Learning Graphical Models with Hypertree Structure Using a Simulated Annealing Approach

Christian Borgelt and Rudolf Kruse
Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany

Abstract— **A major topic of recent research in graphical models has been to develop algorithms to learn them from a dataset of sample cases. However, most of these algorithms do not take into account that learned graphical models may be used for time-critical reasoning tasks and that in this case the time complexity of evidence propagation may have to be restricted, even if this can be achieved only by accepting approximations. In this paper we suggest a simulated annealing approach to learn graphical models with hypertree structure, with which the complexity of the popular join tree evidence propagation method can be controlled at learning time by restricting the size of the cliques of the learned network.**

I. Introduction

In recent years graphical models [20], [13] — especially Bayesian networks [16], [9] and Markov networks [12], but also the more general valuation-based networks [18] and, though to a lesser degree, the newer possibilistic networks [7], [3] — gained considerable popularity as powerful tools to model dependences in complex domains and thus to make inferences under uncertainty in these domains feasible. They are based on the idea that under certain conditions a multi-dimensional (probability or possibility) distribution can be decomposed into (conditional or marginal) distributions on lower dimensional subspaces. This decomposition is represented by a graph, in which each node stands for an attribute and each edge for a direct dependence between two attributes.

The graph representation also supports drawing inferences, because the edges indicate the paths along which information about attribute values has to be transmitted, usually in the form of local messages [9], [4]. However, in order to derive correct evidence propagation methods, the graphs have to satisfy certain conditions. In general, cycles pose problems, because they make it possible that the same information can travel on two or more different routes to other attributes. In order to avoid erroneous results in this case, the graphs are often transformed into singly connected structures, namely so-called *join* or *junction trees* [12], [9], [4]. An important step in this transformation is the construction of a graph with so-called hypertree structure, from which the join tree can easily be derived.

Since constructing graphical models manually can be tedious and time consuming, a large part of recent research in graphical models has been devoted to learning them from a dataset of sample cases [5], [8], [6], [7], [3]. However, most known learning algorithms do not take into account that the learned graphical model may later be used to draw time-critical inferences and that in this case the time complexity of evidence propagation may have to be restricted, even if this can only be achieved by accepting approximations. The main problem is that during join tree construction edges may have to be added, which can make the graph more complex than is acceptable. In such situations it is desirable that the complexity of the join tree (and thus the evidence propagation) can be controlled at learning time, even at the cost of a less exact representation of the domain under consideration. Therefore we suggest a learning algorithm that is based on the well-known paradigm of simulated annealing [15], [10] and that directly learns graphs with hypertree structure. In this way a transformation is not necessary and the complexity can be controlled at learning time by restricting the size of the cliques of the graph.

II. Hypertree Structure and Join Trees

As already mentioned above, a popular method for evidence propagation in multiply connected networks is so-called join tree propagation [12], [9], [4]. Here we only review briefly some of the basic notions that are used in connection with this method and which we need in the following.

The idea of join tree propagation is to transform the graph of a graphical model into a singly connected structure, namely a *join tree*. This is achieved by first modifying the graph in such a way that it has *hypertree structure*. This notion is defined by the so-called *running intersection property* [12]:

Definition 1: Let $\mathcal{M}$ be a finite family of subsets of a finite set U and let $m = |\mathcal{M}|$. $\mathcal{M}$ is said to have the *running intersection property* iff there is an ordering $M_1, \ldots, M_m$ of the sets $M_i \in \mathcal{M}$, such that

$$\forall i, 2 \leq i \leq m : \exists k, 1 \leq k < i : \quad M_i \cap \Big(\bigcup_{1 \leq j < i} M_j \Big) \subseteq M_k.$$

A undirected graph G is said to have *hypertree structure* if all pairs of nodes of G are (possibly indirectly) connected in G and the family $\mathcal{M}$ of the node sets that induce the maximal cliques[1] of G has the running intersection property. An order of the cliques that demonstrates the running intersection property is called a *construction sequence* for the graph. □

The idea underlying the notion of hypertree structure is as follows: In normal graphs an edge connects only two nodes. However, we may drop this restriction and introduce *hypergraphs*, in which we have *hyperedges* that can connect any number of nodes. It is very natural to use a hyperedge to connect the nodes of a maximal clique, because by doing so we can make these cliques easier to recognize. If the sets of nodes that are connected by hyperedges have the running intersection property, then the hypergraph is, in a certain sense, *acyclic*. Consequently, such hypergraphs are called *hypertrees*.

There is a simple algorithm to turn a given graph into one that has hypertree structure, which is based on the fact that *triangulating* the graph automatically ensures this property [19]. Here, however, we only need a test for this property [11]:

[1] A clique is a complete subgraph (a subset of nodes together with the corresponding edges, so that each node is (directly) connected to all other nodes in the subset), and it is maximal if it is not part of another complete subgraph.

Algorithm 1: (Graham reduction)

Input: A finite family $\mathcal{M}$ of subsets of a finite set U.

Output: Whether $\mathcal{M}$ has the running intersection property.

The family $\mathcal{M}$ of sets of elements is reduced by iteratively applying one of the following two operations:

1. Remove an element $A \in U$ that is contained in only one set $M \in \mathcal{M}$.
2. Remove a set $M_1 \in \mathcal{M}$ that is a subset of (or identical to) another set $M_2 \in \mathcal{M} \backslash \{M_1\}$.

The process stops if neither operation is applicable. If all elements $A \in U$ appearing in $\mathcal{M}$ could be removed, $\mathcal{M}$ has the running intersection property, otherwise it does not have it. □

Note that this algorithm is non-deterministic, since both operations may be applicable or one of them may be applicable to more than one element or more than one set, respectively. Note also that this algorithm yields a construction sequence: The reverse of the order in which the sets $M \in \mathcal{M}$ were removed from $\mathcal{M}$ obviously is such a sequence.

A graph with hypertree structure can be turned into a *join tree* [4]. In a join tree there is a node for each maximal clique of the graph and its edges connect nodes that represent cliques having nodes of the original graph in common. In addition, any node of the original graph that is contained in two join tree nodes must also be contained in all join tree nodes on the path between them. (Despite this strong requirement, a join tree for a given graph with hypertree structure need not be unique.) A join tree also provides an intuition for Graham reduction: Consider a join tree of a graph corresponding to a family $\mathcal{M}$. The node sets are removed by starting with those represented by the leaves of the join tree and then working inwards.

III. Learning Graphical Models

We now turn to how simulated annealing [15], [10] can be used to learn graphs with hypertree structure. Our main task is to find efficient methods to randomly generate and modify graphs with hypertree structure. We consider two alternatives, both of which exploit that in order to ensure hypertree structure we only have to make sure that the set of maximal cliques has the running intersection property, which guarantees acyclicity, and that there is a path between any pair of nodes, which guarantees connectedness (this latter condition may be relaxed).

Our first approach relies directly on the defining condition of the running intersection property, namely the ordering $M_1, \ldots, M_m$, which specifies a construction sequence. We select the node sets in this order starting with a random set M_1 of nodes. In step i, $i \geq 2$, a set M_k, $1 \leq k < i$, is selected at random and the set M_i, which is to be added in this step, is formed by randomly selecting nodes from $M_k \cup \left(U - \bigcup_{1 \leq j < i} M_j\right)$ making sure that at least one node from M_k and at least one node not in $\bigcup_{1 \leq j < i} M_j$ is chosen. This is repeated until each node is contained in some set M_j, $1 \leq j \leq i$. The probabilities of the different set sizes and the probability with which a node in M_k or a node in $\bigcup_{1 \leq j < i} M_j$ is selected are parameters of this method. Convenient choices are uniform distributions on sizes as well as on nodes.

In order to randomly modify a given graph with hypertree structure, we rely on the fact that Graham reduction can also be seen as a join tree pruning method: We simply execute a few steps of Graham reduction, randomly selecting the set to be removed if more than one can be removed at the same time, until only a certain percentage of the sets remain or only a certain percentage of the nodes is still covered. The reduced set $\mathcal{M}$ is then extended again in the same way in which it was generated in the first place (see above).

Unfortunately, this approach has a serious drawback: Suppose that by accident the initial graph is a simple chain. Then in each step only the two sets corresponding to the (hyper)edges at the ends of (the remainder of) the chain can be removed. Hence there is no or only a very small chance that the (hyper)edges in the middle of the chain are removed. In general, "inner cliques" are much less likely to be removed, since certain "outer cliques" have to be removed first. Therefore this method is severely biased, which renders it unsuited for most applications. Nevertheless, it was necessary to consider this approach first, since it directly suggests itself.

Our second approach is based on the insight that a family of node sets has the running intersection property if it is constructed by successively adding node sets M_i to an initially empty family observing to the following two conditions:

1. M_i must contain at least one pair of nodes that are unconnected in the graph represented by the node set family $\{M_1, \ldots, M_{i-1}\}$.
2. For each maximal subset S of nodes of M_i that are (directly or indirectly) connected to each other in the graph represented by $\{M_1, \ldots, M_{i-1}\}$ there must be a set M_k, $1 \leq k < i$, so that $S \subset M_k$.

It is clear that the first condition ensures that all nodes are covered after a certain number of steps. It also provides us with a stopping criterion. The running intersection property is ensured by the second condition alone.

With this method the family $\mathcal{M}$ can be constructed by forming *subfamilies* of node sets, each of which represents a connected component of the graph represented by the current family $\mathcal{M}$. That the resulting family actually has the running intersection property is ensured by the following theorem:

Theorem 1: If a family $\mathcal{M}$ of subsets of elements of a given set U is constructed observing the two conditions stated above, then this family $\mathcal{M}$ has the running intersection property.

Proof: Adding a node set to a given family $\mathcal{M}$ either adds isolated nodes (not contained in any subfamily) to a subfamily, or connects two or more subfamilies, or both. Hence one can show that the method referred to indeed results in a family $\mathcal{M}$ having the running intersection property by a simple induction argument, which proves that all subfamilies that are created during the construction have the running intersection property:

A subfamily with a single node set trivially has the running intersection property (induction anchor). So assume that all subfamilies up to a certain size, i.e. with a certain number of node sets, have the running intersection property (induction hypothesis). If a new node set only adds isolated nodes to a subfamily, then the enlarged family obviously has the running intersection property, because in this case the second condition stated above is equivalent to the last part of the defining condition of a construction sequence (cf. definition 1). Hence the construction sequence of the subfamily (which must exist due to the induction hypothesis) is simply extended by one set.

So assume that a new node set connects two or more subfamilies (and maybe adds some isolated nodes, too). In order to show that there is a construction sequence for the resulting subfamily of node sets, we show first that any set of a family of sets having the running intersection property can be made the first set in a construction sequence for this family: Reconsider the join tree illustration of Graham reduction. Obviously, the reduction can be carried out w.r.t. a join tree even if a given set (i.e. a given join tree node) is chosen in advance to be the last to be removed, simply because we can work from the (other) leaves of the join tree towards the corresponding node. Since the reverse of the order in which the node sets are removed is a construction sequence, there is a construction sequence starting with the chosen set, and since the choice is arbitrary, any set can be made the first of a construction sequence.

With this established, the remainder of the proof is simple: For each of the subfamilies connected by the new node set we find a construction sequence starting with the set M_k mentioned in the second condition. Then we form a construction sequence of the resulting enlarged subfamily: The new node set is the first set of this sequence. We append the construction sequences for the subfamilies, one after the other. The result is a construction sequence, because the sets M_k obviously satisfy the defining condition w.r.t. the first set due to the way in which they where chosen. Within the appended sequences the condition holds, because they are construction sequences for the subfamilies. There is no interaction between these sequences, because the subfamilies are node disjoint. Hence the new subfamily has the running intersection property. ■

The main advantage of this approach is that we can connect subfamilies of node sets, whereas with the first approach we can only extend one family. This provides us with considerable freedom w.r.t. a random modification of the represented graph. At first sight, one may even think that one could select any subset of a given family of node sets and fill it, respecting the two conditions, with randomly generated sets to cover all nodes. However, an entirely unrestricted selection is not possible, because a subset of a family of node sets having the running intersection property need not have this property: Consider the family $\mathcal{M} = \{\{A_1, A_2, A_3\}, \{A_2, A_4, A_5\}, \{A_3, A_5, A_6\}, \{A_2, A_3, A_5\}\}$, which has the running intersection property, as can easily be verified with Graham reduction. However, if the last set is removed, this property is lost. Therefore we have to choose the sets to be retained carefully.

Fortunately, there is a very simple selection method, which ensures that all resulting subfamilies have the running intersection property. We shuffle the sets of the given family $\mathcal{M}$ and try to add them in the resulting random order to an initially empty family, observing the two conditions, until a certain percentage of the sets has been added or a certain percentage of the nodes is covered. If a node set cannot be added, it is simply discarded. Clearly, the above theorem ensures that the subfamilies selected in this way have the running intersection property. The resulting family of node sets is then filled, again observing the two conditions, with randomly generated node sets to cover all nodes, which yields a randomly modified graph.

Obviously, this method to modify randomly a given graph with hypertree structure is much less biased than the first method. However, it is not completely unbiased, because the conditions a new set has to satisfy are, in a way, too strong. Situations can arise, in which a set is rejected, although adding it would not destroy the running intersection property. As an example consider the family $\{\{A_1, A_3, A_4\}, \{A_2, A_4, A_5\}\}$ and the new set $\{A_3, A_4, A_5, A_6\}$. Since the nodes A_3, A_4, and A_5 are connected, but not contained in a single set of the family, the new set is rejected. However, if the family were enlarged by this set, it would still have the running intersection property.[2] It is evident, though, that this bias is negligible.

Having constructed a random graph, we must evaluate it. There is a large variety of evaluation measures that may be used [1], [3], but here we only consider the (penalized) log-likelihood of the training data for probabilistic networks, also known as *information criteria* [13], and the (penalized) sum of possibility degrees of the training data for possibilistic networks [2], [3]. Note that in order to compute the value of some evaluation measures it may be necessary or at least convenient to turn the graph into a directed one. Such a transformation can easily achieved by exploiting a construction sequence as found by Graham reduction. Details can be found in [3].

Finally, for a simulated annealing search, we must consider the probability of accepting a solution that is worse than the current one.[3] The problem here is that we usually do not know in advance the maximal quality difference of two graphical models and hence we cannot compute the normalization constant in the exponential distribution $P(\text{accept}) = ce^{-\frac{\Delta Q}{T}}$, which is often used to describe the acceptance probability (ΔQ is the quality difference of the current and the new candidate solution, T is a temperature parameter, which is lowered with time, and c is a normalization constant). To cope with this problem we may use an adaptive approach, which estimates the maximal quality difference from the best and the worst graphical model inspected so far. A simple choice is the unbiased estimator for a uniform distribution [14],

$$\widehat{\Delta Q}_{\max} = \tfrac{n+1}{n} \, |Q_{\text{best}} - Q_{\text{worst}}|,$$

where n is the number of graphical models evaluated so far, although the uniform distribution assumption is, of course, debatable. However, it is not very likely that the exact estimation function used has a strong influence on the results.

With these considerations we eventually have all components needed for a simulated annealing approach to learn a graphical model. However, it is clear that the methods to randomly generate and modify graphs with hypertree structure can also easily be adapted for use in a genetic algorithm.

IV. Experimental Results

We implemented our learning method in a prototypical fashion as part of the INES program (Induction of NEtwork Structures) [2], [3] and tested it on the well-known Danish Jersey cattle blood group determination problem [17]. For this problem there is a Bayesian network designed by human domain experts, which serves the purpose to verify parentage for pedigree registration. The domain modeled in this example is described by 21 attributes, eight of which are observable.

[2] Note that this family can be constructed if the sets are generated in a different order, e.g. if the set $\{A_3, A_4, A_5, A_6\}$ is added first.

[3] Recall that in simulated annealing better solutions are always accepted, while worse ones are accepted only with a probability that depends on how much worse the solution is and that, in addition, decreases with time.

TABLE I
EXPERIMENTAL RESULTS: PROBABILISTIC NETWORKS

measure	edges	params.	train	test
original	22	219.0	−11391.0	−11506.1
indep.	0	59.0	−19921.2	−20087.2
tree	20.0	169.5	−12149.2	−12292.5
K2	23.3	229.9	−11385.4	−11511.5
sim. ann.	28.3	438.1	−13280.2	−13594.9
penalized	27.9	397.6	−13255.7	−13521.8

TABLE II
EXPERIMENTAL RESULTS: POSSIBILISTIC NETWORKS

measure	edges	params.	min.	avg.	max.
original	22.0	308.0	9.888	9.917	11.318
indep.	0.0	80.0	10.064	10.160	11.390
tree	20.0	404.0	8.466	8.598	10.386
greedy	33.0	774.0	8.206	8.344	10.416
sim. ann.	22.6	787.2	8.013	8.291	9.981
penalized	20.6	419.1	8.211	8.488	10.133

For our probabilistic tests, we generated from this network ten pairs of databases with 1000 tuples each. The first database of each pair was used to induce a graphical model, the second to test it. The results were averaged over all ten pairs. As a baseline for comparisons we used the original graph and a graph without any edges, i.e. independent attributes. In addition, we used an optimum spanning tree constructed with mutual information as edge weights [16] and the well-known K2 algorithm [5]. Our simulated annealing method we applied in two version, one using only the log-likelihood of the database and one using also a penalty term computed from the number of parameters of the model. The clique size was restricted to three attributes. Table I shows the results. Unfortunately, the simulated annealing approach fares worse than the alternatives. We assume, however, that better results can be achieved by initializing the search with a good base structure, for instance, with an optimum weight spanning tree.

Besides the domain expert designed reference structure mentioned above there is a database of 500 real world sample cases. We used this database for our possibilistic tests, because it contains a large number of missing values (only a little over half of the tuples are complete) and is thus well suited to show the strengths of a possibilistic approach. As for probabilistic networks we compared the results of our approach with the reference structure (that is, a possibilistic network with the same structure as the human expert designed network and degrees of possibility determined from the database), and a graph without edges. In addition, we used optimum spanning tree construction and greedy parent selection (both employing the evaluation measure that led to the best results [3]). The results are shown in table II, with the last three columns showing the sums of the minimum, average, and maximum degree of possibility over the tuples in the database [2] (incomplete tuples do not have a unique degree of possibility). In contrast to the probabilistic case, our approach clearly led to the best results, even though it was not initialized with a good base structure.

V. SUMMARY

In this paper we suggested a simulated annealing approach to learn graphical models from data. Our algorithm learns graphs with hypertree structure, the complexity of which can be controlled at learning time by restricting the clique size. Hence it is well suited to find good approximations that allow for efficient evidence propagation. The main contribution of this paper is an efficient and almost unbiased method to randomly generate and modify graphs with hypertree structure. Our experiments show that this approach seems to be especially useful for learning possibilistic graphical models.

REFERENCES

[1] C. Borgelt and R. Kruse. Evaluation Measures for Learning Probabilistic and Possibilistic Networks. *Proc. 6th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'97, Barcelona, Spain)*, Vol. 2:1034–1038. IEEE Press, Piscataway, NJ, USA 1997

[2] C. Borgelt and R. Kruse. Some Experimental Results on Learning Probabilistic and Possibilistic Networks with Different Evaluation Measures. *Proc. 1st Int. J. Conf. on Qualitative and Quantitative Practical Reasoning (ECSQARU/FAPR'97, Bad Honnef, Germany)*, LNAI 1244, 71–85. Springer, Berlin, Germany, 1997

[3] C. Borgelt. *Data Mining with Graphical Models.* Ph.D. thesis, University of Magdeburg, Germany 2000

[4] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, NY, USA 1997

[5] G.F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347. Kluwer, Dordrecht, Netherlands 1992

[6] J. Gebhardt and R. Kruse. Learning Possibilistic Networks from Data. *Proc. 5th Int. Workshop on Artificial Intelligence and Statistics (Fort Lauderdale, FL, USA)*, 233–244. Springer, New York, NY, USA 1995

[7] J. Gebhardt. *Learning from Data: Possibilistic Graphical Models.* Habil. thesis, University of Braunschweig, Germany 1997

[8] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20:197–243. Kluwer, Dordrecht, Netherlands 1995

[9] F.V. Jensen. *An Introduction to Bayesian Networks*. UCL Press Ltd./Springer, London, Great Britain 1996

[10] S. Kirkpatrick, C.D. Gelatt, and M.P. Vercchi. Optimization by Simulated Annealing. *Science* 220:671–680. High Wire Press, Stanford, CA, USA 1983

[11] R. Kruse, J. Gebhardt, and F. Klawonn. *Foundations of Fuzzy Systems*, J. Wiley & Sons, Chichester, United Kingdom 1994.

[12] S.L. Lauritzen and D.J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society, Series B*, 2(50):157–224. Blackwell, Oxford, United Kingdom 1988

[13] S.L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, United Kingdom 1996

[14] R.J. Larsen and M.L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Prentice-Hall, Englewood Cliffs, NJ, USA 1986

[15] N. Metropolis, N. Rosenblut, A. Teller, and E. Teller. Equation of State Calculations for Fast Computing Machines. *Journal of Chemical Physics* 21:1087–1092. American Institute of Physics, Melville, NY, USA 1953

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, USA 1988 (2nd edition 1992)

[17] L.K. Rasmussen. *Blood Group Determination of Danish Jersey Cattle in the F-blood Group System (Dina Research Report 8)*. Dina Foulum, Tjele, Denmark 1992

[18] G. Shafer and P.P. Shenoy. *Local Computations in Hypertrees (Working Paper 201)*. School of Business, University of Kansas, Lawrence, KS, USA 1988

[19] R.E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing* 13:566–579. Society of Ind. and Applied Mathematics, Philadelphia, PA, USA 1984

[20] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. J. Wiley & Sons, Chichester, United Kingdom 1990