2	0.069	0.069	0.450

(b)

Figure 3: Joint probability distribution $\hat{P}_2(X,Y)$ (b) obtained using a simpler decision tree for $P(X|Y)$ (a).

Therefore in spite of the use of probabilistic decision trees we still have an approximation which could not be good enough for some applications. In next section we present a simple heuristic method that can reduce inconsistencies in DNs improving its accuracy.

4 How to Improve Consistency

As it has been seen in the previous Section even in a simple case like Example 1 we cannot expect to get a consistent DN if the CPDs are learned independently. If two variables are dependent this equation $P(X,Y) = P(X|Y) \cdot P(Y|X)$ will never be true, nonetheless this expression $P(X,Y) = P(X) \cdot P(Y|X) = P(X|Y) \cdot P(Y)$ is always true and does not matter if both variable are dependent or not. Bearing this in mind, in Example 1 we can ensure consistency if at the end of the learning process we realize that X is a predictive variable for Y and vice versa and then instead of maintaining both CPDs we replace $P(X|Y)$ by $P(X)$ or $P(Y|X)$ by $P(Y)$. This is the basic idea of our proposal, but there is not so easy when there are more variables involved. In that case we do not expect to obtain the best set of probabilities whose composition yield the right JPD, but a good approximation and, more important, more consistent.

More precisely the proposal consists in estimating a set of CPDs of a BN that encode the same (in)dependencies that the learned DN. We have to point out that our proposal only changes the set of probability distributions but not the graph, so the model learned still has the same advantages about visualization. However, given that the relationships represented in a DN can be encoded by several BNs with different factorizations of the JPD, and that the conversion from DN to BN can not be attractive from the computational point of view, this proposal is based on a heuristic approach whose complexity order is linear in the number of dependencies found. The method proposed is shown in Figure 4.

```
1 For each variable $X_i$
2   For each $Y_j$ parent of $X_i$
3     If $X_i$ is also a parent of $Y_j$
4       If the conditioning set of $X_i$ is
              grater that $Y_j$'s
5         $Y_j$ is removed as parent of $X_i$
6       Else
7         $X_i$ is removed as parent of $Y_j$
```

Figure 4: Proposed method to obtain a more consistent set of CPDs.

We can call this new step in the learning process as parametric reduction. An important point in this procedure is in line 4. With this condition we want avoid large conditioning sets, what can reduce overfitting in the parameters estimation. The order in which the links can be traversed can be any although not all of then will yield the same solution. The reason for that is the heuristic nature of this algorithm and that a more sophisticated search would not be interesting for practical reasons. One of the benefits of DNs is ease of learning so we do not want to change that by introducing a complicated post-learning algorithm.

After performing this step is needed to recompute every probability distribution which has been modified. In the case that these probability distributions are in form of probability trees, if the removed variables are in the leaves the only thing to do is to aggregate its values to the up node in the tree, otherwise the entire tree should be re-built. However, if in the learing process we have cached the statistics the new tree can be built without computational cost. In the case of probability tables we can postpone leaning these tables after that step.

5 Experimental Results

This section is devoted to evaluate our proposal with some experiments. Our testing framework is base on the one used in (Heckerman et al., 2000) for testing probabilistic inference with real data. We use the same score function for a test dataset with N instances $\{d_1, \ldots, d_N\}$ and n variables:

$$score(d_1, \ldots, d_N|model) = -\frac{\sum_{i=1}^{N} \ln P(d_i|model)}{nN}. \quad (4)$$

However, instead of using real dataset we prefer using data sampled from known BNs. The reason is that we want focus only in parametrical learning and inference so if the real dependencies are known we can give this information to the different algorithms in order to avoid that the results were affected by the structural learning. Next we present a detailed description of our experimentation.

5.1 Description of the Experiments

We have selected seven BNs from different sources: `alarm` (Beinlich et al., 1989), `asia` (Lauritzen and Spiegelhalter, 1988), `car-starts` and `headache` (Elvira Consortium, 2002), `insurance` (Binder et al., 1992), `credit` (DSL) and `water` (Jensen et al., 1989), which is a dynamic network and we have use only the two first slices. Some details of these networks can be seen in Table 1. From each of these networks we have sampled two datasets with 5000 instances each one, one for training and one for testing.

We have defined eight models to make a comparison between them. First one is the reference model and is a BN in which the structure is fixed

Table 1: Set of Bayesian networks used in our experiments.

network	Num. vars	States range	Aver. states	MB range	Aver. MB
alarm	37	2-4	2.84	1-12	3.89
asia	8	2-2	2.00	1-5	2.50
car-starts	18	2-3	2.06	1-9	3.44
credit	12	2-4	2.83	2-6	3.67
headache	12	1-4	2.92	1-4	2.67
insurance	27	2-5	3.30	1-16	6.22
water	16	3-4	3.63	1-12	6.00

with the real links (BN-f). Second model is an empty network (Empty). Next we have three dependency networks models, one with probability tables in which links have been fixed from the real MB for each variable in the network (PT-f), other with probabilistic decision trees learned from data (PDT), and other with probabilistic decision trees but in which the search space for each PDT have been restricted to the real MB (PDT-f). In both cases we use the suggested value for $\kappa = 0.1$. For any of these three models we have another version in which we have used our method for reducing the CPDs. These new models are labeled with an asterisk (PT-f*, PDT* and PDT-f*). In all cases parameters are learned from data by using Laplace smoothing.

Every model has been learned with each training dataset. For all of them it has been computed their score (Equation (4)) with the test dataset. As the model BN-f is the reference one we have also obtained the absolute difference of score between each model and BN-f. This value is more informative because we are looking for models closer to the true probability distribution what is represented by BN-f. Besides, we have computed also the summation of all possible configurations, i.e. total joint probability, which should be equal to 1, but only for those models with a tractable number of configurations (asia, car-starts, credit, headache).

5.2 Results

In Table 2 we report the score value for every model and dataset. At the botton line we show the average value for each model. Lower values should indicate a better model, so all pure DN models should be taken as the best ones. However that does not make sense because they are even better than our reference model (BN-f) which represents the true JPD. The reason is that, as we have seen in Section 3, inconsistent DNs tend to have greater probability values in average so their score is lower. That is the reason why we prefer paying more attention to the difference with respect to the reference model.

Thus, these new results are shown in Table 3. There we can see that always the model closer to BN-f is the one in which we have applied our proposal. Specially the model based on probability tables is always the best one but in two datasets. Also is important to notice that our proposal improves the original model in every dataset for PT-f model. However, in PDT model our proposal deteriorates the accuracy in alarm and headache dataset, although in average its application improves the global accuracy.

Another interesting point is that PDT models without our proposal are much better than PT-f. That corroborate the idea that for DNs the use of more general encoding for the CPDs is advisable despite that this encoding is also an approximation in many cases.

Previous results give us an idea about the quality of those model. We can suppose that the increment in accuracy must be related with the reduction in the inconsistency. Additionally we have checked if the models encode a real probability distribution, i.e. whether the total joint probability for a given model is equal to one. This computation has been only done for the models learned with the smaller networks because this computation is computationally unfeasible for the others. The result is shown in Table 4. According to the table is clear that the pure DN models are quite far form being a probability distribution, but our proposal achieve that condition for all of them.

5.2.1 Time

Given that our proposal is a post-process to any learning algorithm, it seems that we will need more running time. In our experiments involving PDT we see that running time increases

Table 2: Score for each model and dataset.

	BN-f	Empty	PT-f	PT-f*	PDT	PDT*	PDT-f	PDT-f*
`alarm`	0.282	0.397	0.173	0.298	0.253	0.342	0.242	0.338
`asia`	0.287	0.342	0.224	0.289	0.225	0.287	0.225	0.289
`car-starts`	0.127	0.175	0.070	0.127	0.070	0.136	0.070	0.127
`credit`	0.879	0.959	0.765	0.886	0.807	0.888	0.807	0.900
`headache`	0.435	0.609	0.214	0.435	0.419	0.585	0.419	0.593
`insurance`	0.490	0.651	0.399	0.519	0.420	0.556	0.420	0.550
`water`	0.401	0.410	0.417	0.410	0.388	0.409	0.388	0.408
	0.414	0.506	0.323	0.423	0.369	0.458	0.367	0.458

Table 3: Absolute score difference between BN-f and the other models.

	Empty	PT-f	PT-f*	PDT	PDT*	PDT-f	PDT-f*
`alarm`	0.115	0.110	**0.015**	0.029	0.060	0.040	0.056
`asia`	0.055	0.062	0.002	0.062	**0.000**	0.062	0.002
`car-starts`	0.048	0.057	**0.000**	0.057	0.009	0.057	**0.000**
`credit`	0.080	0.114	**0.007**	0.071	0.009	0.071	0.021
`headache`	0.174	0.222	**0.000**	0.017	0.150	0.017	0.158
`insurance`	0.161	0.092	**0.029**	0.070	0.066	0.071	0.059
`water`	0.009	0.016	0.010	0.013	**0.008**	0.013	0.007
	0.092	0.096	**0.009**	0.046	0.043	0.047	0.043

Table 4: Total joint probability for tested models.

	BN-f	Empty	PT-f	PT-f*	PDT	PDT*	PDT-f	PDT-f*
`asia`	1.00	1.00	3.60	1.00	3.42	1.00	3.42	1.00
`car-starts`	1.00	1.00	20.04	1.08	11.40	1.00	11.40	1.00
`credit`	1.00	1.00	6.41	1.00	4.26	1.00	4.26	1.00
`headache`	1.00	1.00	29.68	1.00	5.70	1.00	5.70	1.00

1% in average, but also we have to take into account that we do not include the improvements explained at the end of Section 4. However with PT, if we assume that structural learning will be the same in both cases and compare our post-process and parameter learning we can see in Table 5 that we always obtain a faster algorithm. The reason is that our post-process makes the probability tables to estimate are much smaller and so that parametrical learning will be much faster because the complexity order is $O(M \times S)$ where M is the number of instances in the dataset and S is the size of each table.

6 Conclusions

In this paper we have presented a method which aims to improve DNs. The main advantage of (general) DNs is that they can be learned from data easily, easier than BNs because of the lack

Table 5: Percentage of run time our proposal can reduce the original algorithm.

dataset	% reduction
`alarm`	43
`asia`	13
`car-starts`	23
`credit`	18
`headache`	11
`insurance`	95
`water`	98

of restrictions about cyclicity and easier than MNs because CPDs can be learn independently. Nonetheless this is also the main problem, the independent learning can lead to inconsistencies. They can be both structural and parametrical, however the later are more important. Whereas structural inconsistencies can be interesting for a better interpretation of the model (strong and weak dependencies), parametrical ones deteriorate model performance.

Thus our proposal is based on improving DNs accuracy by reducing CPDs, because, as it has been seen in Section 3, the use of full distributions is the cause of that inconsistencies. Is worthy to point out that our proposal does not change the qualitative component of a model, i.e. its links. This new method, that can be seen as a post-learning stage, works by trying to recover a set of CPDs similar to a BN which represents the same relationships between variables. In order not to lose the computational advantage of the DNs learning we have chosen a heuristic approach which has a linear complexity order in the number of links. Its heuristic nature can be object of complaint, nonetheless in our experimentation we have made clear its benefit.

We plan to extend this work in two lines as future work. First we plan to make a deeper analysis of our proposal checking the performance with different sample sizes and different ordering in the reduction step and see whether it affects the results. Besides we want to test probabilistic queries for different set of variables with evidence in which we have to use Gibbs sampling. The second line of work is applying this method to scenarios where DNs have been use in order to improve their results, such as classifiers or Estimation of Distribution Algorithms.

Acknowledgments

This work has been partially supported by Spanish Ministerio de Educación y Ciencia (TIN2007-67418-C03-01); Junta de Comunidades de Castilla-La Mancha (PBI-08-048) and FEDER funds.

References

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 *2nd European Conf. on Artificial Intelligence in Medicine*, pages 247–256, 1989.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2):213–244, 1992.

W. Buntine. Theory refinement on bayesian networks. In *Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

Decision Systems Laboratory DSL. Genie. http://genie.sis.pitt.edu/.

Elvira Consortium. Elvira: An Environment for Creating and Using Probabilistic Graphical Models. In *1st European Workshop on Probabilistic Graphical Models*, pages 222–230, 2002. http://leo.ugr.es/elvira.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Dependency networks based classifiers: learning models by using independence test. In *3rd European Workshop on Probabilistic Graphical Models*, pages 115–122, 2006.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 147–156, 1984.

D. Heckerman, D. M. Chickering, and C. Meek. Dependency networks for inference, collaborative filtering and data visualization. *Machine Learning Research*, 1:49–75, 2000.

F. V. Jensen and T. D. Nielsen. *Bayesian networks and decision graphs.* Springer, 2007.

F. V. Jensen, U. Kjærulff, K. G. Olesen, and J. Pedersen. Et forprojekt til et ekspertsystem for drift af spildevandsrensning (an expert system for control of waste water treatment — a pilot project). Technical report, Judex Datasystemer A/S, Aalborg, Denmark, 1989.

S. L. Lauritzen. *Graphical Models.* Oxford Univesity Press, 1996.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Royal Statistics Society, Series B*, 50: 157–194, 1988.