On the Complexity of Belief Network Synthesis and Refinement

Marco Valtorta
University of South Carolina, Columbia, South Carolina
Donald W. Loveland
Duke University, Durham, North Carolina

ABSTRACT

Belief networks are important objects for research study and for actual use, as the experience of the MUNIN project demonstrates. There is evidence that humans are quite good at guessing network structure but poor at settling values for the numerical parameters. Determining these parameters by standard statistical techniques often requires too many sample points (test cases) for larger systems, so knowledge engineers have sought direct algorithms to define or adjust the parameters by appeal to selected test cases. It is shown for both Dempster–Shafer networks and Bayesian networks that for very simple networks (trees), defining parameter values (synthesis) or refining expert-estimated values (refinement) can be computationally intractable. These unpleasant results hold even when we settle for approximate values or demand agreement on only a certain percentage of cases.

KEYWORDS: ***knowledge base refinement, expert systems, knowledge acquisition, belief nets, Dempster–Shafer theory of evidence, Bayesian networks, computational complexity, NP-completeness***

1. INTRODUCTION

Belief networks are gaining popularity as a formalism for implementing knowledge bases for expert systems. With respect to the more common MYCIN-style rule bases, belief networks overcome the problems arising from a truth-functional approach to evidence propagation, by adopting a model-based (or intensional) approach, as explained, for example, by Pearl

Address correspondence to Marco Valtorta, Department of Computer Science, University of South Carolina, Columbia, SC 29208.

Received July 1, 1991; accepted February 26, 1992.

International Journal of Approximate Reasoning 1992; 7:121–148

655 Avenue of the Americas, New York, NY 10010 0888-613X/92/$5.00

[1, Chapter 1]. A disadvantage of belief networks is the computational effort needed to process them.

A key feature of belief networks is their use of numerical parameters. These parameters are probability masses in Dempster–Shafer networks (see Smets [2] for a brief introduction) and conditional probabilities (or related parameters, such as likelihood ratios) in Bayesian networks (see Pearl [1] for the canonical treatment). These numerical parameters can be estimated by using statistical techniques. However, such techniques require a large number of sample points (cases), and developers often resort to knowledge engineering techniques, in the presence of fewer test cases, as documented in the construction of MUNIN.

At present, there are few large applications of belief networks. Of these, probably the best known as MUNIN (Andreassen et al. [3]), a medical expert system for the diagnosis of muscle and nerve diseases from bioelectric signals. As of mid-June 1989, MUNIN had grown to a network of approximately 1000 nodes (Steen Andreassen, personal communication, June 1989),[1] and its developers assessed the system as too large to set the parameters directly from test cases:

> Estimating the 270 conditional probabilities [in part of the MUNIN belief network of 1987]...would require at least 10000 cases [for use of standard statistical techniques]. Instead of relying on this empirical approach, we have tried to rely as much as possible on "deep knowledge," using an understanding of pathophysiological processes as expressed in medical textbooks and papers (Andreassen et al. [3, p. 369]).

However, the designers appeal to test cases to validate the estimates and refine them when necessary. In MUNIN, "Discrepancies between the network and the medical experts lead to revision of the model parameters" (Andreassen et al. [3, p. 370]). The expert here defines test cases, and the designers have the task of refining the parameters to fit the test cases.

There are several points to consider regarding synthesis and refinement of parameter values for diagnostic systems. In addition to the well-documented MUNIN case, there is ample evidence that refinement of parameters is normally performed in the construction of knowledge-based systems that use numerical uncertainty representations. Indeed, parameter refinement capability is needed to achieve diagnostic accuracy, because in many applications test cases reveal inaccuracies in the initial parameter settings that would lead to rejection of the system. However, it is difficult to refine parameters to fit test cases ("memorize"). One result we show is that there is no computationally feasible algorithm (explained below) that uniformly performs the refinement task.

[1] MUNIN has been developed in the context of ESPRIT project 599.

With the increasing interest in belief networks and the consideration of algorithms for them comes the need to understand the computational resources needed. There has already been work showing that important tasks in belief networks are computationally intensive, and these results restrict our vision of what is possible. The primary example so far is the work of Cooper [4], where it is shown that, even provided with the correct local conditional probabilities (direct causal relations), the calculation of the desired output conditional probability is computationally intractable (NP-hard). This occurs in rather simple multiply connected graphs and raises serious questions about freedom to employ apparently commonly occurring graphs. Our paper gives perhaps even more disturbing results in that simple trees can underlie computationally intractable problems.

To define a belief network involves a knowledge acquisition task. To either directly define the network or validate it after some oracle (e.g., expert) has provided the initial specifications, one relies on test cases. In essence, we show here that defining the numerical parameters (masses for Dempster–Shafer nets, conditional probabilities for Bayes nets) for Dempster–Shafer networks and for Bayesian networks of a common form is, in very simple cases, computationally intractable. That is, to define the network that agrees with the test cases can be intractable. Moreover, this intractability remains true even when the expert can offer a good approximation to the parameters. It holds true if we demand agreement with only $k\%$ of the cases offered. As perhaps a final blow, this intractability is still present if we desire only an approximate answer.

This state of affairs has recently been shown to hold for MYCIN-type rule-based systems (Valtorta [5]). Results of the same flavor appear for neural nets. (Further comments are made in Section 9.) However, it has not been known until now how strong the computational intractability results are for belief networks.

This paper addresses the problem of synthesis and refinement of numerical parameters in belief networks from an algorithmic standpoint. Both Dempster–Shafer networks and Bayesian networks are considered. Section 2 formalizes the problem already described in this introduction, by using the notion of case in the Dempster–Shafer framework. Section 3 shows that the synthesis of masses in Dempster–Shafer networks from cases is NP-hard. Section 4 is devoted to the proof that the refinement of masses in Dempster–Shafer networks from cases is NP-hard. Section 5 does for Bayesian network what Section 2 did for Dempster–Shafer networks, and Section 6 is concerned with the synthesis of likelihood ratios in Bayesian networks. This problem is also shown to be NP-hard. Many readers will find Sections 7 and 8 to contain the most interesting results. We study approximations in Section 7. We reconsider the notion of case in Section 8. Informally, a case is an input–output pair that represents a point of the function realized by the net. There are two kinds of cases. Up to Section 8, we view the network as mapping inputs to *beliefs* in particular states. We call these *oracular cases*. In Section 8, we consider a particular class of applications—those in which the task of the expert system that uses the belief network is to classify, such as in diagnostic systems. For these applications, it is more natural to assume that the output part of a case is an indication of class. We call these cases *observed cases*. For example, in a medical diagnosis setting, given a set of symptoms as input, the output part of an oracular case would the beliefs in a list of diagnostic hypotheses, whereas the output part of an observed case is either the actual diagnosis (if there is such a thing) or the most likely diagnosis or a ranking of several of the most likely diagnoses. Section 9 discusses related work, and Section 10 concludes the paper with an assessment of results and the sketch of an alternative refinement model.

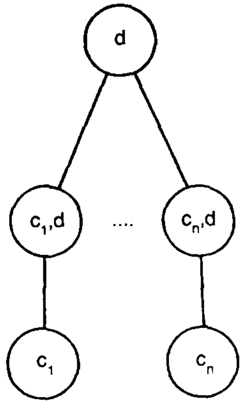
2. FORMALIZING MASS REFINEMENT

Without loss of generality, since we are after lower-bound results, we consider Dempster–Shafer networks (from now on, DS-nets) in the form of a tree (and call them, simply, DS-trees). There are many different versions of DS-trees in the literature, all related in rather straightforward ways. In this paper, alternating Markov trees are used. Markov trees are defined in various places, such as the works of Shafer [6], Shenoy and Shafer [7, 8], and Zarley [9]. Our definition is adapted from Mellouli [10, pp. 66, 85]. A *(qualitative) Markov tree* of variables in a set S is a tree $T = (N, E)$, such that N is a subset of the power set of S (i.e., the nodes of the tree are subsets of S) and such that the intersection of two nodes n_1 and n_2 is contained in node n_0 if n_0 lies between n_1 and n_2 in some branch of T.[2] A Markov tree $T = (N, E)$ is *alternating* if every node in the tree either is contained in all its neighbors or contains all its neighbors. (The reader who is unfamiliar with the definitions may want to verify that the tree in Figure 1 is an alternating Markov tree for variables in the set $S = \{c_1, c_2, \ldots, c_n, d\}$.)

For simplicity (and again, without loss of generality), each of the variables in the DS-tree will be assumed to be two-valued. Call the two values 0 and 1. We will express the mass assigned to a subset of the frame of discernment[3] of variable a as $m(a = v_i)$, where $v_i = 0, 1$, or as $m(l_a)$,

[2] As noted by Shenoy and Shafer [8], Markov trees are called *join trees* in the database literature.

[3] See, for example, Smets [2] or Gordon and Shortliffe [11] for the definition. Informally, the frame of discernment for a set S of variables (where all variables have a discrete range of values) is the Cartesian product of the set of variable-value pairs for all variables in S.



$$m_1(c_1 \Rightarrow d) = s_1, \qquad m_1(\Theta) = 1 - s_1$$

$$m_2(c_2 \Rightarrow d) = s_2, \qquad m_2(\Theta) = 1 - s_2$$

....

$$m_n(c_n \Rightarrow d) = s_n, \qquad m_n(\Theta) = 1 - s_n$$

Figure 1. DS-tree and associated compatibility relations for problem MS.

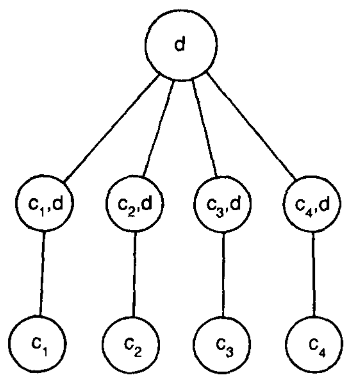
where $l_a = a, \sim a$. The mass assigned to the frame of discernment of variable a will be indicated as $m(\Theta(a))$ or, when there is no ambiguity, simply as $m(\Theta)$. For joint variables, we will indicate a subset of the frame of discernment by a propositional formula. For example, consider the joint variable (a, b) whose frame of discernment is $\Theta((a, b)) = \{\{a = 0, b = 0\}, \{a = 0, b = 1\}, \{a = 1, b = 0\}, \{a = 1, b = 1\}\}$. The mass assigned to the subsets $\{\{a = 0, b = 0\}, \{a = 0, b = 1\}, \{a = 1, b = 1\}\}$ will be indicated as $m(\sim a \vee b)$, or as $m(a \Rightarrow b)$. The mass assigned to the frame of discernment of the joint variable (a, b) will b indicated as $m(\Theta((a, b)))$ or, when there is no ambiguity, simply as $m(\Theta)$. Following Pearl [1, p. 418], we call a subset of the frame of discernment such as $(\sim a \vee b)$ a *compatibility relation*. [Intuitively, $m(\sim a \vee b)$ quantifies the constraint that a is not compatible with $\sim b$.]

A *DS-presentation* is defined as a triple consisting of a DS-tree, a set of compatibility relations, and an assignment of masses to some of the compatibility relations and their frames of discernment. Figure 1 shows a DS-presentation when values $s_1, s_2, \ldots, s_n$ are fixed. DS-presentations will be considered as realizing a function from the vector of masses assigned to the leaf nodes of a DS-tree to a belief (simply a mass for the nets that we consider in this paper) for the root node (via a process involving Dempster's rule and summarized later). A point in the graph of the function will be called a *case*. In other words, a case is an input–output pair that represents the point of the function realized by the net. In a classification system, the input part of the case is an assignment of masses to traits of the item to be classified, and the output part of the case is an assignment of masses to the possible classifications. T_1 through T_4 of Figure 2 are examples of cases. We now describe how this models the situation described by Andreassen et al. [3] and summarized in the introduction. The output part of a case describes the desired "answer" of the DS-tree when "queried" with the evidence encoded as the input part of the same case. Typically, there will be a discrepancy between the value of the belief as computed by the tree and the belief given as the output part of the case. This discrepancy must be eliminated in order for the DS-tree to work correctly.

Before addressing the task of refinement, we address the more basic task of parameter instantiation (assigning values to the parameters). We call this the *synthesis* task.

3. INITIAL MASS SYNTHESIS IS NP-HARD

The problem considered in this section is a problem of synthesis rather than a problem of refinement of masses in DS-presentations.



$$m_1(c_1 \Rightarrow d) = s_1, \qquad m_1(\Theta) = 1 - s_1$$

....

$$m_4(c_4 \Rightarrow d) = s_4, \qquad m_4(\Theta) = 1 - s_4$$

$$T_1\colon ((a, a, a, 0), a)$$

$$T_2\colon ((b, b, b, 0), b)$$

$$T_3\colon ((a, a, 0, a), a)$$

$$T_4\colon ((b, b, 0, b), b)$$

Figure 2. Instance of MS corresponding to $(x_1 \vee x_2 \vee x_3) \& (x_1 \vee x_2 \vee x_4)$.

PROBLEM NAME Mass Synthesis (MS).

PROBLEM INSTANCE A DS-tree and associated compatibility relations as given in Figure 1; a set of cases.

QUESTION Is there an assignment of values to $s_1, \ldots, s_n$, such that the function realized by the DS-tree satisfies the cases?

THEOREM 1 *MS is NP-hard.*

Proof One-in-three satisfiability (OTS) (Garey and Johnson [12, p. 259]) will be transformed into MS. The variant in which no clause in the formula contains a negative literal will be used. The generic OTS instance is a propositional formula in 3-conjunctive normal form, with no negated variables, such as $(x_1 \vee x_2 \vee x_3)$ & $(x_1 \vee x_2 \vee x_4)$. The question is whether there is a model (i.e., a satisfying assignment of true or false to each variable) for the expression such that each clause has exactly one true variable.

Given a formula E in OTS, the following algorithm produces in time polynomial in the size of E an instance of MS such that the question has answer yes if and only if E has a model in which only one variable per clause is true.

Let n be the number of distinct propositional variables in E and m the number of clauses in E. (n and m can be obtained in polynomial time from any reasonable encoding of E.) (Name the variables $x_1, \ldots, x_n$ for convenience.)

The number of leaves in the DS-tree of the corresponding MS-instance is n. The number of cases in the corresponding MS-instance is $2m$.

There are two cases for each clause in E. Let a and b be a pair of numbers such that $0 < a < b < 1$. Let a generic clause contain the variables x_i, x_j, x_k. The input part of the first case for each clause has $m(c_i) = m(c_j) = m(c_k) = a$, and 0 everywhere else. The output part of the first case for each clause is a. To obtain the second case for this clause, substitute b for a.

The reader can easily verify that the algorithm just given runs in time polynomial in the size of E.

As an example, Figure 2 shows the instance of MS corresponding to $E = (x_1 \vee x_2 \vee x_3)$ & $(x_1 \vee x_2 \vee x_4)$. In the figure, T_1 and T_2 correspond to the first clause in E, while T_3 and T_4 correspond to the second clause in E.

Now we prove the following statement. An instance of MS built according to the algorithm just given is a yes-instance if and only if the corresponding instance of OTS is a yes-instance. The following fact is useful.

Let $[p+]$ denote the *probabilistic sum* operator, defined as $a\,[p+]b = a + b - ab$. It is easy to show, on the basis of an observation by Gordon and Shortliffe [11, Section 3.3], that the mass of d, $m(d)$, can be computed as follows[4]:

$$m(d) = m(c_1) * s_1[p+] \cdots [p+]m(c_n) * s_n$$

The details of the computation, which involves projections over different frames of discernment and the use of Dempster's rule, are left to the reader. Here, $m(d)$ is also the Dempster–Shafer *belief* in d, written Bel(d), as defined, for example, by Smets [2].

We first prove the "if part" of the statement. If variable x_i in the model for E is true, set s_i to 1. Otherwise, set it to 0. This ensures that exactly one of the masses corresponding to each case is 1 and the other two are 0. Therefore, the computed Bel is equal to the mass, and each case is satisfied.

The "only if" part is proved now. Assume that we have a yes-instance of MS. It will be shown that in order for an instance of MS to be a yes-instance, it must be that exactly one of the s_i corresponding to each case is 1 and the other two are 0. By assigning true to the variable corresponding to this single s_i, a model for E is obtained that satisfies the "one in three" condition. Consider a generic pair of cases corresponding to a clause in E. We show, by algebraic manipulation, that this pair is satisfied if and only if exactly one of the three s_i corresponding to the cases is 1 and the other two are 0. Call the strengths x, y, and z. The pair of cases is satisfied if and only if the following system has a solution:

$$ax[p+]ay[p+]az = a, \qquad bx[p+]by[p+]bz = b$$

that is, after carrying out the probabilistic sums and dividing each side by a, we have

$$x + y - axy + z - axz - ayz + a^2xyz = 1$$

$$x + y - bxy + z - bxz - byz + b^2xyz = 1$$

If any two of x, y, and z have value 0, the system has a solution if and only if the other variable has value 1.

To show that the system has no solution if only one of the three variables is 0, subtract the second equation from the first side by side, and divide by $(b - a)$:

$$xy + xz + yz = (b + a)xyz$$

This equation has no solution if only one of the three variables is 0.

[4] Each m should have a different subscript, which is dropped here for readability.

The only case left is that in which the three variables are all positive (and, of course, no greater than 1). In this case, each of the products xy, xz, and yz is greater than or equal to xyz:

$$xy + xz + yz > 2xyz > (b + a)xyz$$

and therefore it is impossible that $xy + xz + yz = (b + a)xyz$. ■

It has been shown that MS is NP-hard. It is appropriate to ask if we can bound the computation from above; in particular, is MS in NP? Consider the following argument. We guess the correct value for each mass (if there is one, else guess 0) and compute the mass of d. If the guessed values have short representations, then the computation of $m(d)$ is do-able in polynomial time, and the check against cases is possible in polynomial time. However, we have no assurance that when masses exist for $s_1, \ldots, s_n$, values of sufficiently small representation exist. Without further analytic results, we must allow for the possibility that some s_i has such a long representation that it cannot even be scanned in polynomial time in the length of the problem input. Thus, we can have the best possible upper bound on computation time for $s_1, \ldots, s_n$, given the circumstances, if the sought values are reasonable. This is somewhat of a moot point anyway; the problem would be intractable enough even if it were in NP. A similar argument applies for the other NP-hardness results in this paper.

4. MASS REFINEMENT IS NP-HARD

PROBLEM NAME Mass Refinement, Search Version (MRS).

PROBLEM INSTANCE A DS-presentation with an assignment of values[5] for $s_1, \ldots, s_n$ as in Figure 1; a positive constant e; a set of cases.

QUESTION Find an assignment of values to $s_1, \ldots, s_n$ each of which is at most e away from the given assignment such that the function realized by the DS-tree satisfies the cases.

MRS is NP-hard if the next decision problem is NP-hard.

PROBLEM NAME Mass Refinement (MR).

PROBLEM INSTANCE A DS-presentation with an assignment of values for $s_1, \ldots, s_n$, as in Figure 1; a positive constant e; a set of cases.

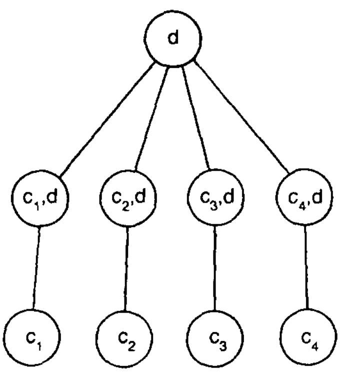
[5] These values may be expert-given or otherwise estimated.

QUESTION Is there an assignment of values to $s_1, \ldots, s_n$ each of which is at most e away from the given assignment such that the function realized by the DS-tree satisfies the cases?

THEOREM 2 *MR is NP-hard for any fixed e.*

Proof OTS (defined in the proof for problem MS in the previous section) will be shown to be reducible to MR. Given a formula E in positive 3-conjunctive normal form, the following algorithm produces in time polynomial in the size of E an instance of MR such that the question has answer yes if and only if E has a model in which only one literal per clause is true. Let $k = e/2$. The algorithm is totally analogous to the one in the proof for problem MR except that the output of the first case in each pair is $a * k$, and the output of the second case in each pair is $b * k$.

As an example, Figure 3 shows the instance of MR corresponding to $E = (x_1 \vee x_2 \vee x_3) \,\&\, (x_1 \vee x_2 \vee x_4)$. In the figure, T_1 and T_2 correspond to the first clause in E, and T_3 and T_4 correspond to the second clause in E.



$m_1(c_1 \Rightarrow d) = s_1, \qquad m_1(\Theta) = 1 - s_1$

....

$m_4(c_4 \Rightarrow d) = s_4, \qquad m_4(\Theta) = 1 - s_4$

Estimates: $s_1 = s_2 = s_3 = s_4 = 0$

T_1: $((a, a, a, 0), ak)$

T_2: $((b, b, b, 0), bk)$

T_3: $((a, a, 0, a), ak)$

T_4: $((b, b, 0, b), bk)$

Figure 3. Instance of MR corresponding to $(x_1 \vee x_2 \vee x_3) \& (x_1 \vee x_2 \vee x_4)$.

An instance of MR built according to the algorithm just given is a yes-instance if and only if the corresponding instance of OTS is a yes-instance.

The "if part" is so similar to that of the proof that MS is NP-hard that it will not be detailed.

The "only if" part is also analogous. The system to be satisfied is

$$ax[p+]ay[p+]az[p+] = ak, \qquad bx[p+]by[p+]bz[p+] = bk$$

that is, after carrying out the probabilistic sums and dividing each side of the equation by a and b respectively, we have

$$x + y + z - axy - axz - ayz + a^2xyz = k$$

$$x + y + z - bxy - bxz - byz + b^2xyz = k$$

If any two of the variables have value 0, the system has a solution if and only if the other has value k.

To show that the system has no solution if only one of the three variables is 0, subtract the second from the first, side by side:

$$xy + xz + yz = (b + a)xyz$$

From this point on, the proof is the same as for MS. ∎

5. FORMALIZING LIKELIHOOD SYNTHESIS

We now consider Bayesian networks, which provide a related but alternative approach to belief networks. The relationship between Dempster–Shafer networks and Bayesian networks is not clear enough for us to translate any intractability results directly from one formulation of belief networks to the other.[6] We do prove similar intractability results for Bayesian networks (from now on, Ba-nets) in this and the following sections but in a somewhat different manner, because we have no counterpart to this section's Lemma 1 that applies to Dempster–Shafer networks. We also suspect that, should the relationship be made entirely precise, the translation of arguments here would be less clear than the direct reproofs.

As in Section 2, we consider only Ba-nets that are trees (Ba-trees), and assume that all variables are two-valued. Call the two values 0 and 1. The event corresponding to variable E being 1 will be denoted as e, and the

[6] Lauritzen and Spiegelhalter [13] show that join trees can be used as a representation for Bayesian networks as well as DS-nets. Shenoy and Shafer [8] show that the rules for traversing a join tree are the same whether one is computing a belief according to Dempster–Shafer theory or according to Pearl. Also see Pearl [1, Chapter 9].

event corresponding to variable E being 0 will be denoted as $\sim e$. The *prior odds* of A are indicated as $O(A)$ and defined as $O(A) = P(A)/P(\sim A)$. The *likelihood* (*ratio*) of $A \mid B$ is indicated as $L(A \mid B)$ and defined as $L(A \mid B) = P(A \mid B)/P(A \mid \sim B)$. The *posterior odds* of $A \mid B$ are indicated as $O(A \mid B)$ and defined as $O(A \mid B) = P(A \mid B)/P(\sim A \mid B)$.

The belief in E, Bel(E), is defined as the (posterior) conditional probability of E given all available evidence. For example, in the network of Figure 4, Bel(H) is the conditional probability of H given all evidence. By "all evidence," we mean the prior probability of H, whether e_i or $\sim e_i$ holds for each i, and the two conditional probabilities $P(E \mid H)$ and $P(E \mid \sim H)$ for each of the links between E_i and H. Alternatively, the evidence can consist of the prior odds of H, whether e_i or $\sim e_i$ holds for each i, and the two likelihoods $L(e \mid H)$ and $L(\sim e \mid H)$ for each of the links between E_i and H. Note that there is a (very) simple procedure to compute prior probabilities from prior odds, conditional probabilities from likelihood ratios, and vice versa. Moreover, there is a (very) simple procedure to compute the posterior odds of H from Bel(H) and vice versa. Therefore, we are at liberty to choose either presentation when we examine the computational complexity of operations in Bayesian networks. We choose the odds-likelihood ratio presentation for simplicity.

To define a case for a Bayesian network, we need to recall a distinction made by Pearl [1] between tangible and intangible (or virtual) evidence. Tangible evidence is direct, categorical evidence for an event. Intangible evidence is a summary (in uncertain terms) of evidence bearing on an event. For example, there could be tangible evidence that an alarm sounds, or there could be an uncertain testimonial of the same event. Pearl argues that the uncertain testimonial (and, generally, intangible evidence) can be summarized using likelihood ratios. He also proposes [1, pp. 151–152] to introduce special events ("dummy nodes") for testimonials and link them to the events of interest (e.g., alarm activations) through a link whose likelihood ratio is the summary of intangible evidence. With the introduction of dummy nodes, intangible evidence is reduced to a special case of

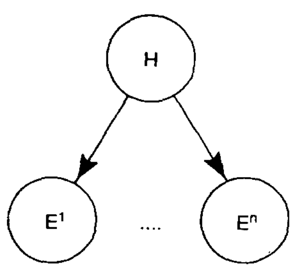


Figure 4. Ba-tree for Example 1.

tangible evidence.[7] Therefore, only tangible evidence is considered in this section.

A *Ba-presentation* is a triple consisting of a Ba-tree, prior odds, and likelihood ratios. Ba-presentations will be considered as realizing a function from a set of outcomes for the events that are the leaves of the Ba-tree to a belief for the root node. A point in the graph of the function is called a *case*. We do not require a case to indicate the value of don't care variables, as illustrated in the following example.

EXAMPLE 1 (Pearl [1])[8] Suppose that a shop has installed a set of n burglar alarms. Each burglar alarm consists of a different sensor device (e.g., photocell, air pressure sensor) that produces a distinct alarm signal. Let H stand for the event that a burglary takes place and E^i stand for the event that the ith alarm goes off, so that e^i indicates that the ith alarm goes off and $\sim e^i$ indicates that the ith alarm does not go off. This setup can be represented by the simple Ba-tree in Figure 4. (Recall that Bayesian networks are directed graphs and the direction of the edges is determined by causality.[9]) The reliability and sensitivity of alarm k are characterized by the conditional probabilities $P(e^k \mid H)$ and $P(e^k \mid \sim H)$ or, more succinctly, by $L(e^k \mid H) = P(e^k \mid H)/P(e^k \mid \sim H)$. (Reliability and sensitivity are the terms used by Pearl. Related terms used in the medical domain are true positive rate, selectivity, and false positive rate.)

A case for this example consists of an indication of active and inactive alarms and of the corresponding belief in a burglary taking place. For instance, suppose that the belief in a burglary taking place when the first alarm goes off but the second does not is 0. The corresponding case is represented as $((e^1, \sim e^2), 0)$. Note that, as the belief in a burglary depends only on the state of alarms 1 and 2, we are not required to specify the state of all other ("don't care") alarms. The specification of the state of all alarms in this example would require an unreasonably large number of data, even for a moderately small number of alarms, because each don't care alarm can be either on or off in a case.

The situation described in the introduction has been modeled as follows. The output part of a case describes the desired answer of the Ba-tree when queried with the evidence encoded as the input part of the case. Typically,

[7] This trick works in the Dempster–Shafer case too, as shown (with some technical restrictions) by Kyburg [14, Section 7], and, of course, in MYCIN-style rule bases.

[8] This example is adapted from Section 2.1.3 of Pearl [1].

[9] We do not take a position here for or against the "true" causal nature of the links. As in [1], the term "causality" is used here in a very broad sense.

there will be a discrepancy between the value of the belief computed by the tree and the belief given as the output part of the case. This discrepancy must be eliminated in order for the Ba-tree to work correctly. We consider a different definition of case in Section 8.

6. LIKELIHOOD SYNTHESIS IS NP-HARD

With reference to the situation described in Figure 4, Pearl [1, Section 2.1.3], exploiting the conditional independence relation expressed by the BA-tree, shows that

$$O(H \mid \underline{E}) = O(H \mid E^1, \ldots, E^n) = O(H) \Pi L(E^k \mid H),$$

where E^i is either e^i or $\sim e^i$. For notational convenience, define $L_1^i = L(e^i \mid H)$ (the ith *positive likelihood*) and $L_0^i = L(\sim e^i \mid H)$ (the ith *negative likelihood*).

Recall that $O(H \mid \underline{E})$ is the output of the Ba-presentation and is easily converted to belief.

The properties that cause the synthesis and refinement of likelihoods to be NP-hard are related to the fact that positive and negative likelihoods cannot be set independently. (Ironically, builders of expert systems have criticized this lack of independence as a requirement that "does violence to intuition" (Duda et al. [15], p. 1077]). This situation is summarized in the lemma that follows. We use only the first property in most proofs. The other property is used for problems LSO and LRO, in Section 9.

LEMMA 1

(i) *If* $L_0^i = 0$, *then* $L_1^i > 1$. *If* $L_1^i = 0$, *then* $L_0^i > 1$.

(ii) *If* L_0^i *and* L_1^i *are both nonzero, either* (1) *they are both equal to* 1, *or* (2) *one is greater than* 1 *and the other is less than* 1.

Proof Property 1 is proved first. The proof consists of the application of the definition of likelihood ratio.

If $L_0^i = 0$, then $P(\sim e^i \mid H) = 0$, because the numerator of L_0^i is $P(\sim e^i \mid H)$. But then $P(e^i \mid H) = 1$, and since this is the numerator of $L_1^i, L_1^i > 1$.

Similarly, if $L_1^i = 0$, then $P(e^i \mid H) = 0$, because the numerator of L_0^i is $P(e^i \mid H)$. But then $P(\sim e^i \mid H) = 1$, and since this is the numerator of L_0^i, $L_0^i > 1$.

A proof of property 2 is given by Duda et al. [15, p. 1077] and is not repeated. (See also Tanimoto [16, Section 7.4.1] and O'Leary [17].) ∎

PROBLEM NAME Likelihood Synthesis (LS).

PROBLEM INSTANCE A Ba-tree as given in Figure 5 with the associated prior odds; a set of cases.

QUESTION Is there an assignment of likelihood ratios to the links of the Ba-tree such that the function realized by the Ba-presentation satisfies the cases?

THEOREM 3 *LS is NP-hard.*

Proof 3-Conjunctive normal form satisfiability (3SAT) (Garey and Johnson [12, p. 259]) will be transformed to LS.

Given a formula F in 3-conjunctive normal form, the following algorithm produces in time polynomial in the size of F an instance of LS such that the question has answer yes if and only if F has a model. Let n be the number of distinct propositional variables in F, and m the number of clauses in F. (Name the variables $x_1, \dots, x_n$ for convenience.)

The number of leaves in the Ba-tree of the corresponding LS instance is n. The number of cases in the corresponding LS instance is m. The prior odds are any positive constant ($o > 0$).

A case is built for each clause $c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ as follows. Each case has output $O(H \mid \underline{E}) = 0$. The input is built as follows:

$$\text{If } l_{ij} = x_{ij} \text{ (i.e., } l_{ij} \text{ is a positive literal), then } \sim e_{ij}.$$

$$\text{If } l_{ij} = \sim x_{ij} \text{ (i.e., } l_{ij} \text{ is a negative literal), then } e_{ij}.$$

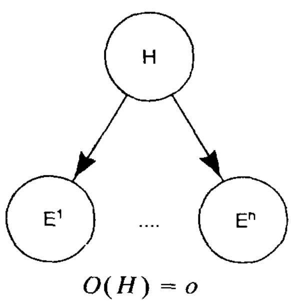


Figure 5. Ba-tree and associated prior odds for problem LS.

As an example, Figure 6 shows the instance of LS corresponding to $F = (\sim x_1 \vee \sim x_2 \vee x_3) \mathbin{\&} (x_1 \vee \sim x_2 \vee \sim x_4)$. Case T_1 corresponds to the first clause in F, and case T_2 corresponds to the second clause in F.[10]

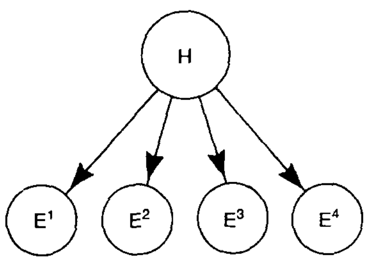
An instance of LS built according to the algorithm just given a yes-instance if and only if the corresponding instance of 3SAT is a yes-instance.

The "if part" is proved first. Let k be a positive constant, $k > 1$. Consider the generic variable x_i that is false in the model. Set $L_1^i = 0$ and $L_0^i = k$. Consider the generic variable x_i that is true in the model. Set $L_1^i = k$ and $L_0^i = 0$. If E has a model, then each clause of E is true in the model. Consider case T_i corresponding to clause c_i with literals l_{i1}, l_{i2}, l_{i3}. In order for the clause to be true, at least one of the literals must be true. Say that the literal is l_{ij}. There are two possible situations:

1. l_{ij} is a positive literal. Then x_{ij} is true in the model. Then L_0^{ij} is 0 in the corresponding instance of LS. Test T_i is satisfied.
2. l_{ij} is a negative literal. Then x_{ij} is false in the model. Then L_1^{ij} is 0 in the corresponding instance of LS. Test T_i is satisfied.

Since all clauses are true, all tests are satisfied, and therefore the LS instance is a yes-instance.

The "only if" part is proved now. Assume that LS is a yes-instance and therefore has a satisfying assignment of values to the likelihoods for each



$O(H) = o$, where o is positive

T_1: $((e^1, e^2, \sim e^3), 0)$

T_2: $((\sim e^1, e^2, e^4), 0)$

Figure 6. Instance of LS corresponding to $F = (\sim x_1 \vee \sim x_2 \vee x_3) \mathbin{\&} (x_1 \vee \sim x_2 \vee \sim x_4)$.

[10] Note that, as discussed at the end of Example 1, in case 1, $O(H \mid \underline{E})$ is the product of the likelihood of E^4 being either e^4 or $\sim e^4$ times $O(H) * L_1^1 * L_1^2 * L_0^3$. However, since the probability that E^4 is either e^4 or $\sim e^4$ is 1, we can just write $O(H \mid \underline{E}) = O(H) * L_1^1 * L_1^2 * L_0^3$. Clearly, this argument applies to all cases considered in this proof and in other proofs of this paper.

case. Construct an interpretation for F as follows. Consider case T_i. In order for T_i to be satisfied, at least one of the corresponding likelihoods must have value 0. There are two possibilities.

1. Assume that this is L_1^{ij}. Then assign false to x_{ij}.
2. Assume that this is L_0^{ij}. Then assign true to x_{ij}.

Lemma 1 guarantees that it is impossible for any pair of complementary likelihoods L_1^k and L_0^k to both be 0. However, there may be some complementary likelihoods L_1^k and L_0^k that are both nonzero in the solution of the yes-instance of LS. In this case, assign true or false, arbitrarily, to variable x_k, because it must be that one of the other likelihoods corresponding to T_i is 0. Note that the assignment of true and false just defined is an interpretation, because it is impossible for a variable in F to be assigned both true and false at the same time.

We now show that the interpretation just defined is a model and therefore the 3SAT instance is a yes-instance. Consider each clause c_i of F in isolation. The corresponding case T_i is satisfied because LS is a yes-instance. In order for the test to be satisfied, one of the corresponding likelihoods must be 0. First, assume that this likelihood is L_1^{ij}. By construction of T_i from c_i, c_i is true if x_{ij} is false. But this is exactly what has been assigned to x_{ij} in the model of the 3SAT instance, according to the rules stated in the previous paragraphs. The case in which the likelihood is L_0^{ij} is analogous and will not be shown. Since each of the clauses of F is true in the interpretation, F is true in the interpretation, and therefore the interpretation is a model for F. ■

Like all NP-hardness results, Theorem 3 does not exclude the possibility that many or even most of the problem instances in LS are readily computable. To show that the complexity of a class of problems is high is to show that some bad members exist; this usually happens at "extreme" points. (As an analogy, consider that the undecidability of the predicate calculus does not say that you can't establish the validity of most formulas of real interest.) In our proof, we use problem instances in which likelihoods are zero. This does not seem unnatural, because the net structure is extremely simple. In fact, the structure is so simple that the class would likely not be suspect until after Theorem 3 is proved. We also emphasize that in theorems proved in subsequent sections of this paper we study problems with restrictions on allowable instances. For example, we do not need any likelihoods to be 0 in the proofs concerning synthesis and refinement in classification (LSO and LRO, Section 8). The more theoretically oriented reader may be able to use the techniques in our proofs to show results for problems with different, but similar, restrictions. Also, we study refinement, approximate solutions, and percentage error in later sections.

The problem of synthesis of likelihoods *and* prior odds will also be shown to be NP-hard.

PROBLEM NAME Likelihood and Prior Odds Synthesis (LPS).

PROBLEM INSTANCE A Ba-tree as given in Figure 4; a set of cases.

QUESTION Is there an assignment of likelihood ratios to the links of the Ba-tree and of odds to node H such that the function realized by the Ba-presentation satisfies the cases?

THEOREM 4 *LPS is NP-hard.*

Proof

Proof (sketch) Adapt the proof that LS is NP-hard as follows:

1. The tree of the instance of LS corresponding to the generic instance of 3SAT has $n + 1$ leaves (instead of n).
2. Cases are built as before, ignoring the $(n + 1)$th leaf, except for the additional case $((e^{n+1}), p)$, where p is any positive constant.

The new case ensures that in any solution to an LPS instance corresponding to a 3SAT instance $O(H) = o$, where o is a positive constant. Therefore, the modified proof that LS is NP-hard is a proof that LPS is NP-hard. ■

7. APPROXIMATIONS

This section presents two problems that arise from attempts to define meaningful approximate solutions to the refinement of numerical parameters. Both problems deal with DS-trees. The two corresponding problems for Ba-trees are also NP-hard.

PROBLEM NAME Approximate Mass Synthesis (AMS).

PROBLEM INSTANCE As for problem MS in Section 3.

QUESTION Find an assignment to $s_1, \ldots, s_n$ strictly within 0.5 of the correct assignment, where the correct assignment is the assignment such that the function realized by the DS-presentation satisfies the cases.

THEOREM 5 *AMS is NP-hard.*

Proof (sketch) Consider a variant of problem MS with the possible values of $s_1, \ldots, s_n$ restricted to the set $\{0, 1\}$. It is easy to show, by adapting the proof of Theorem 1, that the variant is still NP-hard.

If AMS were tractable, problem MS with instances having masses restricted to $\{0, 1\}$ would be tractable. But this contradicts the variant of Theorem 1 just mentioned. ■

PROBLEM NAME Noisy Mass Refinement (NMR).

PROBLEM INSTANCE A DS-tree and associated compatibility relations as given in Figure 1; an assignment of values for $s_1, \ldots, s_n$; a positive constant e; a positive constant k less than 100; a set of cases.

QUESTION Is there an assignment of values to $s_1, \ldots, s_n$ each of which is at most e away from the given ones such that the function satisfies $k\%$ of the cases?

THEOREM 6 *NMR is NP-hard.*

Proof (sketch) Consider a generic MR instance with $m - mk/100$ cases (for arbitrary m).[11] Consider the instance of NMR with m cases built as follows. The NMR instance is identical to the MR instance except that it has m cases. $m - mk/100$ of the cases are as for the MR instance, while each of the remaining $mk/100$ cases is a copy of the first case of the MR instance. By this reduction, if NMR were tractable, MR would be tractable. But this contradicts Theorem 2. ■

8. CLASSIFIERS

In applications in which the *task* (as defined, e.g., by Breuker et al. [18]; cf. Karbach et al. [19]) of the expert system using the belief network is to classify, the user is concerned with the relative ranking (rather than the exact values) of beliefs associated with the terminal nodes[12] of a DS-net or a Ba-net. Cooper [4, Section 5.4 and 5.5] and Valtorta [5] provide additional motivation and techniques.

The corresponding refinement problems are NP-hard. We present the problem for Ba-nets; it is easy to show that the analogous problems for DS-nets are also NP-hard.

In the first problem, we redefine the function computed by the network as follows. The Ba-net *classifies* the input part of a case to be of *class 1* if the computed odds of the net is less than or equal to 1. The Ba-net classifies the input part of a case to be of *class 2* if the computed belief in the output node is greater than 1. Correspondingly, we change the defini-

[11] Round $mk/100$ to the nearest integer.

[12] When suitably defined, in the obvious way.

tion of the output part of a case to be the specification of a class, that is, either "class 1" or "class 2." Note that the problem given below is stated identically to problem LS. The problems differ only because they use a different definition of case.

PROBLEM NAME Likelihood Synthesis, Ordered Output (LSO).

PROBLEM INSTANCE A Ba-tree as given in Figure 5 with the associated prior odds, a set of cases.

QUESTION Is there an assignment of likelihood ratios to the links of the Ba-tree such that the function realized by the Ba-presentation satisfies the cases?

THEOREM 7 *LSO is NP-hard.*

This result is proved by the same technique as for problem LRO, which follows.

PROBLEM NAME Likelihood Refinement, Ordered Output (LRO).

PROBLEM INSTANCE A Ba-tree as given in Figure 5 with the associated prior odds; an assignment of values for $L_0^1, \dots, L_1^n$; a positive constant e; a set of cases.

QUESTION Is there an assignment of values to $L_0^1, \dots, L_1^n$ each of which is at most e away from the given ones such that the function realized by the Ba-presentation satisfies the cases?

THEOREM 8 *LRO is NP-hard.*

Proof Monotone 3-conjunctive normal form satisfiability (MSAT) (Garey and Johnson [12, p. 259]) will be transformed to LRO. The generic MSAT instance is a formula in conjunctive normal form, where each clause contains only three positive (*positive clause*), or three negated (*negative clause*) variables [e.g., $(x_1 \vee x_2 \vee x_3)$ & $(\sim x_1 \vee x \sim x_2 \vee \sim x_2)$]. The question is whether there is a model for the formula.

Given an MSAT formula F, the following algorithm produces in time polynomial in the size of F an instance of LRO such that the question has answer yes if and only if F has a model. Let n be the number of distinct propositional variables in F and m the number of clauses in F. (Name the variables $x_1, \dots, x_n$ for convenience.)

The number of leaves in the Ba-tree of the corresponding LRO instance is n. The number of cases is m. The prior odds are set to 1. The given assignment for each of the positive likelihood ratios is $1 + e/2$. The given assignment for each of the negative likelihood ratios is $1 - e/2$.

The following case (a *positive* case) is built for each positive clause $(x_i \vee x_j \vee x_k)$: $((e^i, e^j, e^k)$, class 2)

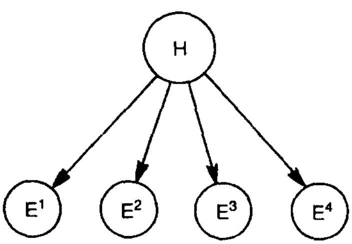
Similarly, the following case (a *negative* case) is built for each negative clause $(\sim x_i \vee \sim x_j \vee \sim x_k)$: $((\sim e^i, \sim e^j, \sim e^k)$, class 2).

As an example, Figure 7 shows part of the LRO instance corresponding to $F = (x_1 \vee x_2 \vee x_3)$ & $(\sim x_1 \vee \sim x_2 \vee \sim x_4)$. Case T_1 corresponds to the first clause in F, and case T_2 corresponds to the second clause in F. The assignment of values to the positive and negative likelihood ratios is not given in Figure 7.

An instance of LRO built according to the algorithm just given is a yes-instance if and only if the corresponding instance of MSAT is a yes-instance.

The "if part" is proved first. Let $g = e/2$. Let $h = 1 + g$ and $k = 1 - g/8$. Note that $h > 1, 0 < k < 1$, and $hkk > 1$ and that h and k are within e of the values $1 + e/2$ and $1 - e/2$, the given assignments of the problem specification. Assume that the MSAT instance is a yes-instance. Since the MSAT instance is a yes-instance, it has a model. Consider the generic variable x_i that is true in the model. Set $L_1^i = h$ and $L_0^i = k$. Similarly, consider the generic variable x_i that is false in the model. Set $L_0^i = h$ and $L_1^i = k$. These assignments are consistent with the definition of positive and negative likelihood and with the restriction that likelihood ratios be within e of the ones originally assigned.

Consider the generic clause, which is true in the model. The corresponding test is satisfied by the setting of likelihoods just given, because the computed output is equal to either *ohhh* or *ohhk* or *ohkk*, and each of these terms is greater than 1, by definition of h and k. Therefore, all the tests are satisfied, and the LRO instance is a yes-instance.



$O(H) = 1$

T_1: $((e^1, e^2, e^3)$, class 2)

T_2: $((\sim e^1, \sim e^2, \sim e^4)$, class 2)

Figure 7. Part of the instance of LRO corresponding to $F = (x_1 \vee x_2 \vee x_3)$ & $(\sim x_1 \vee \sim x_2 \vee \sim x_4)$.

The "only if" part is proved now. Assume that the LRO instance is a yes-instance. Therefore, each of the tests is satisfied. Build an interpretation for the corresonding MSAT instance as follows:

Set x_i to true if both $L_1^i > 1$ and $0 < L_0^i < 1$.

Set x_i to false if both $L_0^i > 1$ and $0 < L_1^i < 1$.

Set x_i to true if both $L_0^i = 1$ and $L_1^i = 1$.

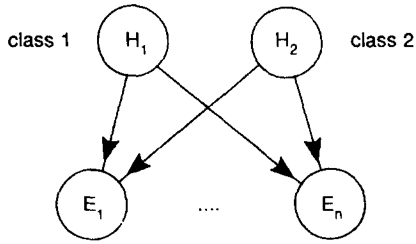
Because all the cases built by the transformation algorithm have output "class 2," we can ignore the situation in which the ith positive or negative likelihood is 0. Therefore, by Lemma 1, the three cases considered just above are exhaustive, and an interpretation has indeed been built. It will next be shown that the interpretation satisfies the MSAT instance and is therefore a model for it.

Consider the generic positive case $((e^i, e^j, e^k)$, class 2). In order for it to be satisfied, it must be that at least one of the ith, jth, and kth positive likelihoods is greater than 1. But in this situation the interpretation described in the previous paragraph has at least one of x_i, x_j, x_k set to true. Therefore, the positive clause corresponding to the generic positive test is true.

Similarly, consider the generic negative case $((\sim e^i, \sim e^j, \sim e^k)$, class 2). In order for it to be satisfied, it must be that at least one of the ith, jth, and kth negative likelihoods is greater than 1. But in this situation the interpretation described in the previous paragraph has at least one of x_i, x_j, x_k set to false. Therefore, the negative clause corresponding to the generic negative test is true.

Finally, since there exists a one-to-one correspondence between all cases in the LRO instance and corresponding clauses in the MSAT instance, and all of the cases are satisfied, then all of the clauses are true and the MSAT instance is a yes-instance. ■

For the next problem, the output part of a case is again an indication of a class. However, now we have a net architecture different from a tree; we consider a multiple-output net. In particular, in place of a root node, we may have many output or *terminal* nodes. Terminal nodes are labeled by class number. We say that the Ba-net *classifies* a case input as being of class i if the computed belief for the terminal node labeled i is at least as high as the belief for the other terminal nodes. We consider explicitly this different network, because there is much interest in applying belief networks to situations in which a decision among multiple competing hypotheses must be made. The fact that we need only slightly modify a previous proof to obtain the following result helps show the robustness of this class of results.



$$O(H_1) = o$$

$$O(H_2) = p$$

Figure 8. Ba-net and associated prior odds for problem LSM.

PROBLEM NAME Likelihood Synthesis, Multiple Outputs (LSM).

PROBLEM INSTANCE A Ba-net as given in Figure 8, with the associated prior odds, and labels on the terminal nodes; a set of cases.

QUESTION Is there an assignment of likelihood ratios to the links of the Ba-net such that the Ba-net correctly classifies all cases?

THEOREM 9 *LSM is NP-hard.*

Proof (sketch) Change the proof that LS is NP-hard as follows:

1. Change the output of each case to 2.

2. Set $p = 0$.

Observe that $O(H_2 \mid \underline{E}) = 0$, because $O(H_2) = p$ and p has been set to 0. Further observe that each of the other new cases is satisfied if and only if the belief in the node labeled 1 is 0 and therefore if and only if each of the cases in the proof that LS is NP-hard is satisfied. Therefore, a proof that LSM is NP-hard has been obtained. ■

9. RELATED WORK

The main reference on the algorithmic complexity of computations in belief networks is Cooper's work [4]. The relationship between that paper and ours is discussed in Sections 1 and 10. Other work on the complexity of computation in belief networks includes that of Orponen [20] and Provan [21].

Belief neworks are a kind of knowledge base. There is a substantial body of literature dealing with the refinement of (truth-functional) rule bases, another kind of knowledge base that is historically prevalent in expert systems. The Sixth International Workshop no Machine Learning included a track on knowledge base refinement and theory revision. See the proceedings for contributions (Segre [22]). In particular, it has been shown that automatic refinement of rule bases from cases is NP-hard (Valtorta [5]).

Synthesis of numerical parameters in rule bases is somewhat analogous to training in neural networks. Indeed, there are apparently similar results of intractability (see Judd [23, 24], Blum and Rivest [25], and Lin and Vitter [26]). However, the functions used in neural networks to process weights are different from those used in rule bases or belief networks (see Fu [27], Valtorta [5]). Laskey [28] proposes to use a kind of neural network (the Boltzmann machine) to "adjust conditional probabilities on the links of a Bayes network."

One would expect parameter refinement to fit cases ("memorization") to be related to and at least as hard as refining parameters to deal with new situations ("generalization"). Chapter 7 of Judd's monograph on the complexity of neural network design [29] is an excellent explanation of the "memorization" versus "generalization" issue. Much of that explanation is applicable to rule bases and belief networks and explains better than we could in a limited space why we do not address generalization directly in this paper.

An alternative to direct refinement of numerical parameters in Ba-nets is the introduction of "dummy nodes" representing unknown events whose only purpose is to account for the failure of a Ba-net with given numerical parameters.[13] (For example, the failure of the Ba-net in Figure 4 to compute the correct belief in H could be attributed to the influence of an unknown event linked to H rather than to errors in the likelihood ratios in the network. The numerical parameters related to the unknown event could be computed using cases.) It is conjectured that this kind of refinement with limited changes in the structure of the belief network is also NP-hard, an interesting conjecture that we are not prepared to address in this paper. The arguments in this paper are based on fixed network structure.

10. CONCLUSION

A major technical contribution of this paper is the proof that refinement of (Dempster–Shafer and Bayesian) belief networks from cases is NP-hard. Two points regarding this result deserve amplification for their potential

[13] Judea Pearl showed one of us this technique and noted its apparent complexity in August 1989. Spiegelhalter and Lauritzen consider a related technique in a recent manuscript.

impact on the practice of expert system construction and will be addressed now.

The networks used in the problem instance of MS, MR, and LS (and, in fact, of all the other problems with the exception of LSM) are trees. They are much simpler than the nets used by Cooper [4] in proving that the computation of beliefs in belief networks is NP-hard. Even more strongly, it is well known (e.g., Kong [30], Pearl [1], Lauritzen and Spiegelhalter [13], Shenoy and Shafer [8]) that the computation of beliefs in trees[14] is tractable. Therefore, a developer could be faced with the unpleasant situation in which the belief network is nicely structured for efficient *computation* of beliefs, but acquisition of the parameters (e.g., likelihood ratios, masses) for the calculation is extremely difficult, and this holds even if good estimates of the parameters are available.

Synthesis of numerical parameters in knowledge bases, be they rule bases or belief networks, is a kind of refinement of knowledge bases: the numerical estimates of the parameters are not available although the structure of the knowledge base is. This situation is particularly interesting. The structure of the knowledge base can be determined by answering relatively simple questions about independence of events. Therefore, the knowledge engineer should believe more strongly in the (qualitative) network structure than in the values of the numerical parameters,[15] and it is natural to use an expert to obtain the knowledge structure and initial guesses. These considerations suggest a methodology for the construction of knowledge-based systems that use belief networks. At the heart of this methodology is a *propose-and-fit* cycle. In this cycle, after interviewing an expert, the designer proposes a structure for the belief network. The network parameters are set or adjusted to fit selected test cases. If parameters can be set to fit the cases, the development is complete. If they cannot, the designer must consult the expert further until a new (qualitative) network structure is proposed. Another attempt is made to set the parameters to fit the cases, and so on. Our results indicate that it is difficult to automate the "fit" step of this methodology.

In this paper, we have not assumed that an expert who can give the correct probability for any set of (input) evidence is on call. Instead, we have assumed (more realistically, according to MUNIN designers[16]) that

---

[14] The same is true for some kinds of graphs that are not trees.

[15] This is related to the argument (Wilkins and Buchanan [31], Valtorta [32], Andreassen et al. [3], Lauritzen and Spiegelhalter [13]) that the structure of the knowledge base should be changed only if the numerical parameters cannot be set in such a way that the knowledge base performs correctly.

[16] Stig Andersen, personal communication, November 1989.

an arbitrary collection of cases is determined a priori. These may come from an expert, a textbook, a database, or some other source. As an example of alternative methods for refinement, consider the use of a type of oracle. If an expert is available after construction of the knowledge base, the expert could be used as an oracle to facilitate knowledge base refinement. In this mode, the expert would be queried with specially focused questions allowing the synthesis or refinement of specific masses or likelihoods. A result in the setting in which the oracle is on call concerning rule bases (Valtorta [32, Chapters 4 and 7; 33]) indicates that automatic synthesis or refinement in certain belief networks that are trees is do-able in polynomial time and suggests that it is intractable for graphs. More remains to be done along this line of work.

ACKNOWLEDGMENTS

M.V. thanks several members of the MUNIN team, especially Stig Andersen and Steen Andreassen, for several useful conversations concerning MUNIN and, more generally, the design, construction, and maintenance of Bayesian networks. Research was supported by a University of South Carolina summer grant to M.V. and by grant AFOSR-83-8205, with D.W.L. as principal investigator.

References

1. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan-Kaufmann, San Mateo, Calif., 1988.

2. Smets, P., Belief functions, in *Non-Standard Logics for Automated Reasoning* (P. Smets, E. H. Mamdani, D. Dubois, and H. Prade, Eds.), Academic, London, 1988.

3. Andreassen, S., Woldbye, M., Falck, B., and Andersen, S. K., MUNIN—a causal probabilistic network for interpretation of electromyographic findings, *Proceedings of the Tenth International Joint Conference on AI* 366–372, 1987.

4. Cooper, G. F., Probabilistic inference using belief networks is NP-hard, Stanford Univ. Knowledge Systems Laboratory Memo KSL-82-27, May 1987 (revised July 1988). (Short version appeared as The computational complexity of probabilistic inference using Bayesian belief networks, *AI* **42**, 393–405, 1990.)

5. Valtorta, M., Some results on the computational complexity of refining confidence factors, *Int. J. Approx. Reasoning* **5**, 123–148, 1991.

6. Shafer, G., Shenoy, P. P., and Mellouli, K., Propagating belief functions in qualitative Markov trees, *Int. J. Approx. Reasoning*, 349–400, 1987.

7. Shenoy, P. P., and Shafer, G., Propagating belief functions with local computations, *IEEE Expert* **1**(3), 43–52, 1986.

8. Shenoy, P. P., and Shafer, G., An axiomatic framework for Bayesian and belief-function propagation, *Proceedings of the Fourth Workshop on Uncertainty in AI*, 307–314, 1988.

9. Zarley, D. K., An evidential reasoning system, Working Paper 206, Businesss School, Univ. Kansas, 1988.

10. Mellouli, K., On the propagation of beliefs in networks using the Dempster–Shafer theory of evidence, Ph.D. Dissertation and Working Paper No. 196, School of Business, Univ. Kansas, April 1988.

11. Gordon, J., and Shortliffe, E. H., A method for managing evidential reasoning in a hierarchical hypothesis space, *AI* **26**, 323–357, 1985.

12. Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

13. Lauritzen, S. L., and Spiegelhalter, D. J., Local computations with probabilities on graphical structures and their applications to expert systems, *J. Roy. Stat. Soc. Ser. B* **50**(2), 157–224, 1988.

14. Kyburg, H. E., Jr., Bayesian and non-Bayesian evidential updating, *AI* **31**, 271–293, 1987.

15. Duda, R. O., Hart, P. E., and Nilsson, N. J., Subjective Bayesian methods for rule-based inference systems, *Proceedings of the 1976 National Computer Conference*, 1075–1082. [Reprinted in *Readings in Artificial Intelligence* (B. L. Webber and N. J. Nilsson, Eds.), Morgan-Kaufmann, San Mateo, Calif., 192–199, 1981.]

16. Tanimoto, S. L., *The Elements of Artificial Intelligence Using Common LISP*, Computer Science Press, New York, 1990.

17. O'Leary, D. E., Soliciting weights or probabilities from experts for rule-based expert systems, *Int. J. Man-Mach. Stud.* **32**, 293–301, 1990.

18. Breuker, J., Wielinga, B., van Someren, M., de Hoog, R., Schreiber, G., de Greef, P., Bredeweg, B., Wielemaker, J., Billault, J.-P., Davoodi, M., and Hayward, S., Model-driven knowledge acquisition: interpretation models, Deliverable task A1, ESPRIT Project 1098, and Memo 87, VF Project Knowledge Acquisition in Formal Domains, 1987. (This report is available from the Dept. of Social Science Informatics, Univ. Amsterdam.)

19. Karbach, W., Linster, M., and Voss, A., Models, methods, roles, and tasks: many labels—one idea?, *Knowledge Acquisition* **2**, 279–299, 1990.

20. Orponen, P., Dempster's rule of combination is #P-complete, *AI* **44**, 245–253, 1990.

21. Provan, G. M., A Logic-based analysis of Dempster–Shafer theory, *Int. J. Approx. Reasoning* **4**(5/6), 451–495, 1990.

22. Segre, A. M. (Ed.), *Machine Learning: Proceedings of the Sixth International Workshop*, Morgan-Kaufmann, San Mateo, Calif., 1989.

23. Judd, S., Complexity of connectionist learning with various node functions, Tech. Rep. 87-60, Univ. Massachusetts at Amherst, July 1987.

24. Judd, S., Learning in neural networks, *Proceedings of the 1988 Workshop on Computational Learning Theory* (COLT-88), 2–8, 1988.

25. Blum, A., and Rivest, R. L., Training a 3-node neural network is NP-complete, *Proceedings of the 1988 Workshop on Computational Learning Theory* (COLT-88), 9–18, 1988.

26. Lin, J.-H., and Vitter, J. S., Complexity issues in learning by neural nets, *Proceedings of the Second Annual Workshop on Computational Learning Theory* (COLT-89), 118–133, 1989.

27. Fu, L., Truth maintenance under uncertainty, *Proceedings of the Fourth Workshop on Uncertainty in AI* 119–126, 1988.

28. Laskey, K. B., Adapting connectionist learning to Bayes networks, *Int. J. Approx. Reasoning* **4**, 261–282, 1990.

29. Judd, S., *Neural Network Design and the Complexity of Learning*, MIT Press, Cambridge, Mass., 1990.

30. Kong, C. T. A., Multivariate belief functions and graphical models, Ph.D. Dissertation, Dept. of Statistics, Harvard Univ., 1986. (Available as Res. Rep. S-107, Dept. of Statistics, Harvard Univ.)

31. Wilkins, D. C., and Buchanan, B. G., On debugging rule sets when reasoning under uncertainty, *Proceedings of AAAI-86*, 448–454, 1986.

32. Valtorta, M., Automating rule strength in expert systems, Ph.D. Dissertation, Dept. of Computer Science, Duke Univ., April 1987. (Also Tech. Rep. CS-1987-15, Dept. of Computer Science, Duke Univ., and available as ADG87-25869 from University Microfilm International.)

33. Valtorta, M., Some results on knowledge base refinement with an oracle, Tech. Rep. TR89005, Dept. of Computer Science, Univ. South Carolina, April 1989.