ELSEVIER Artificial Intelligence 68 (1994) 399–410

Artificial Intelligence

Research Note

Finding MAPs for belief networks is NP-hard

Solomon Eyal Shimony*

Mathematics and Computer Science Department, Ben-Gurion University, P.O. Box 653, 84105 Beer-Sheva, Israel

Received August 1993; revised February 1994

Abstract

Given a probabilistic world model, an important problem is to find the maximum a-posteriori probability (MAP) instantiation of all the random variables given the evidence. Numerous researchers using such models employ some graph representation for the distributions, such as a Bayesian belief network. This representation simplifies the complexity of specifying the distributions from exponential in n, the number of variables in the model, to linear in n, in many interesting cases. We show, however, that finding the MAP is NP-hard in the general case when these representations are used, even if the size of the representation happens to be linear in n. Furthermore, minor modifications to the proof show that the problem remains NP-hard for various restrictions of the topology of the graphs. The same technique can be applied to the results of a related paper (by Cooper), to further restrict belief network topology in the proof that probabilistic inference is NP-hard.

Keywords. Probabilistic reasoning; Explanation; Abductive reasoning; Diagnosis; Complexity

1. Introduction

Graphical representations of statistical or causal dependence frequently constitute an important component when a probabilistic world knowledge is used. Three such representations of interest are Bayesian belief networks [8], Markov random fields [5], and a generalization of the latter, Markov networks [8]. In all these cases, the graphical representation makes the dependencies in our

* E-mail: shimony@bengus.bitnet.

world knowledge explicit, as well as saving many orders of magnitude in the size of the representation. If our model has n random variables, then representing the complete distribution might require space exponential in n. By using the structuring (and independence assumptions) available in the graphical representation, and a very sparse graph, space linear in n is frequently sufficient. In this paper, we focus on Bayesian belief networks (also called causal networks or probabilistic influence diagrams in the literature), and discuss the other models only briefly at the end.

Diagnostic reasoning, also known as abductive reasoning or explanation, is an important problem in artificial intelligence and its applications, such as natural language understanding, medical diagnosis, circuit fault diagnosis, common-sense explanation, and pattern recognition. Abductive reasoning under (probabilistic) uncertainty can be modeled either as finding posterior distribution given some evidence, e.g. [1,8], or as a maximum a-posteriori probability (MAP) instantiation of all the variables in the network given the evidence [5] (also called *most probable explanation* (MPE), [8]), as well as other schemes. Each of these schemes has its merits, discussed elsewhere [9,14]. We are concerned here with the computational complexity of the problems. In the special case of singly connected networks, both problems are known to have polynomial-time algorithms [6,8]. The case of multiply connected networks, however, was suspected to be hard for both problems. Indeed, the former problem, also called "probabilistic inference", was shown to be NP-hard for Bayesian belief networks in [2]. The latter (MAP) problem, also seems to be hard for belief networks, but no such result seems to have been published. In fact, several papers misquote [2] as containing the proof, while the paper addresses only the problem of computing (prior or posterior) distributions.[1] The above proof was later used in [3] to show that even *approximating* probabilistic inference is NP-hard. In [14], a problem very close to MAP called "incomplete assignment cost-based abduction" was shown to be NP-hard, and indeed we use components of that proof here. Since the incomplete assignment cost-based abduction problem is somewhat different from MAP, additional tricks had to be added for the proof to go through.

The fact that the problem is hard suggests that either approximation algorithms or domain-specific algorithms be used for computing MAPs. As it turns out, such algorithms have in fact been used in the literature. For example, a simulated annealing algorithm was used for finding the MAP over a Markov field representing pixel and edge processes for image restoration in [5]. More recently, genetic algorithms have been applied to probabilistic reasoning problems [10]. Other approaches may also prove fruitful, such as converting the

[1] Actually, by reading between the lines of the proof in [2], it is possible to adapt the proof to MAPs with evidence. However, to show that the "empty evidence" MAP problem is NP-hard, the modifications needed are complicated, since we are not allowed to clamp a variable to a certain value, and because finding *any* assignment with non-0 probability can be done in polynomial time.

problem to linear inequalities and obtaining a 0–1 solution using simplex and branch and bound [11,12].

In what follows, we review belief networks and the MAP problem, and proceed to show that it is NP-hard. We extend the results to belief networks with several topological restrictions. Finally, we discuss the MAP problem for the other graph representations, and we show how our proof can be used to make a minor improvement to the results of [2].

2. Bayesian belief networks

A (Bayesian) belief network is a directed acyclic graph (DAG) augmented with conditional probability distributions residing in each node. Nodes stand for random variables, which are used to represent domain ("real world") events. Henceforth, we will use the terms "variable" and "node" interchangeably. In general, random variables may be either continuous or discrete, but in order to prove NP-hardness it is sufficient to assume that the variables are binary (and thus discrete) variables. Let us designate the states of the binary variables by T for "true" and F for "false".

For each node, a probability of each state of the node given each possible state of its parents is given. Let us denote the set of parents of a node v by $\pi(v)$. The conditional probability[2] $P(v \mid \pi(v))$ is given for each possible instantiation of the variables v and $\pi(v)$. We define the probability of an event given the empty set as the prior probability. Thus, for a root node v (i.e. a node with no parents) the probability $P(v \mid \pi(v)) = P(v \mid \emptyset) \equiv P(v)$ is given for each state of v.

We represent a belief network by a triple (V, E, P), where V is the set of nodes in the network (with $n = |V|$), E is the set of directed edges (v, u), and P is a set of conditional probabilities (as defined above), indexed by the nodes of V. A belief network represents a distribution over the sample space consisting of all possible instantiations to all of its random variables, as follows [8]:

$$P(v_1, v_2, \ldots, v_n) = \prod_{i=1}^{n} P(v_i \mid \pi(v_i)). \tag{1}$$

Eq. (1) shows us where the savings in representation size come from. Representing the full distribution of n binary variables directly requires $C_f = 2^n$ probabilities. Using a belief network (with its inherent independence assumptions [8]), we only need:

[2] A variable or set of variables appearing inside a probability term stand for the event where the variables assume every possible instantiation. For example, let v and w be binary nodes. Then $P(v)$ stands for $P(v = T)$ and $P(v = F)$, and $P(v) = P(v \mid w)$ stands for the four equations: $P(v = T) = P(v = T \mid w = T)$, $P(v = T) = P(v = T \mid w = F)$, $P(v = F) = P(v = F \mid w = T)$, and $P(v = F) = P(v = F \mid w = F)$.

$$C_b = \sum_{i=1}^{n} 2^{\pi(v_i)}. \tag{2}$$

In the research community, it is assumed that the maximum in-degree in the graph k is much less than n, or even constant. This provides a considerable savings in space. Additionally, algorithms exist for performing probabilistic inference or MAP computation that do not expand the entire probability space during computation [6–8,13]. However, except for singly-connected graphs, where polynomial-time algorithms for MAP and probabilistic inference exist [8], all existing exact algorithms have an exponential runtime term, based on maximum clique size [7], cut-set size [8], or other factors [13].

3. MAP problems

The MAP problem is defined as follows. Given a probability distribution over a set of variables V, and evidence $\mathcal{E}$, which is an instantiation of a (possibly empty) set of variables $E \subseteq V$, find the instantiation (or value assignment) $\mathcal{A}$ to all the variables V that maximizes $P(\mathcal{A} \mid \mathcal{E})$. (It does not matter whether $\mathcal{A}$ assigns values to all of V or just to $V - E$, because $P(A \cap B \mid B) = P(A \mid B)$ for any events A and B.) We will show that even the restricted form, where the evidence is empty, is NP-hard. Since belief networks represent probability distributions, we have a special case of the MAP problem, which we call MAPBNET. Note that we need to show that the problem is hard for a small maximum in-degree, as otherwise the size of the problem *representation* may be exponential in $|V|$.

In order to prove NP-hardness, we will use a related decision problem, defined as follows: given a belief network, is there an instantiation of variables such that its probability is at least p? We call this problem MAPBNETD. If we have a solution to MAPBNET, we can easily obtain a solution to MAPBNETD: plug the instantiation obtained as the solution to MAPBNET into Eq. (1) (computation time linear in n) and compare the result with p.

4. MAPBNET is NP-hard: proof

To prove that MAPBNETD is NP-complete, it is sufficient to show that MAPBNETD is in NP, and to reduce a problem known to be NP-complete to MAPBNETD. It is possible to use several different such problems. We choose to use the well-known vertex cover (VC) problem [4, p. 190], as this choice allows the NP-completeness results to hold for special cases where the topology of the belief network is severely restricted. Since a solution to MAPBNET allows us to find a solution to MAPBNETD in linear time, this will show that MAPBNET is NP-hard as well.

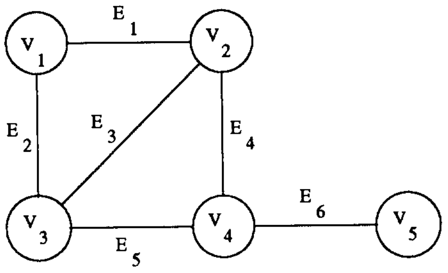

Fig. 1. Vertex cover example graph.

4.1. *Definition of vertex cover*

The vertex cover problem (VC), is defined as follows: given a graph, is there a set of vertices in the graph, with size at most K, that "covers" all the edges of the graph? (That is, such that removing all the vertices in this set, together with their incident edges, leaves us with a graph with no edges.) Formally, the problem is: given a graph $G = (V,E)$, and an integer K, is there a subset $C \subseteq V$ such that $|C| \leqslant K$, and for all edges $e = (v_1,v_2)$ where $e \in E$, at least one of v_1 and v_2 is in C?

For example, consider the graph of Fig. 1, consisting of vertices $V = \{V_1,V_2,V_3,V_4,V_5\}$ and edges $E = \{E_1,E_2,E_3,E_4,E_5,E_6\}$. In this graph there is no vertex cover of size $K = 2$, but there is at least one vertex cover of size $K = 3$, e.g. $\{V_1,V_2,V_4\}$. Thus, the answer to the vertex cover decision problem is "yes" for $K \geqslant 3$ and "no" for $K \leqslant 2$ in this example. We call this VC problem instance VC_{ex}.

4.2. *Transforming VC to MAPBNETD*

We now transform VC to MAPBNETD. Let the vertex cover problem instance be the graph $G = (V',E')$, where we need to decide whether there is a vertex cover of size K or less. Let $n' = |V'|$ and $m' = |E'|$. Assuming that neither the edge set nor the vertex set are empty (in which case the VC problem would have a trivial solution), we construct a belief network (V,E,P) from G. The nodes V of the belief networks are constructed as follows (except for the AND nodes, to be explained later on):

(1) For each vertex in V' and each edge in E' construct a unique binary node in the belief network. For each vertex or edge $x \in V' \cup E'$, let us denote its counterpart node in the belief network by the one-to-one function $B(x)$. By construction, B is invertible, so let B^{-1} denote the inverse of B. If node $B^{-1}(v)$ is a vertex in V', we call it a vertex setting node. Otherwise ($B^{-1}(v)$ is an edge in E'), we call it an edge setting node.

(2) Construct a binary node S, the "evidence" node (it will eventually have to be set to T in any MAP).

(3) Construct $2n'$ binary nodes $D_1,\ldots,D_{2n'}$, the "probability drain" nodes.

The edge set E of the belief network (initially empty) and the conditional distributions P are constructed as follows:

(1) All vertex setting nodes are root nodes (i.e. have no parents). If v is a vertex setting node, then its distribution is $P(v = T) = 0.25$ and thus $P(v = F) = 1 - P(v = T) = 0.75$.

(2) For each edge setting node v, perform the following operations: Let $e = B^{-1}(v)$, the edge in G corresponding to v. Let x_1 and x_2 be the vertices in G incident on e, $u_1 = B(x_1)$, and $u_2 = B(x_2)$. Add to E edges from u_1 to v and from u_2 to v. Set the conditional distribution of v so that v is a pure OR node, i.e.

$$P(v = T \mid u_1, u_2) = \begin{cases} 0, & u_1 = F, u_2 = F, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

(3) Connect the edge setting nodes to the evidence node S, using pure AND nodes where necessary. A pure AND node v with parents $u_1, \ldots, u_k$ has the following distribution:

$$P(v = T \mid u_1, \ldots, u_k) = \begin{cases} 1, & \forall i \;\; k \geqslant i \geqslant 1 \rightarrow u_i = T, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Use the following algorithm for connecting the AND nodes:

(a) Set A to the set of edge setting nodes.

(b) If $|A| \leqslant 2$, then add to E an edge from each of the nodes in A to S, let S be a pure AND node, and stop.

(c) Let A' be a set of $\lfloor \frac{1}{2}|A| \rfloor$ newly constructed pure two-input AND nodes (i.e. construct them now). Let there be some index on the nodes of A and on the nodes of A', i.e. A_i denotes node i of A, starting at 0, and A'_i denotes node i of A'. Partition A into a set of disjoint pairs $a_j = (A_{2j}, A_{2j+1})$ (with one unpaired node v if $|A|$ is odd). For each such pair, a_j, add to E an edge from A_{2j} to A'_j, and an edge from A_{2j+1} to A'_j.

(d) If $|A|$ is even, then set A to A'. Otherwise, set A to $A' \cup \{v\}$.

(e) Go to step 3(b) .

(4) For each probability drain node D_i (with $1 \leqslant i \leqslant 2n'$), add to E an edge from S to D_i. The distribution of each node D_i is

$$P(D_i = T \mid S) = \begin{cases} 1, & S = T, \\ 0.5, & S = F. \end{cases} \tag{5}$$

Let a denote the number of AND nodes. Clearly, $a < m'$. Let AND$_i$ denote the ith AND node (the order is immaterial).

For example, the belief network constructed for the graph in VC_{ex} and Fig. 1, is shown in Fig. 2, where edge and vertex setting nodes retain the same names as their counterparts in the original graph, and AND nodes appear as A_i. The intuition behind the construction is as follows: vertex setting nodes are constructed to be true if the corresponding vertex is in the vertex cover. Edge

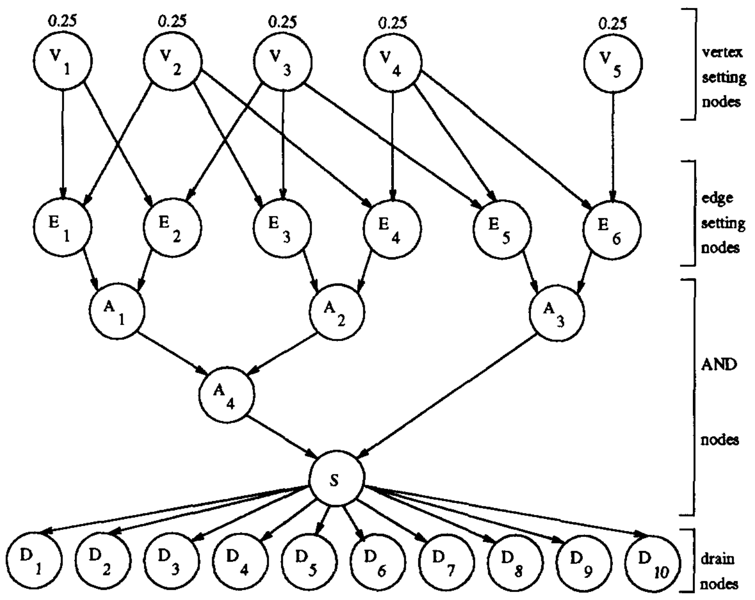

Fig. 2. Belief network for vertex cover example.

setting nodes are true if the respective edge is covered. All edge setting nodes are AND'ed together in S in order to force them to be true, i.e. to force all edges to be covered. Since we are not allowed to introduce evidence (we would want to state $S = T$), we use the probability drain nodes: if S is true, then all drain nodes are true with probability 1. If S is false, however, then each drain node is true or false with equal probability, and thus the probability of each such instantiation is very low, thereby "draining" the probability of any instantiation that has $S = F$. This essentially forces S to be true.

The AND node construction is not really necessary: we could have made S into an m'-input AND node (essentially this is what the AND-node part of the network is doing). In that case, however, specifying the distribution as an array would have taken exponential space and time. Additionally (even if we assumed a linear-space functional distribution representation), this construction allows us to have a belief network constrained to an in-degree of 2.

As to the probability of any complete instantiation, we can rewrite Eq. (1), enumerating the product terms:

$$P(V) = P(S \mid \pi(S)) \prod_{i=1}^{2n'} P(D_i \mid S) \prod_{i=1}^{a} P(\text{AND}_i \mid \pi(\text{AND}_i))$$
$$\prod_{e \in E'} P(B(e) \mid \pi(B(e))) \prod_{x \in V'} P(B(x)). \tag{6}$$

The construction of the network takes time polynomial in the size of G: the number of vertices in the belief network is n (consisting of n' vertex setting nodes, m' edge setting nodes, node S, $2n'$ probability drain nodes, and at most $m' - 1$ AND nodes). Thus, $n \leqslant 3(n' + m')$. Since, by construction, the maximum in-degree is 2, the number of edges in the belief network is at most $2n$. More importantly, the number of entries in the distribution representation is bounded by $2 \times 2^2 \times n = 8n \leqslant 24(n' + m')$, and thus the size of the belief network representation, as well as the time for constructing it, are polynomial (actually linear) in the size of G.

We now show that the vertex cover problem has a vertex cover of size at most K if and only if the constructed belief network has an instantiation $\mathcal{A}$ to all the nodes such that:

$$P(\mathcal{A}) \geqslant p \triangleq \frac{3^{n'-K}}{4^{n'}}. \tag{7}$$

Let C be a vertex cover of size $L \leqslant K$. We construct instantiation $\mathcal{A}$ to the nodes of the belief network as follows: set all edge setting nodes, all AND nodes, node S, and all probability drain nodes to T. For each vertex setting node v, if $B^{-1}(v) \in C$ then $v = T$, otherwise $v = F$.

By construction of $\mathcal{A}$, the first term of Eq. (6), as well as the first and second products of the equation, are 1. The third product is also 1, because for each edge setting node v (nodes appearing as E_i in Fig. 2), at least one of its parents is T, and v is a pure OR node. That occurs because $B^{-1}(v)$ is incident on at least one vertex $x \in C$. Since by construction, $B(x)$ is a parent of v, and it is set to T by the construction of $\mathcal{A}$, we have that v is T with probability 1. We remain with the last product term, which is the only one not equal to 1. In fact, its terms are either 0.25 (for nodes v such that $B^{-1}(v) \in C$) or 0.75 (for all other vertex setting nodes). Thus,

$$P(\mathcal{A}) = \prod_{x \in V'} P(B(x)) = \prod_{x \in C} \frac{1}{4} \prod_{x \in V' - C} \frac{3}{4} = \frac{3^{n'-L}}{4^{n'}}, \tag{8}$$

and, since $L \leqslant K$, we have

$$P(\mathcal{A}) = \frac{3^{n'-L}}{4^{n'}} \geqslant \frac{3^{n'-K}}{4^{n'}} = p.$$

We now show that the existence of an instantiation $\mathcal{A}$ with $P(\mathcal{A}) \geqslant p$ implies the existence of a vertex cover with size at most K. First, we will show that for any $n' \geqslant K \geqslant 1$, an instantiation such that $P(\mathcal{A}) \geqslant p$ must have all the nodes set to T, except for some vertex setting nodes. If $S = F$, then by looking at Eq. (6) we can see that the first product is $2^{-2n'} = 4^{-n'}$, and since all terms in the last product of Eq. (6) are less than 1, and no probability term can be greater than 1, any such instantiation has a probability less than $4^{-n'}$. Thus, clearly $P(\mathcal{A}) < p$ for any K, and to have any way of having a greater probability, S must be instantiated to T. This, in turn, forces $D_i = T$, all the AND nodes to be T, and all edge setting nodes to be T (if any of these nodes are F, some probability product term will be 0, thus $P(\mathcal{A}) = 0$). Since all non-vertex-setting nodes have to be T, all but the last product in Eq. (6), are 1. Also, for each edge setting node v, at least one parent has to be T in any non-0-probability instantiation (as v is a pure OR node). This means that for instantiation $\mathcal{A}$ with $P(\mathcal{A}) \geqslant p$ for any K, the set of vertex setting nodes that are T induce a vertex cover on G: simply let C include all nodes x such that $B(x) = T$ in the instantiation.

Let $\mathcal{A}$ be an instantiation which obeys Eq. (7) for some K. Clearly, in addition to having all the properties shown in the previous paragraph, it must have at most K vertex setting nodes instantiated to T. That is because for such instantiations, if exactly L vertex setting nodes are T, then

$$P(\mathcal{A}) = \frac{3^{n'-L}}{4^{n'}},$$

and thus $P(\mathcal{A}) \geqslant 3^{n'-K}/4^{n'}$ only if $L \leqslant K$. Now, since the instantiation induces a vertex cover of size L (the number of vertex setting nodes instantiated to T), and $L \leqslant K$, we have a vertex cover of size at most K, as required. This completes the proof that MAPBNETD is NP-hard.

To show that MAPBNETD is in NP, note that to compute the probability of an instantiation takes time linear in n (multiplying n terms). Thus, using a nondeterministic machine, guess an instantiation $\mathcal{A}$, compute its probability, and answer "yes" if $P(\mathcal{A}) \geqslant p$. Note that despite the fact that the probabilities are real numbers, computing the probability of the constructed network requires for the computation or for representing the probability to compare with, at most $4n + 1$ bits. That is because all the probabilities are either 0, or 1, or a multiple of $\frac{1}{2}$ ($2n'$ nodes), or a multiple of $\frac{1}{4}$ (n' nodes). Thus, MAPBNETD is NP-complete in the strong sense.

Finally, MAPBNET is the computational version of MAPBNETD, i.e. it comes up with the highest probability complete instantiation. Given this instantiation, it takes time linear in n to find a solution to MAPBNETD: just use Eq. (1), or even its more specific case, Eq. (6), and compare the result with p. Answer "yes" if greater or equal, "no" otherwise. Thus, MAPBNET is NP-hard.

5. Corollaries and other results

We can use the proof of the previous section to obtain complexity results on belief networks with restricted topology. First, note that if we allow functional specification of distributions, we can allow node S to have all edge setting nodes as parents, without making the representation exponential in size. In this case, the depth of the network (longest directed path) is independent of the problem size, and is equal to 4 (i.e. 3 arcs). Thus, the complexity results hold for networks restricted to a depth of 4.

More important results are the bounds on the in-degree and out-degree of the network. In the proof, the maximum in-degree was 2, but the maximum

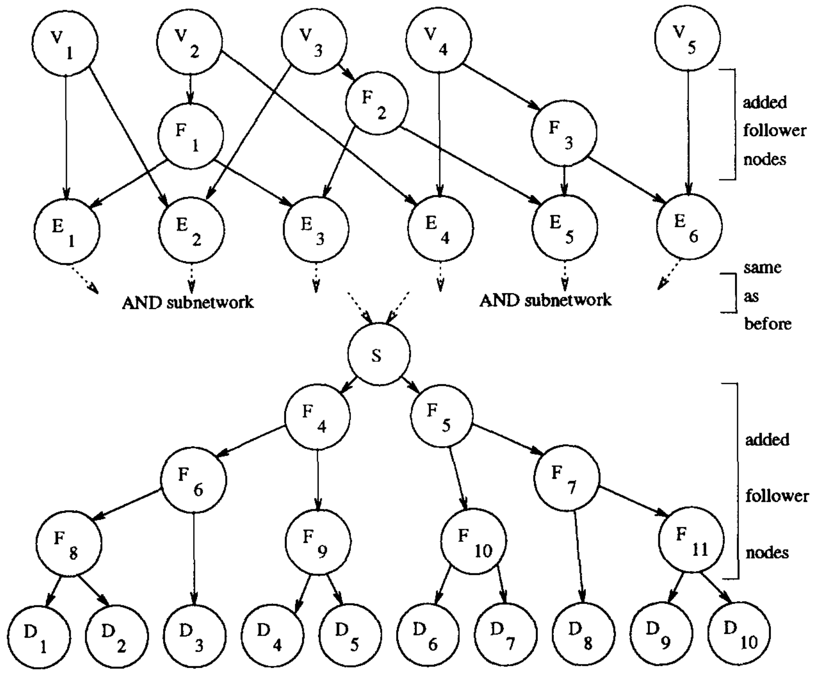

Fig. 3. Expanded out-degree-2 belief network.

out-degree was unbounded. By using a simple trick, adding more nodes that act as "followers", we can limit the out-degree to 2 as well. A follower node v has one parent u, and the following degenerate distribution:

$$P(v = T \mid u) = \begin{cases} 1, & u = T, \\ 0, & u = F. \end{cases} \tag{9}$$

Now, if a node v has more than two outgoing edges, simply attach its outgoing edges to one or two follower nodes, and have the original children of v be children of the follower nodes, for up to 4 original outgoing edges. If a greater number is needed, construct a binary tree of followers of the required depth. Fig. 3 shows one way to convert our example network into a network with a maximum out-degree 2, where only the modified regions are shown. Since these follower nodes do not change the distribution of the network, as long as their value equals their parent node, and cause the probability of any instantiation to drop to 0 if they do not, the NP-completeness proof still goes through. Note that the depth of the network is now logarithmic in n (the number of the follower nodes is less than n, and they only add to the network a depth logarithmic in n).

We thus have the following corollary: MAPBNETD remains NP-complete (and MAPBNET remains NP-hard) even if we restrict the topology of the network so as to obey all of the following constraints:

- Maximum in-degree is 2.

- Maximum out-degree is 2.
- Maximum depth is logarithmic in n.

In addition to these results, the tricks of using gate (e.g. AND or OR) nodes and follower nodes should also apply to the reduction of [2], thus extending the NP-hardness results for the probabilistic inference and for the approximation of probabilistic inference [3] problems to maximum in-degree and out-degree of 2. Logarithmic depth can be achieved by rearranging the part called "overall-satisfaction-testing component" in [2], which is a ladder (*unbalanced* binary tree) of pure AND nodes, into a *balanced* binary tree.

Finally, it should be possible to use a similar proof to show that the MAP problem is NP-hard for Markov networks, the undirected version of belief networks. We can use the same reduction from VC for the proof, by converting the belief network into a Markov network, e.g. as in [7]. In doing the conversion, one must still overcome several minor complications. Thus, the details of the conversion are beyond the scope of this paper (see [15] for details).

Acknowledgements

We wish to thank Eugene Santos Jr for commenting that the NP-hardness result was previously unpublished, and that previous work was frequently misquoted as if containing the proof.

References

[1] E. Charniak and R.P. Goldman, A logic for semantic interpretation, in: *Proceedings 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY (1988).

[2] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, *Artif. Intell.* **42** (2–3) (1990) 393–405.

[3] P. Dagum and M. Luby, Approximating probabilistic inference in Bayesian belief networks is NP-hard, *Artif. Intell.* **60** (1) (1993) 141–153.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness* (Freeman, San Fransisco, CA, 1979).

[5] S. Geeman and D. Geeman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Mach. Intell.* **6** (1984) 721–741.

[6] J.H. Kim and J. Pearl, A computation model for causal and diagnostic reasoning in inference systems, in: *Proceedings IJCAI-83*, Karlsruhe, Germany (1983).

[7] S.L. Lauritzen and D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their applications to expert systems, *J. Roy. Stat. Soc.* **50** (1988) 157–224.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).

[9] D. Poole and G.M. Provan, What is an optimal diagnosis? in: *Proceedings Sixth International Workshop on Uncertainty in AI*, Cambridge, MA (1990) 46–53.

[10] C. Rojas-Guzman and M.A. Kramer, GALGO: a Genetic ALGOrithm decision support tool for complex uncertain systems modeled with Bayesian belief networks, in: *Proceedings Ninth Annual Conference on Uncertainty in AI*, Washington, DC (1993).

[11] E. Santos Jr, On the generation of alternative explanations with implications for belief revision, in: *Proceedings Seventh International Conference on Uncertainty in AI*, Los Angeles, CA (1991) 339–347.

[12] E. Santos Jr, A linear constraint satisfaction approach to cost-based abduction, *Artif. Intell.* **65** (1) (1994) 1–27.
[13] R.D. Shachter, Evaluating influence diagrams, *Oper. Res.* **34** (6) (1986) 871–882.
[14] S.E. Shimony, A probabilistic framework for explanation, Ph.D. Thesis, Tech. Report CS-91-57, Brown University, Providence, RI (1991).
[15] S.E. Shimony, Finding MAP for Markov networks is NP-hard, Tech. Report, Mathematics and Computer Science Department, Ben-Gurion University, Beer-Sheva, Israel (in preparation).