AN ANALYSIS OF ATMS-BASED TECHNIQUES FOR COMPUTING DEMPSTER-SHAFER BELIEF FUNCTIONS

Gregory M. Provan*
Department of Computer Science
University of British Columbia
Vancouver, BC
Canada V6T 1W5

Abstract

This paper analyzes the theoretical underpinnings of recent proposals for computing Dempster-Shafer Belief functions from ATMS labels. Such proposals are intended to be a means of integrating symbolic and numeric representation methods and of focusing search in the ATMS. This synthesis is formalized using graph theory, thus showing the relationship between graph theory, the logic-theoretic ATMS description and the set-theoretic Dempster Shafer Theory description. The computational complexity of calculating Belief functions from ATMS labels using algorithms originally derived to calculate the network reliability of graphs is analyzed. Approximation methods to more efficiently compute Belief functions using this graphical approach are suggested.

1 Introduction

To bridge the gap between the claimed lack of a "logical semantics" in uncertainty calculi and the lack of notions of uncertainty (claimed essential to modeling human reasoning) in logic, several attempts have been made to integrate formal logic with an uncertainty calculus. In this paper the relationships between an uncertainty calculus, Dempster Shafer Theory, and propositional logic are shown.

It has been proposed that Dempster Shafer (DS) Theory rivals Probability Theory in expressive power and effectiveness as a calculus for reasoning under uncertainty. However, because of the computational complexity associated with computing DS Belief functions, only subsets of the full problem domain expressible in DS Theory have been implemented, with the exception of recent Assumption-based TMS (ATMS) implementations. The number of subsets of a set of propositions Θ increases exponentially with $|\Theta|$, and given that the DS normalizing function can sum over all of these subsets, computing a single normalization function can be computationally expensive. The total space necessary to compute DS belief functions over a set of n propositions is $|2^{\Theta}| = 2^{2^n}$ in the worst case.

*The author completed this research with the support of the University of British Columbia Center for Integrated Computer Systems Research, BC Advanced Systems Institute and NSERC grants to A.K. Mackworth.

Examples of such restricted implementations include work by Shafer and Logan and by d'Ambrosio. Shafer and Logan [1987] have implemented DS Theory restricted to the case of hierarchical evidence, based on proposals by Barnett [I98I] and Gordon and Shortliffe [1985]. D'Ambrosio [1987] has implemented DS theory for the restricted case defined by the Support Logic Programming of Baldwin [1985]. D'Ambrosio attaches a *simplification* of the Dempster-Shafer uncertainty bounds to ATMS labels.

Laskey and Lehner [1988], Provan ([1988b], [1989a]) and Pearl [1988] have independently extended the ATMS with the full DS theory in similar manners. Such an extension represents a synthesis of the symbolic (logic-theoretic) ATMS representation and the numeric (set-theoretic) DS Theory representation. However, there has been no analysis of the underlying theoretical foundations or the complexity of this DS theory implementation.

This paper describes the theoretical and computational issues raised by extending an ATMS to compute DS Belief functions without significantly compromising the semantic clarity or computational properties of either. This extension is motivated by the need to introduce a weighting system into the ATMS, and to improve the poor performance of the ATMS as observed in practice and predicted by the average-case results of Provan [1989b]. Existing ATMSs cannot rank hypotheses, but the incorporation of DS Theory allows hypotheses to be ranked and offers efficiency improvements, such as the ability of the Problem Solver to prune the search space by identifying and focusing search only on highly likely partial solutions. Using a graph theoretic formulation of the ATMS we show that simply weighting the edges of the graph corresponds to a representation from which DS Belief functions can be computed. Such a graph theoretic formulation provides clear intuitions into the properties of symbolic ATMS-based DS algorithms and their relationship to the computation of network reliability [Agrawal and Barlow, 1984].

The complexity of the problems solved when using the ATMS to compute DS Belief functions is also defined.

Such an analysis identifies the sources of intractability in computing DS Belief functions, and explicitly shows why most implementations are of particular restrictions, such as hypertree embeddings [Shenoy and Shafer, 1988], [Shafer and Logan, 1987]. Such theoretical results are supported by empirical evidence (e.g. as pointed out in [Provan, 1986], [Ball, 1986]) which indicates that only moderately-sized problems can be computed efficiently. To counter such intractability, approximation methods for computing the full DS Theory, and not for restrictions of the theory are proposed.

This paper is organized as follows. Section 2 reviews the theoretical basis of DS theory. Section 3 presents three complementary theoretical formulations of the ATMS based on logic, Boolean expression minimization and graph theory. The correspondence between the graph theoretic ATMS formulation and DS theory is shown. Section 4 describes the ATMS-based DS theory algorithm and Section 5 the complexity of the underlying problems. Finally, in Section 6 the conclusions of this study are stated.

2 Dempster-Shafer Theory Review

Many good descriptions of Dempster-Shafer (DS) theory exist, e.g. [Dempster, 1968], [Shafer, 1976]. We assume familiarity with DS theory, state a few basic relationships, and refer the reader to the references.

In DS theory, weights are assigned to subsets as well as elements of a mutually exclusive set of focal propositions Θ. A mass function $\varrho : 2^{\Theta} \rightarrow [0,1]$ assigns weights to subsets θ of Θ subject to the following properties: $\varrho(\theta) \in [0,1]$, $\sum_{\theta \subseteq \Theta} \varrho(\theta) = 1$ and $\varrho(\emptyset) = 0$.

One measure in DS theory which is derived from this mass function is *Belief,* the degree of belief in proposition subsets from which a proposition θ can be proven:

$$Bel(\theta) = \sum_{\varphi \subseteq \theta} \varrho(\varphi). \qquad (1)$$

Dempster's Rule of Combination defines an updated mass function for a proposition *0* provable in terms of θi, and *θj,* for all θ,θi,θj C Θ, as:

$$\varrho'(\theta) = \frac{\sum_{i,j:\theta_i \cap \theta_j = \theta} \varrho_1(\theta_i)\varrho_2(\theta_j)}{1 - \sum_{i,j:\theta_i \cap \theta_j = \emptyset} \varrho_1(\theta_i)\varrho_2(\theta_j)} \qquad (2)$$

The Belief function ϱ' is also denoted as *Bel*\ ⊕ jBe/2. The numerator assumes independence of propositions. Viewed in set-theoretic terms, this is simply "summing" the mass functions of all sets in which θ is provable.[1] The denominator of Equation 2 is a normalizing term, given that DS Belief is assigned only to non-contradictory subsets.

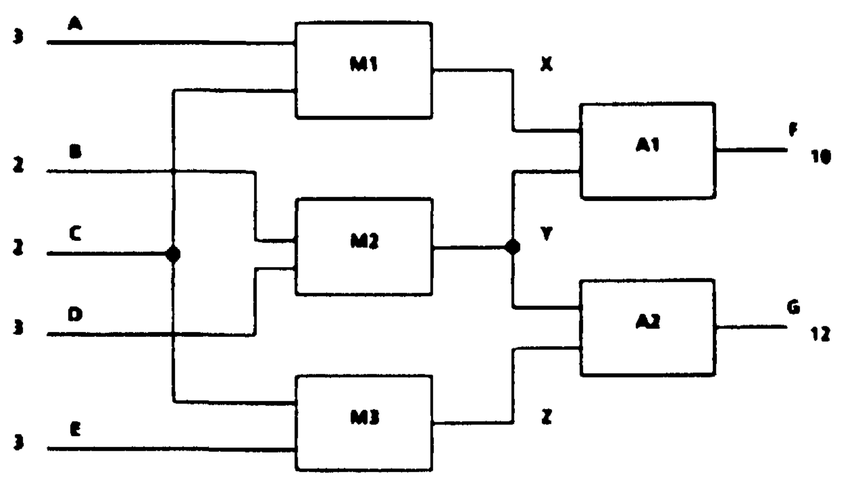
3 Theoretical ATMS Formalization

The ATMS [de Kleer, 1986] is a database management system which, given a set Σ of propositioned clauses, computes a label, $\mathcal{L}(x)$, for each database literal *x.* $\mathcal{L}(x)$ summarizes the "proofs" for *x* in terms of a Boolean formula consisting of assumptions only. The assumptions, which are denoted by *A* = {A1,...,A/}, are a distinguished subset of the database literals and are the primitive data representation of the ATMS with which the derivation of all other literals is recorded.

The ATMS records contradictions in terms of a conjunction of assumptions called a *nogood.* By ensuring null intersections of all labels with the set of nogoods, the ATMS maintains a consistent assignment of labels to database literals. The ATMS can incrementally update the database labeling following the introduction of new clauses. It does this by storing the entire label and nogood set to avoid computing them every time they are needed.

A typical problem for which an ATMS is used is circuit diagnosis, such as that done by GDE [de Kleer and Williams, 1987]. The circuit analyzed in [de Kleer and Williams, 1987] consists of multipliers M1, M_2 and M_3 and adders *A1* and *A2,* as shown in Figure 1. Assumptions can be: (1) each component is working,

Figure 1: Circuit with faulty components



where W A , signifies that adder *Ai* is functioning correctly, and W M , signifies that multiplier M_i is functioning correctly; or (2) input data, e.g. *A* = 3, *B* = 2, etc. In the course of diagnosis, an assumption like *WM2,* i.e. "M2 is working", may be proved incorrect. For the circuit in Figure 1, the output at F is 10 instead of 12, implying that some combination(s) of M_1, M_2, M_3, A_1 and A_2 is(are) faulty. In GDE, the ATMS identifies hypothesized sets of circuit components whose faulty behavior could cause discrepancies between predicted and observed circuit measurements. Taking observations at points like X, Y or Z narrows the set of diagnoses consistent with the observations and guides future decisions about where to make further readings. A solution consists of a set of faulty multipliers and adders which explains all the observations.

Given the set of input clauses and assumptions, the ATMS computes what de Kleer and Williams call minimal conflict sets, which are the labels for circuit malfunctions. The two conflict sets are represented logically as $\neg(W_{A_1} \wedge W_{M_1} \wedge W_{M_2})$ and $\neg(W_{A_1} \wedge W_{A_2} \wedge W_{M_1} \wedge W_{M_3})$. Hence, the malfunctioning of the circuit shown in Figure 1 can be explained by the simultaneous malfunctioning of A_1, M_1 and M2, or

[1] DS Theory thus has both underlying set-theoretic and logic-theoretic notions.

that of *A1,* A2, M1 and M3.

We now present three complementary methods of viewing the problems solved by the ATMS. We first consider the two frameworks traditionally used to formalize the ATMS, namely propositional logic and Boolean algebra. We then show the equivalence of a graphical framework.

3.1 Logic Formulation

The input alphabet (using terminology introduced in [Reiter and de Kleer, 1987]) is represented by a set x = {x1,...,x_n} of propositional symbols. A propositional literal is a propositional symbol or its negation. A propositional clause consists of a finite disjunction of propositional literals with no literal repeated, e.g. $\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \ldots \vee \overline{x_k} \vee x$, $k \geq 0$.[2] $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_k}$ are called the *antecedents* of x, $ant(x)$, in keeping with ATMS terminology.

We define an *environment* E to be a set of assumptions and a *context* C to be the set of literals derivable from the assumptions in E given the clauses.

The input consists of a set $\Sigma = \{\Sigma_1, \ldots, \Sigma_l\}$ of *Horn clauses*, referred to in [de Kleer, 1986] as justifications. A Horn clause is a clause with at most one unnegated literal.

The ATMS's operation consists of two distinct phases, label manipulation and interpretation construction, which is described below.

Label Manipulation: In the label manipulation phase, for each literal the ATMS maintains a *label*, which is a set of environments in which the literal can be proven. A typical label is $\{\{A_1, A_3, A_4\}, \{A_5, A_6\}\}$, showing two environments. In logical terms, the Boolean expression for the label is $(A_1 \wedge A_3 \wedge A_4) \vee (A_5 \wedge A_6)$, or the disjunction of the environments. Each environment in the label for x_i is defined in terms of a minimal support clause ξ^* for x_i, and is given by $\xi^* = \{\bigwedge_{A_j \in \mathcal{A}} A_j \mid (\bigvee_{A_j \in \mathcal{A}} \overline{A_j})$ is a minimal support clause of $\Sigma\}$. ξ is a *support clause* for x_i with respect to Σ if $\Sigma \not\models \xi$, $x_i \cup \xi$ does not contain a complementary pair of literals (i.e. both x_j and $\overline{x_j}$), and $\Sigma \models x_i \cup \xi$. ξ^* is a minimal support clause for x_i with respect to Σ if no proper subset of ξ is a support clause for x_i with respect to Σ. A closely related type of clause, a prime implicate,[3] is defined as follows: a prime implicate of a set Σ of clauses is a clause π such that (1) $\Sigma \models \pi$, and (2) for no proper subset π' of π does $\Sigma \models \pi'$. The relationship between the label (minimal support clause) for a literal x and a prime implicate for Horn clauses is given as follows [Reiter and de Kleer, 1987]: ξ is a minimal support clause for x with respect to Σ iff there is a prime implicate π for Σ such that $x \in \pi$ and $\xi = \pi - x$. The ATMS thus computes the set $\Pi(\Sigma)$ of prime implicates and assigns as a label for each literal the disjunction of support clauses (restricted to assumptions) for that literal.

If a new clause $\overline{x_1} \vee \overline{x_2} \vee \cdots \vee x$ is added such that the antecendents of x already have labels, or if (x) must be updated due to database updating, label updating must take place. An updated label is assigned to a literal x by taking the conjunction of the labels of the antecedents of x: $\mathcal{L}(x) = \bigwedge_{x_i \in ant(x)} \mathcal{L}(x_i)$. Labels are always represented minimally with respect to set inclusion.

The process of determining solutions, called Interpretation Construction in ATMS terminology, can be described in this logical framework in terms of a CNF to DNF conversion, but is more intuitively described within the Boolean expression framework, in terms of well-known set covering algorithms. We discuss Interpretation Construction in the next section.

3.2 Boolean Algebra Formulation

This formulation uses the same terminology as the logical formulation. The conjunction of the E^s is called a Boolean expression[4] F, i.e. $F = \bigwedge_{i=1,\ldots,l} \Sigma_i$.

As in the previous formulation, the ATMS label generation algorithm computes the prime implicates for Σ, $\Pi(\Sigma) = \{\pi_1, \ldots, \pi_m\}$. The derivation of the prime implicates enables the expression F to be represented in terms of the prime implicates, i.e. $F_\Pi = \bigwedge_i \pi_i$, such that *Fn* computes *F.* A expression *F'* computes *F* if *F'(x) = F(x)* for every instantiation of x. We note that there may be many other expressions *F'* which also compute *F.*

The ATMS then derives the *minimal* or *irredundant* expression *T* for *F.* By assigning a unit cost to each support clause ξ_i, the cost of *F* can be defined as the sum of the costs of the support clauses ξ_i in F.[5] A minimal expression *T* is an expression such that *T* computes *F* and no expression computing *F* has cost smaller than *T.*

Interpretation construction derives an irredundant (or minimal) set of prime implicates, where a prime implicate *w* is redundant if it can be removed from the set II of prime implicates to form a set II' such that II' still computes *F.* The ATMS uses a set covering algorithm for interpretation construction, but other algorithms may be used, such as the Quine-McCluskey algorithm [McCluskey, 1956], or the hitting set algorithm used by Reiter [1987] for a diagnostic reasoning application. The minimal Boolean expression corresponds in ATMS terminology to an *interpretation,* which de Kleer [1986] defines as the smallest set of assumptions from which all literals in the context are derivable.

For the circuit in Figure 1, a minimal DNF expression $\mathcal{F}$ is $(\overline{W_{A_1}}) \vee (\overline{W_{M_1}}) \vee (\overline{W_{A_2}} \wedge \overline{W_{M_2}}) \vee (\overline{W_{M_2}} \wedge \overline{W_{M_3}})$, which is what de Kleer and Williams call a minimal diagnosis.

3.3 Graph Theoretic Formulation

In describing this formulation, we introduce some graph-theoretic notation, which is used to show the equivalence

---

[2] Any clause can be represented as either $x_i \supset x_j$ or $\overline{x_i} \vee x_j$. The latter method is used by convention.

[3] The dual to prime implicate (in Boolean algebra) is called a prime implicant. We use the prime implicate terminology to avoid confusion between the dual representations.

[4] This is standard conjunctive normal form (CNF), the dual representation of traditional disjunctive normal form (DNF) Boolean expressions.

[5] There are many ways to define such a cost function. For example, the cost of each support clause ξ_i can be assigned as the number of assumptions in ξ_i.

of the graph-theoretic and logic-theoretic descriptions of the problem.

We can consider a Boolean expression *F* as defining a graph $\mathcal{G}(V, E)$ composed of vertices *V* and edges *E*[6]. More precisely, a Boolean literal *xi* corresponds to an edge *Ei,* and a logical connective $(\vee, \wedge)$ corresponds to a vertex that joins two or more edges between the corresponding components as follows: a A connecting two literals (or clauses) corresponds to an edge connecting two vertices (or vertex sets) in series, and a V connecting two literals (or clauses) corresponds to an edge connecting two vertices (or vertex sets) in parallel. The direction of the edges corresponds to the direction of implication for the clauses.

A *path* consists of a connected sequence of distinct edges. We call £ a path between vertices *s* and *t* in the event that all edges in the path are functioning. A *minimal path* is a path the deletion of any edge of which renders the path disconnected. A *subgraph* $\mathcal{G}'(V', E')$ of $\mathcal{G}(V, E)$ is a graph such that $V' \subseteq V$ and $E' \subseteq E$ A *connected graph* has at least one path between every pair of vertices. A *cutset* of a graph *Q* is a subgraph of $\mathcal{G}$ the removal of any edge (or vertex) of which renders $\mathcal{G}$ disconnected.

Using the correspondence between the Boolean expression and graph theoretic formulations for the ATMS, we outline the graph-theoretic equivalent of the ATMS labels. But instead of the label, we will use the prime implicate π, since a label is just another means of representing π, as shown earlier.

Lemma 1 *The set of prime implicates* $\Pi(\Sigma)$ *for a CNF Boolean expression F defines a set of paths through the corresponding graph* $\mathcal{G}$.

If *F* is expressed in DNF, then we obtain

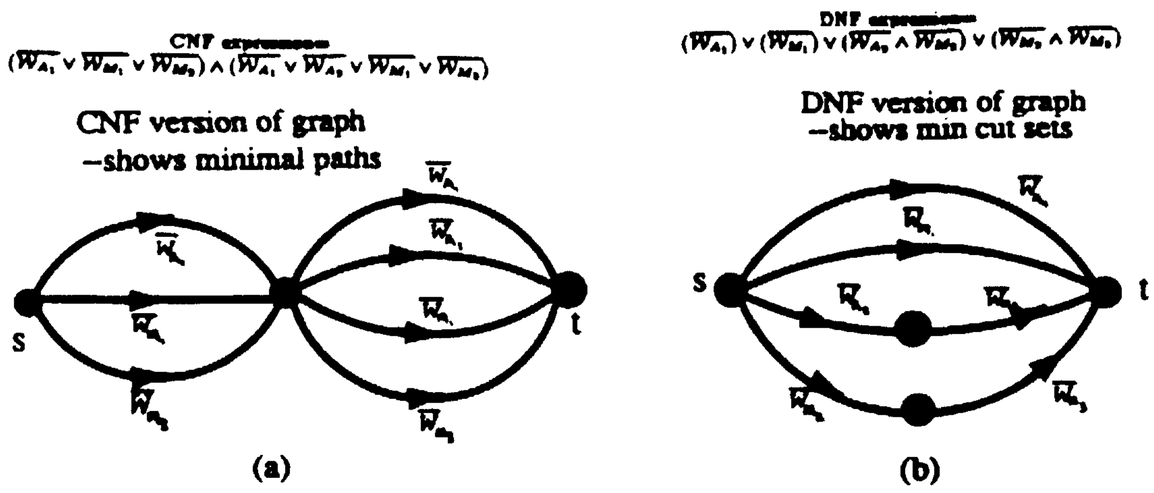
Lemma 2 *The set of prime implicates* $\Pi(\Sigma)$ *for a DNF Boolean expression F defines the cut sets of the corresponding graph* $\mathcal{G}$.

The graph corresponding to the irredundant CNF expression for the circuit diagnosis example is shown in Figure 2(a). The graph corresponding to the irredundant DNF expression is shown in Figure 2(b) which showns the minimal paths derived from the example. We emphasize that these graphs are interconvertible. By finding the edge cuts containing the minimum number of edges for the graph shown in Figure 2(a), we can obtain a minimal cut representation shown in Figure 2(b). Note the graph theoretic relationships between the diagnostic notions of minimal conflict sets (minimal cut sets) and minimal candidates (minimal paths).

Computing the pathsets or cutsets of a graph $\mathcal{G}$ is one means of computing the network reliability of $\mathcal{G}$, which we now show is analogous to using the ATMS label set to calculate DS Belief function assignments for the corresponding Boolean expression *F.*

There are many ways of defining a graph from a Boolean expression. We present one method here, and outline other methods in [Provan, 1989a].

Figure 2: Graphs corresponding to circuit diagnosis formulae



4 Correspondence between the ATMS and Dempster-Shafer Belief Functions

4.1 Extension of ATMS Graph

We now show how the simple assignment of weights to the underlying ATMS graph G enables the computation of DS Belief functions from the very same graph. Assume each edge of G is associated with a statistically independent random variable with only two possible states, functioning or not functioning. For each DS proposition *9* there is an associated ATMS literal x and an antecedent set *ant(x)* containing at least one assumption, such as x3 V *A2* V *x.* An *event* is a state assignment to literals, such as {x1,X2,.....,xn-1} functioning and x_n not functioning. There are 2^n possible events. Assign to each edge a [0,1] weight, *g* : *E* —► [0,1], which is the probability that the edge functions properly. Each edge in G corresponds to an ATMS assumption, and the edge weight corresponds to a weight assigned to the ATMS assumption.

The reliability *R(s,t)* of a graph $\mathcal{G}$ with respect to two distinguished vertices s and *t* is defined as the probability that an *s* — *t* path exists. The *s* — *t* reliability problem is defined as follows:

R(s,t): Given a graph $\mathcal{G}(V, E)$ and a probability assignment *g to E,* compute the probability that a functioning path exists between two distinguished vertices *s* and *t.*

R(s, t) can be computed by summing the probabilities of the disjoint *s* — *t* path set. A disjoint path set is a set of paths such that no pair of paths has a common edge. Any pathset, cutset or equivalent Boolean expression for computing *R(s,t)* must be disjoint.

We summarize the results of [Provan, 1989a] as follows:

Lemma 3 *The label assigned to a literal x is a symbolic (and not necessarily disjoint) representation of the DS Belief assigned to the corresponding DS proposition* θ.

There is an ATMS corollary to DS information pooling (equation 2):

Lemma 4 *The ATMS label updating for a literal x,* $\mathcal{L}(x) = \bigwedge_{x_i \in ant(x)} \mathcal{L}(x_i)$, *is a symbolic ATMS analog*

to the DS Belief updating formula for the corresponding proposition θ, $Bel(\theta) = \bigoplus_i Bel_i(\theta)$.

Numeric assignments *of Bel* can be calculated as given by:

Lemma 5 *The DS Belief assigned to a literal can be computed using an ATMS by converting the weighted ATMS label set to its graphical representation and computing the probability that an s — t path exists in the subgraph formed from the label assigned to the literal.*

Hence calculating DS belief functions for an underlying Boolean expression *F* is identical to computing the network reliability for the graph corresponding to *F.*

Moreover, there is an analog in the ATMS to Dempster's rule of Conditioning. Dempster's Rule of Conditioning is as follows:

Lemma 6 *If Bel and Bel′ are two combinable Belief functions,*[7] *let* $Bel(\cdot|\theta_2)$ *denote* $Bel \bigoplus Bel'$. *Then*

$$Bel(\theta_1|\theta_2) = \frac{Bel(\theta_1 \cup \overline{\theta_2}) - Bel(\overline{\theta_2})}{1 - Bel(\overline{\theta_2})} \qquad (3)$$

for all $\vartheta_1 \subset \Theta$.

If we call the set of nogoods Ψ, *then the ATMS's symbolic representation of equation 3 is, for all* $x \in \mathbf{x}$,

$$Bel(x \mid \neg\Psi) = \frac{Bel[\mathcal{L}(x) \cup \Psi] - Bel[\Psi]}{1 - Bel[\Psi]}. \qquad (4)$$

4.2 Dempster-Shafer Belief Function Algorithm

DS Belief functions can be computed from ATMS labels as follows. To each ATMS assumption *Ai* assign a DS mass function *g(Ai).* Then use the ATMS to symbolically compute the labels for all the literals. Similar to the ATMS's removal of nogoods from all contexts as an essential part of its processing, DS theory can assign belief only to non-contradictory subsets, i.e. it "removes" mass assigned to contradictory subsets because *6* is not provable for subsets which have a non-null intersection with contradictory subsets. In the ATMS implementation, nogoods must be explicitly accounted for in computing DS Belief functions. This is done by: (1) assigning belief in the numerator of equation 2 only to subsets with null intersections with nogoods (i.e. conditioning on the consistent sets); and (2) using a normalization function, the denominator of equation 2, to renormalize all Belief assignments given that no mass is assigned to nogoods. Hence, the Belief assigned to any proposition (literal) can be computed as follows:

1. Compute a Boolean expression from the label, as described in Section 3.1.
2. Account for nogoods, using equation 4, or the equivalent form

$$Bel(x|\neg\Psi) = \frac{Bel[\mathcal{L}(x)] - Bel[\mathcal{L}(x) \cap \Psi]}{1 - Bel[\Psi]}. \qquad (5)$$

3. Convert the Boolean expression (4) or (5) into a disjoint form.

[7] cf. [Shafer, 1976], p.67 for a definition of conditions for combinability.

Table 1: Correspondence of Boolean and Network Reliability Forms

Boolean Form	Network Reliability Form
x_i	$\varrho(x_i)$
$\overline{x_i}$	$(1 - \varrho(x_i))$
$\wedge$	arithmetic product
$\vee$	arithmetic sum

4. Substitute mass functions for the Ai's to calculate the Belief function for x, using Table 1.

Note that in Steps 1 to 3 we manipulate Boolean expressions. We refer to steps 3 and 4 as a Network Reliability computation.

There are many methods for symbolically computing the disjoint form of an arbitrary Boolean expression, some of which are reviewed in [Agrawal and Barlow, 1984]. The methods applicable to this application are those based on pathsets and cutsets, such as the Sum of Disjoint Products and Inclusion-Exclusion methods.

5 Complexity of ATMS-based Dempster-Shafer Belief Function Computation

This extension of the ATMS was originally proposed as a means of improving the efficiency of the ATMS by focusing search only on the portions of the search space in which solutions were most likely to be found. This need arose from the poor performance of the ATMS in practice, and the theoretical evidence ([Provan, 1988a], Provan, 1989b]) which suggests that (1) for almost all Boolean expressions[8] the problem of label generation is of complexity exponential in the number of literals; and (2) the interpretation construction problem is NP-hard.

However, the efficiency gains from direct ATMS implementation of DS Theory may not be cost-effective. Using the ATMS to compute DS Belief functions from ATMS labels is intractable in the worst case, a computational barrier similar to that which has previously prevented implementation of the full DS theory.

Lemma 7 *Given the set of ATMS labels for a set x of literals and an assignment of weights to the ATMS assumption set A, computing the DS Belief assigned to x either from the ATMS label set* E* *or from a minimal expression F is of complexity exponential in the number of literals in the Boolean expression F, tn the worst case.*

If the ATMS label set is used as a starting point, the size of the label set is exponential in the number of literals or clauses [Provan, 1989b], just as the number of *s — t* pathsets/cutsets is an exponential function of | *V* \ and | *E* | [Agrawal and Barlow, 1984]. The minimal expression *T* may also be used as a starting point, but even

[8] A property is said to hold for almost all the functions of the algebra of logic if the proportion of functions of n variables which do not satisfy this property (among all the functions of n variables) tends to zero when n —> oo.

though $C08t\{T) < co8t(F_{II})$, it is unknown whether the savings introduced will be worth computing F from F_{II}.

If truth maintenance facilities and numeric assignments of weights to proposition sets are required, this method may be useful, depending on the tradeoff between the cost of computing the DS Belief functions and the search space which can be saved by more focused search. The graphical framework suggests more efficient methods of computing ATMS labels and DS belief functions by exploiting the topology of the underlying graph. Further efficiency improvements may be obtained by using one of the many *R(s,t)* approximation algorithms applicable to computing DS Belief functions from ATMS labels. In [Provan, 1989a] we describe network reliability *approximation* algorithms which avoid the intractability associated with deriving exact solutions. Such algorithms may remove the necessity to restrict the problem domain to a subset such as that of hierarchical evidence [Shafer and Logan, 1987] due to intractability otherwise.

6 Conclusions

The theoretical basis underlying methods of computing DS Belief functions from ATMS label sets have been discussed. These computations are done using an ATMS which incorporates the full DS theory in a semantically clear and efficient manner, as shown by the graph-theoretic equivalence of the ATMS and DS theory. In addition, computing DS Belief functions from ATMS labels was shown to be intractable in the worst case.

The use of the ATMS to compute DS Belief functions is a promising research area, as it is a means of blending logic with an uncertainty representation in a semantically correct manner. However, we argue that the computational cost of computing *exact* DS Belief functions may be prohibitive for large problems, and approximation methods may have to be used.

Acknowledgements: Discussions with Judea Pearl have helped refine my understanding of the correspondence of the semantics of the ATMS and DS theory, and the use of network reliability algorithms.

References

[Agrawal and Barlow, 1984] A. Agrawal and R.E. Barlow. A Survey of Network Reliability and Domination Theory. *Operations Research,* 32(3):478-492, 1984.

[Baldwin, 1985] J.F.Baldwin. Evidential Support Logic Programming. *Fuzzy Sets and Systems,* 24:1-26, 1985.

[Ball, 1986] M.O. Ball. Computational Complexity of Network Reliability Analysis: An Overview. *IEEE Trans. Reliability,* R-35:275-285, 1986.

[Barnett, 198I] J.A. Barnett. Computational Methods for a Mathematical Theory of Evidence. In *Proc. IJCAI* :868-875, 1981.

[D'Ambrosio, 1987] B.D. D'Ambrosio. Combining Symbolic and Numeric Approaches to Uncertainty Management. In *AAAI Uncertainty in AI Workshop* :386-393. Morgan Kaufmann, 1987.

[de Kleer and Williams, 1987] J. de Kleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence,* 32:97-130, 1987.

[de Kleer, 1986] J. de Kleer. An Assumption-based TMS. *Artificial Intelligence,* 28:127-162, 1986.

[Dempster, 1968] A.P. Dempster. Upper and Lower Probability Bounds. *J. Royal Statistical Society,* 1968.

[Gordon and Shortliffe, 1985] J. Gordon and E. Shortliffe. A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. *Artificial Intelligence,* 26:323-357, 1985.

[Laskey and Lehner, 1988] K. Blackmond Laskey and P.E. Lehner. Belief Maintenance: An Integrated Approach to Uncertainty Management. In *Proc. AAAI* :210-214, 1988.

[McCluskey, 1956] E.J. McCluskey. Minimization of Boolean Functions. *Bell Syst. Tech. J.,* 35:1417-1444, 1956.

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

[Provan, 1986] J.S. Provan. Bounds on the Reliability of Networks. *IEEE Trans. Rel,* R-35:260-268, 1986.

[Provan, 1988a] G. Provan. A Complexity Analysis of Assumption-Based Truth Maintenance Systems. In B.M. Smith and G. Kelleher, editors, *Reason Maintenance Systems and their Applications.* Ellis Horwood, 1988.

[Provan, 1988b] G. Provan. Solving Diagnostic Problems Using Extended Truth Maintenance Systems. In *Proc. EC A1*:547-552, 1988.

[Provan, 1989a] G. Provan. A Logic-Based Analysis of Dempster Shafer Theory. Tech. Report 89-8, University of British Columbia, Department of Computer Science, 1989.

[Provan, 1989b] G. Provan. The Computational Complexity of Truth Maintenance Systems. Tech. Report 89-9, University of British Columbia, Department of Computer Science, 1989.

[Reiter and de Kleer, 1987] R. Reiter and J. de Kleer. Foundations of Assumption-based Truth Maintenance Systems. In *Proc. AAAI* :183-188, 1987.

[Reiter, 1987] R. Reiter. Theory of Diagnosis from First Principles. *Artificial Intelligence,* 32:57-96, 1987.

[Shafer and Logan, 1987] G. Shafer and R. Logan. Implementing Dempster's Rule for Hierarchical Evidence. *Artificial Intelligence,* 33:271-298, 1987.

[Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[Shenoy and Shafer, 1988] P.P. Shenoy and G. Shafer. An Axiomatic Framework for Bayesian and Belief Function Propagation. In *AAAI Workshop on Uncertainty in AI* :307-314. Morgan Kaufmann, 1988.

Reflection and Action Under Scarce Resources: Theoretical Principles and Empirical Study

Eric J. Horvitz, Gregory F. Cooper, David E. Heckerman
Medical Computer Science Group
Knowledge Systems Laboratory
Stanford University
Stanford, California 94305

Abstract

We define and exercise the expected value of computation as a fundamental component of reflection about alternative inference strategies. We present a portion of Protos research focused on the interlacing of reflection and action under scarce resources, and discuss how the techniques have been applied in a high-stakes medical domain. The work centers on endowing a computational agent with the ability to harness incomplete characterizations of problem-solving performance to control the amount of effort applied to a problem or subproblem, before taking action in the world or turning to another problem. We explore the use of the techniques in controlling decision-theoretic inference itself, and pose the approach as a model of rationality under scarce resources.

1 Reflection and Flexibility

Reflection about the course of problem solving and about the interleaving of problem solving and physical activity is a hallmark of intelligent behavior. Applying a portion of available reasoning resources to consider the utility of alternative inference strategies or the value of continuing to refine a result before acting enables a computational agent to generate custom-tailored approaches to a wide variety of problems, under different time pressures. Such flexibility can be especially useful in light of uncertain deadlines and challenges. Uncertainty about problems and problem solving plagues simple agents immersed in complex environments. Constraints on an agent's reasoning and representation resources lead to inescapable uncertainties about the problems that may be faced and about the value of future reasoning in solving those problems.

The Protos project has pursued the use of decision theory for real-time control and offline problem-solving design. The work has highlighted opportunities for the principled control of reasoning under scarce resources with problems in sorting and searching and with decision-theoretic inference itself [Horvitz, 1987a]. We havp particularly dwelled on the decision-theoretic control of decision-theoretic inference as a model of rational computational inference under resource constraints. In this paper, we present a component of this work centering on the use of incomplete characterizations of the progression of probabilistic inference to reason about the value of continuing to reflect about a problem versus taking action in the world. This methodology uses knowledge that partially characterizes relevant dimensions of problem-solving performance. Such knowledge can be learned and refined with experience. We shall introduce components of utility for computational or real-world actions, and define the expected value of computation in terms of the likelihood of future probability distributions over the truth of relevant propositions about the state of the world. After discussing the theoretical principles and empirical results, we describe a component of research centering on the offline analysis of problem-solving trajectories. Such offline musing, weighted by expected challenges, can be important in real-time reflection about problem solving.

*This work was supported by a NASA Fellowship under Grant NCC-220-51, by the National Science Foundation under Grant IRI-8703710, by the National Library of Medicine under Grant RO1LM0429, and by the U.S. Army Research Office under Grant P-255I4-EL. Computing facilities were provided by the SUMEX-AIM Resource under N1H Grant RR-00785.

2 Decision-Theoretic Valuation

Decision theory provides the foundations for a principled approach to metalevel decision making under uncertainty. Decision-theoretic metareasoning can be especially useful in reasoning about the selection, and optimal halting time, of reasoning strategies that incrementally refine results as scarce resources are expended [Horvitz, 1987b, Dean and Boddy, 1988].

We usu *comprehensive value,* U_c, to refer to the utility associated with the value attributed to the state of an agent in the world. This value is a function of the problem at hand, of the agent's best default action, and of the stakes of a decision problem. We call the net change expected in the comprehensive value, in return for some allocation of computational resource, the expected value of computation (EVC). It is often useful to view the comprehensive utility, at any point in the reasoning process, as a function of two components of utility: the *object-level* utility, $u_{0,}$ and the *inference-*

related cost, u_i.[1] The *object-level* utility of a strategy is the expected utility associated with a computer result or state of the world. We say that the object-level utility is a function of a vector of attributes, $\vec{v}$. For example we may assign an object-level utility to an incompletely sorted file of records, based on several different dimensions of incompleteness. The *inference-related* component is the sum of the expected disutility intrinsically associated with, or required by, the process of problem solving. This cost can include the disutility of delaying an action while waiting for a reasoner to infer a recommendation. In general, the inference-related cost is a function of a vector of resource attributes, $\vec{r}$, representing the quantity that has been expended of such commodities as time and memory.

There is generally uncertainty in the object-level state resulting from the expenditure of computational resources. Thus, in the general case, we must sum over a probability distribution of object-level attributes to generate an expected comprehensive utility. If $\vec{v}$ and $\vec{r}$ are the vectors representing object-level and inference-related attributes without additional computation, respectively, and the $\vec{v}'$ and $\vec{r}'$ are the revised vectors, expected with additional computation, the net, or change in, comprehensive utility, given some allocation of resources is

$$\mathrm{EVC}(\vec{r}') = \sum_{\vec{v}'} u_c(\vec{v}', \vec{r}')\, p(\vec{v}'|\vec{r}') \; - \; u_c(\vec{v}, \vec{r})$$

In cases where the inference-related and object-level utilities can be decomposed, and are related through addition, the EVC is just the difference between the increase in object-level utility and the cost of the additional computation,

$$\mathrm{EVC}(\vec{r}') = [\sum_{\vec{v}'} u_o(\vec{v}')\, p(\vec{v}'|\vec{r}') - u_o(\vec{v})] - [u_i(\vec{r}') - u_i(\vec{r})]$$

In another study, we considered the refinement of multidimensional attributes of partial results with computation [Horvitz, 1988]. Here, we will simplify our object-level focus to a probability of a state in the world, *H*, and the quality of an associated decision to act, *A,* given uncertainty about the truth of the state. We will simplify the inference-related component to a consideration of computation time.

The decision-theoretic approach to metareasoning in difficult machine intelligence problems was introduced by I.J. Good over 2 decades ago, in the context of the control of game-playing search [Good, 1968]. Good had earlier discussed the explicit integration of the costs of inference within the framework of normative rationality, defining Type 1 rationality as inference that is consistent with the axioms of decision theory, regardless of the cost of inference, and Type II rationality as behavior that takes into consideration the costs of reasoning [Good, 1952]. Related work in decision science has focused on the likely benefit of expending effort for decision analyses [Matheson, 1968, Watson and Brown, 1978]. Our group researched the general applicability of decision-theoretic control of computation, with an emphasis on metareasoning problems with probabilistic inference and knowledge representation [Horvitz, 1987b]. Early investigation demonstrated that multiattribute decision-theoretic control of reasoning had promise for guiding the solution of a variety of tasks, including such fundamental problems as sorting a file of records or searching a large tree of possibilities [Horvitz, 1987a]. Indeed, there have been recent studies of the value of computation in the control of sorting [Horvitz, 1988] and of game-playing search [Russell and Wefald, 1988, Hansson and Mayer, 1989]. In related research on the control of logical inference, Smith, and Treitel and Genesereth, have explored the use of decision theory for selecting alternative logical reasoning strategies [Smith, 1986, Treitel and Genesereth, 1986].

More comprehensive notions of the value of a reasoning system in an environment are discussed in [Horvitz, 1987b].

3 Complexity of Inference

In reasoning about real-world actions under uncertainty, an agent generally must consider alternative decisions and outcomes, preferences about the possible outcomes, and the uncertain relationships among actions and outcomes. We have been investigating the use of *influence diagrams* [Howard and Matheson, 1981] for representing and solving automated reasoning problems. The influence diagram is an acyclic directed graph containing nodes representing propositions and arcs representing interactions between the nodes. Nodes represent a set of mutually exclusive and exhaustive states; arcs capture probabilistic relationships between the nodes. Influence diagrams without preference or decision information are termed *belief networks. A* belief network defines a model for doing probabilistic inference in response to changes in information.

The problem of probabilistic inference with belief networks is $\mathcal{NP}$-hard [Cooper, 1987]. Thus, we can expect algorithms for doing inference to have a worst-case time complexity that is exponential in the size of the problem (e.g., the number of hypotheses and pieces of evidence). Some methods for inference in belief networks attempt to dodge intractability by exploiting independence relations to avoid the explicit calculation of the joint-probability distribution. A variety of exact methods has been developed, each designed to operate on particular topologies of belief networks [Horvitz et al., 1988a]. Other methods forego exact calculation of probabilities; these approximation techniques produce partial results as distributions or bounds over probabilities of interest. The complexity of precise inference and the availability of alternative reasoning approaches highlight the need for robust approximation strategies and intelligent control techniques. We have sought to develop and control decision-theoretic inference for reasoning under uncertainty in high-stakes and time-pressured applications, such as medical decision making.

4 Decisions Under Scarce Resources

Let us explore concerns that arise in automated decision making under scarce resources. The graph in the lower

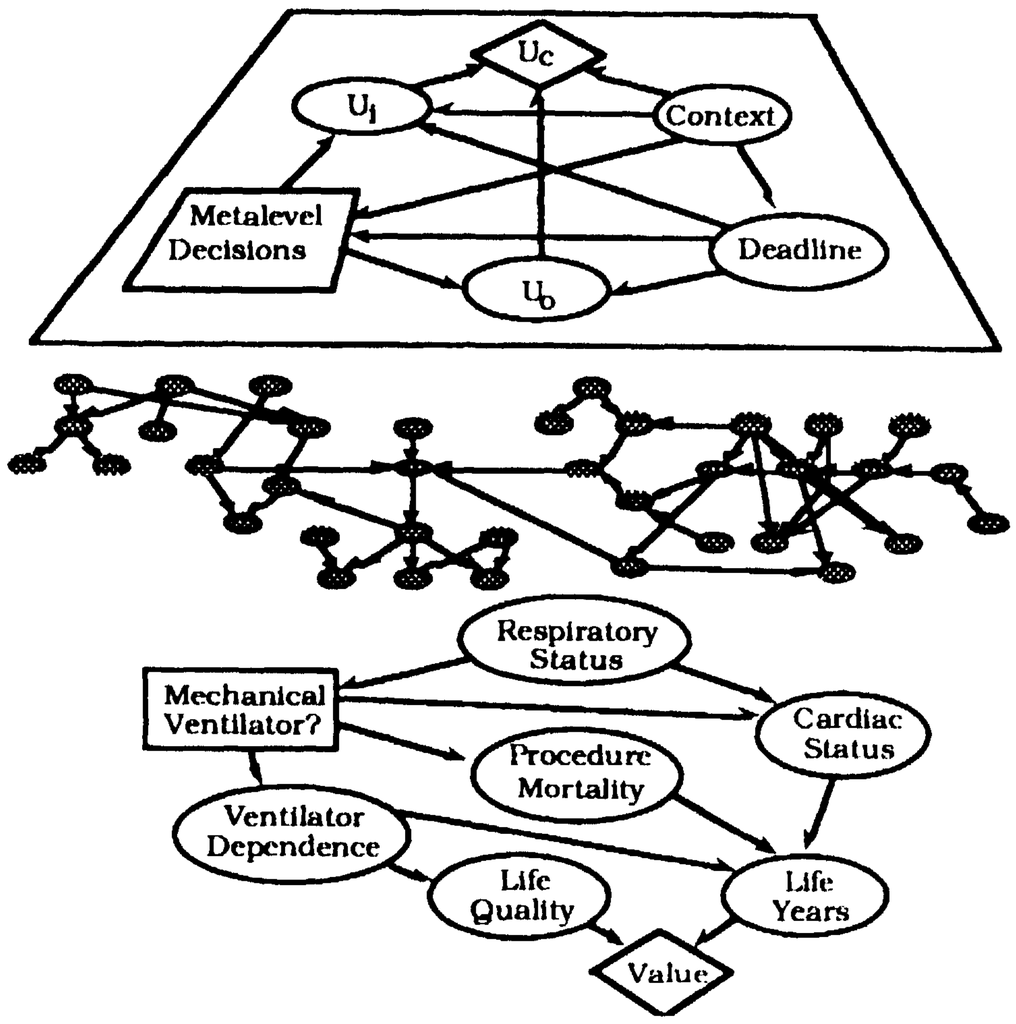


Figure 1: A representation of a time-pressured decision problem. From top to bottom, the three sections of the figure portray (a) the decision-theoretic metareasoning problem, (b) a belief network representing propositions and dependencies in intensive-care physiology, and (c) a closeup on the *respiratory status* node, and its relationship to the current decision problem.

portion of Figure 1 depicts an object-level influence-diagram representation of a time-pressured problem that might face an automated physician's assistant: A 75-year-old woman in the intensive-care unit suddenly shows signs of breathing difficulty. The patient may be merely showing signs of mild *respiratory distress* or may be in the more serious situation of *respiratory failure.* In this context, the primary decision is whether or not to recommend that the patient be placed on a mechanical ventilator. The decision (square node) depends on the probability of respiratory failure, which, in turn, depends on the probabilities of propositions in a large belief network serving as a medical knowledge base (represented by the graph above the object-level problem in Figure 1). The large oval nodes in the base decision problem represent uncertain states associated with placing an older person on a ventilator. The diamond represents the utility associated with different outcomes. Factors to consider in a decision to act include the possibility that it may take a long time to wean a patient with severe lung disease from a ventilator that is applied needlessly; thus, the patient may face a long hospital stay and be placed at high risk of mortality from a disease such as pneumonia. However, if the patient turns out to be in respiratory failure, and is not treated immediately, she faces a high risk of cardiac arrest based on the disrupted physiology associated with abnormal blood levels of oxygen and carbon dioxide.

4.1 Actions and Outcomes in the World

In our simple example, there are only four different fundamental outcomes. The patient either is in respiratory failure *(H)* or is not in respiratory failure $(\neg H)$, and we either will place the patient on a ventilator (.4) or will not do so $(\neg A)$. Thus, we may erroneously decide not to treat a patient who is suffering from respiratory failure $(\neg A, H)$, we may correctly treat a patient who is suffering from respiratory failure *(A,H),* we may erroneously treat a patient who is not suffering from respiratory failure $(A, \neg H)$, or we may correctly forego treating a patient who is not suffering from respiratory failure $(\neg A, \neg H)$. The expected object-level utilities of action $[u(A)]$ and of no action $[u(\neg A)]$ in terms of the probability of respiratory failure, P(H), are described by the following equations:

$$u(A) = p(H)u(A, H) + p(\neg H)u(A, \neg H)$$

$$u(\neg A) = p(H)u(\neg A, H) + p(\neg H)u(\neg A, \neg H)$$

The lines described by these equations intersect at a probability of *11* denoted *p*.* The desired action (the decision with the highest expected utility) changes as the utility lines cross at ;P*. A utility analysis dictates that a patient should not be treated unless a decision maker's belief in the truth of *H* is greater than *p*.*

4.2 Decisions About Computation

Let us now integrate explicit knowledge about the process of reasoning into the decision problem. In answer to a query for assistance, our automated reasoner must propagate observed evidence about the patient's symptomology through a complex belief network. The results of an approximate probabilistic-inference scheme may be a probability distribution over a *final* probability. This probability is the value that a computer will calculate from a belief network, given sufficient time to finish its computation. Assume that our reasoner may apply one of several incremental-refinement algorithms that can iteratively tighten the distribution on the probability of interest over time. We wish the system to make a rational decision about whether to make a treatment recommendation immediately, or to defer its recommendation and continue to reason, given its knowledge about the costs of time needed for computation.

4.2.1 Costs of Inference-Based Delay

The example of a patient gasping for breath, facing the risk of a long hospitalization or a cardiac arrest depending on our decision, poignantly demonstrates the salience of reasoning-resource constraints in a high-stakes situation. Sc far, we have considered the utilities of alternative outcomes to be independent of time. Assume that the utility of treating a patient in respiratory failure depends on how long the patient has been in failure. Assume, also, that the initial presentation of respiratory symptoms occurs in the presence of the reasoner and that analysis of the problem begins at this time, t_o. We represent the cost of delaying treatment, when that treatment is needed, by considering a continuum of mutually exclusive decisions to treat at different times, A(i). where $t = t_o + \Delta t$. A cost function can capture

the decay of utility of action with time. At some time t, the utility of acting in the presence of respiratory failure reverts to the utility of not acting at all. We substitute the static utility equation for *u(A),* defined previously, with a time-dependant equation:

$$u[A(t)] = p(H)u[A(i),H]+p(-,H)u(A,->H)$$

where u[A(t), *H]* reverts to $u(\neg A, H)$ as some function of time[2]. In this example, we assume that delay of action will not affect the utility of a patient that does not require the intervention. With the time-dependent utility function, our p* threshold will change with time.

As indicated by the network in the upper portion Figure 1, a more complete representation of the respiratory decision problem includes knowledge about the costs and benefits of applying different inference strategies. This influence diagram represents the metareasoning problem. The node labeled U_o in the metareasoning network is just the value node from the object-level decision problem represented at the bottom of Figure 1. Rather than seek to optimize the object-level value, our agent's goal is to optimize the utility associated with the value node in the metareasoning problem, labeled U_c. As demonstrated by the relationships among propositions in the metareasoning problem, U_c is a function of the object-level value and the inference-related cost, Ui, which in turn depends on computational delay, time availability, and the context. The integration of inference-related and object-level utility allows agents to treat decisions and outcomes regarding the control of reasoning just as it does decisions about action in the world.

4.2.2 Reflection About Future Belief

Our agent's attention is centered on the calculation of *p(H),* the *probability of respiratory failure.* We define $<\phi$ to be the probability that the agent would compute if it had sufficient time to finish its computation. That is, ϕ is value of *p(H)* that the reasoner will report after complete computation. At the present moment—before the inference is completed—our automated reasoner may be uncertain about what the value of ϕ will be. The current uncertainty is described by some probability distribution over ϕ. We denote the uncertainty about ϕ *at the present moment* by $p(\phi)$. Although this distribution can change with reasoning, investigators [Howard, 1970] have shown that the belief a decision maker should use for decision making, if she has to act immediately, is the mean of *p{),* denoted by $< \phi >$. After spending additional time *t* on inference about ϕ, our reasoner may have a new distribution over ϕ, denoted by $p_t(\phi)$.

An automated reasoner may have useful knowledge about how a distribution over a belief—and thus how the new mean of the distribution -will change with additional computing. An important class of knowledge about ϕ is of the form, $p(p_t(\phi))$. This measure refers to belief *at the present time* about the likelihood of alternative belief distributions over ϕ that might be generated after computation for additional time *t.* This notion is central in reflection about the value of initiating or continuing decision-theoretic inference, as opposed to that of acting with the current best decision.

Expected Value of Perfect Computation Suppose that, after thinking for only a few milliseconds, an automated reasoner has generated a probability distribution over ϕ. We first introduce the *expected value of perfect computation* on ϕ, denoted by $EVPC_\phi$. The EVPC_ϕ may be viewed as the value of instantaneous complete computation of the target probability in a decision setting. Instantaneous complete thinking would collapse the current probability distribution over ϕ into an impulse. Given the current probability distribution $p(\phi)$ over ϕ, we define $EVPC_\phi$ as follows:

$$\mathrm{EVPC}_\phi = \int_\phi p(\phi) \max_D u_o[D(\phi)]\, d\phi - \max_D u_o[D < \phi_o >]$$

where $\max_D u_c[D < \phi_o >]$ is the utility, associated with the best decision *D,* based on taking an immediate action using the current mean belief, $< \phi_o >$. This measure tells us that the value of computing the final answer is just the difference in utility between the current best action and the summation of future best actions weighted by the probability of different final beliefs.

Belief About Changes in Belief Real-world computers rarely deliver the full expected value of perfect computation on difficult problems because they must expend valuable resources in the reasoning process. Assume that our agent in the intensive-care unit, faced with determining the probability that our elderly patient is in respiratory failure, has incomplete knowledge about what $p(\phi)$ will be at some future time t, which we refer to as $p(p_t(\phi))$. For example, the system may have a probability distribution over the future bounds on ø with additional computation. Such knowledge may have been acquired through an empirical analysis of a network in addition to an upper bound that has been proved theoretically. Our reasoner could apply this type of knowledge by considering the EVC(t) based on the information about probability distributions over $p(\phi)$, obtained with computation for an additional time *t,* as

$$\mathrm{EVC}(t) = \int_{p_t(\phi)} p(p_t(\phi)) \int_\phi p(\phi) \max_D u_c[D(\phi), t]\, d\phi\, dp_t(\phi) - \max_D u_o[D(< \phi_o >)]$$

That is, we sum over the new probability distributions on ϕ expected at time *t,* weighted by the *current* belief, $p(p_t(\phi))$, that thinking until *t* will lead to each of the revised distributions. In terms of the mean, $< \phi_t >$ of the future distributions, $p_t(\phi)$,

$$\mathrm{EVC} = \int_{p_t(\phi)} p(p_t(\phi)) \max_D u_c[D(< \phi_t >, t)]\, dp_t(\phi) - \max_D u_o[D(< \phi_o >)]$$

When, for all *t,* the cost of computation, embodied within our comprehensive utility function, becomes

In this domain, we could capture the cost of delay with a stochastic model describing the probability of a cardiac arrest as a function of the time we delay therapy; cost models can be useful summaries of the utility of a large number of outcomes.

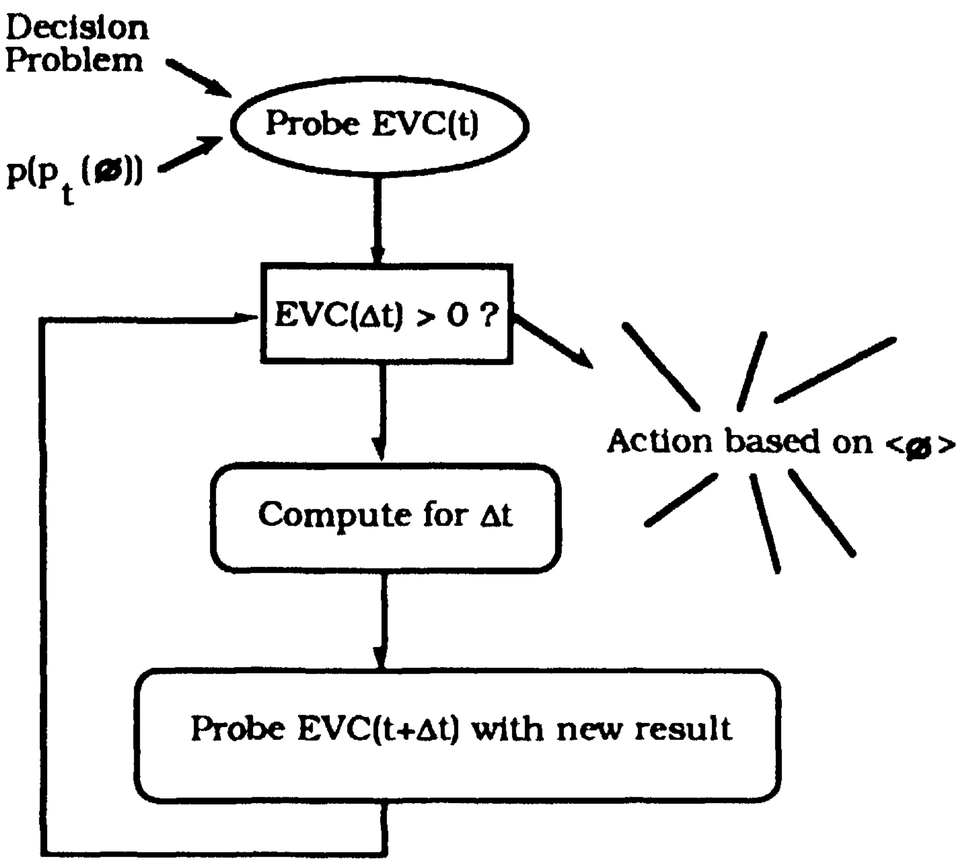


Figure 2: In reasoning about the value of continuing to reflect about belief, versus that of taking immediate action, an EVC-evaluation module considers the decision problem and the probability distribution over future probability distributions [p(pt (ø))] that may be generated with the allocation of computational resource.

greater than the benefit of computing (EVC < *O),* an agent should cease reflection and act. The EVC formula can be used to study the value of alternative inference schemes. Of course, there can be significant overhead in the real-time application of an EVC-based control strategy. Thus, a central goal of research on decision-theoretic control is to identify tractable solutions to the EVC evaluation problem. Alternatively, offline analysis and compilation of control strategies may be useful in situations where the complexity of meta-analysis limits the gains of real-time decision-theoretic control.

Analogous value-of-computation approaches can be used to valuate and control other problem classes. For example, we can use an *EVC(t)* calculation for controlling the nature and extent of a search or sort problem; we associate a cost with the time required to expand another node in a tree, or to perform a set of tests and swaps in a partially sorted file, and consider a probability distribution over the expected object-level gains, given the allocated time. The development of tractable EVC approximations for these and other problems, make possible useful normative control through iterative testing of the value of continuing to reason.

5 Value of Probabilistic Bounding

We have pursued tractable solutions to the EVC through examining parameterized families of distributions. For example, we have explored the use of the EVC approach to control probabilistic bounding methods. Assume that our automated reasoner has, with some initial amount of computation, computed upper and lower bounds on *ø),* with an upper bound at *b* and lower bound at a. If our reasoner does not have any information about where ø is—except that the final computed result will be between the current bounds—then it is reasonable to assume a uniform distribution over ø between the bounds. A uniform distribution within bounds is consistent with an agent being ignorant of the final belief, except for the bounds information. Detailed knowledge about convergence could change this distribution. Let us focus on the value structure of assuming uniformity at both current and future distributions about belief. We denote the problem by EVC/BU, the expected value of computation for a bounding algorithm given an assumption of uniformity.

An agent may have useful knowledge about $p_t(ø)$ *without* having information about how the mean $< ø_t >$ will change, except for knowing that ø will be constrained to tighter bounds. As an example, a system could make use of certain or uncertain knowledge about the *rate* of bounds convergence to valuate a decision to continue to compute. We have analyzed how a system can apply knowledge that the bounds on the belief for a node in a belief network will converge at a rate dictated by a fraction, C, which, when multiplied by the current bounds interval at t_o, dictates the interval at *t.* That is,

$$int_t = C * int_{t_o}$$

where *int* is just the interval, or the difference between the upper and lower bounds. If we were uncertain about the convergence, we would have a probability distribution over this convergence fraction.

We have applied the EVC equation to the bounding problem, considering future distributions expected with additional computation. Given a convergence fraction that allows us to calculate the future bounds, we must consider all possible configurations of the new bounds given the current constraints. As we sweep the expected future interval over the current interval, the mean of the future distribution sweeps between positions within the current bounds. When the mean is above p*, we sum over the utility of acting for all states of belief greater than that threshold; when the mean is below *p**, we similarly consider the utility of not acting. Given our current bounds and a convergence fraction, we sum the utilities of the best decision at the future means and subtract the utility of the best action without additional computation. Solving the uniform distribution case for different possible p* boundary conditions yields functions that report the EVC as a function of (1) the utilities for each of the four outcomes, (2) the current bounds on *ø,* (3) a function describing the expected convergence of bounds (e.g., aC, with time, and (4) the cost of delay. Under uncertain performance, a rational agent's reflection based on the EVC formalism involves the interlacing of probes for positive EVC(t) and continued inference. Computation should continue until action is indicated by a non-positive EVC. This volley of reflection and inference is demonstrated in Figure 2.

5.1 Partial Characterization of Inference

We have experimented with decisions about computation and action within alternative utility contexts. We have

particularly explored the behavior of recently-developed graceful approximation methods for probabilistic inference. These strategies include a flexible variant of Pearl's method of conditioning [Pearl, 1986], called *bounded conditioning* [Horvitz et al., 1988b].

In the method of conditioning, a multiply connected network is reformulated to a set of singly connected networks by locating a set of nodes that break cycles. The complete set of cycle-breaking nodes is called the *loop cutset.* The nodes of the loop cutset are instantiated with each possible value (or combination of values), and the resulting joint probabilities of each *instance* are calculated as prior probabilities of the instantiated variables. Algorithms for solving the singly connected network subproblems can be applied to the solution of each network instance. In bounded conditioning, instances are analyzed in order of their expected contribution to the tightening of bounds. The instances are sorted according to their prior probability, and are solved in sequence. A bounding calculus generates logical bounds on the final probability of interest by considering the maximum and minimum contributions of the unexplored subproblems.

We applied bounded conditioning to several random networks as well as to a belief network describing probabilistic relationships among findings and pathophyisologic states in an intensive-care unit.[3] The structure of this belief network is captured by the graph in the middle of Figure 1. The network consists of 37 multiply connected nodes. We studied the performance of several loop cutsets for this network. A sample loop cutset consists of 5 nodes that leads to 144 different singly connected-network problems.

We sought to characterize the refinement of bounds with additional computation. Our analyses focused on updating belief in the intensive-care network with single pieces of evidence. We found that the convergence of the bounds could by approximated by an exponential decay of the size of the interval with time. This convergence was modeled approximately by the function

$$int = e^{-k(t+1)}$$

Additional discussion of bounded conditioning, including analysis of the basis for such convergence, is found in [Horvitz et al., 1988b]. As an example, the convergence of a typical update in the network is captured by an exponential decay with an approximate half-life of 36 seconds. That is, after 36 seconds of analysis by a Motorola-68020-based computer, running at a 17 MHz clock rate, the bounds converge to one-half of their original bounds. At 72 seconds, the bounds are halved once again to an interval of approximately 0.25. This convergence is modeled by the exponential decay with $k = 0.02$. The convergence is displayed in Figure 3.

This convergence information can be used to calculate an EVC associated with continuing to apply the bounding algorithm. Evaluating the EVC within our testbed intensive-care belief network has shown, for sample updates and associated sets of utility estimates, that a p* decision threshold can be crossed well before the computation of final belief. Experimentation with rational metareasoning to select among alternative inference strategies, and to control the length of time that they are applied is continuing on a variety of belief networks and decision contexts.

[3]This network, called ALARM, was constructed by Ingo Beinlich [Beinlich et al., 1989].

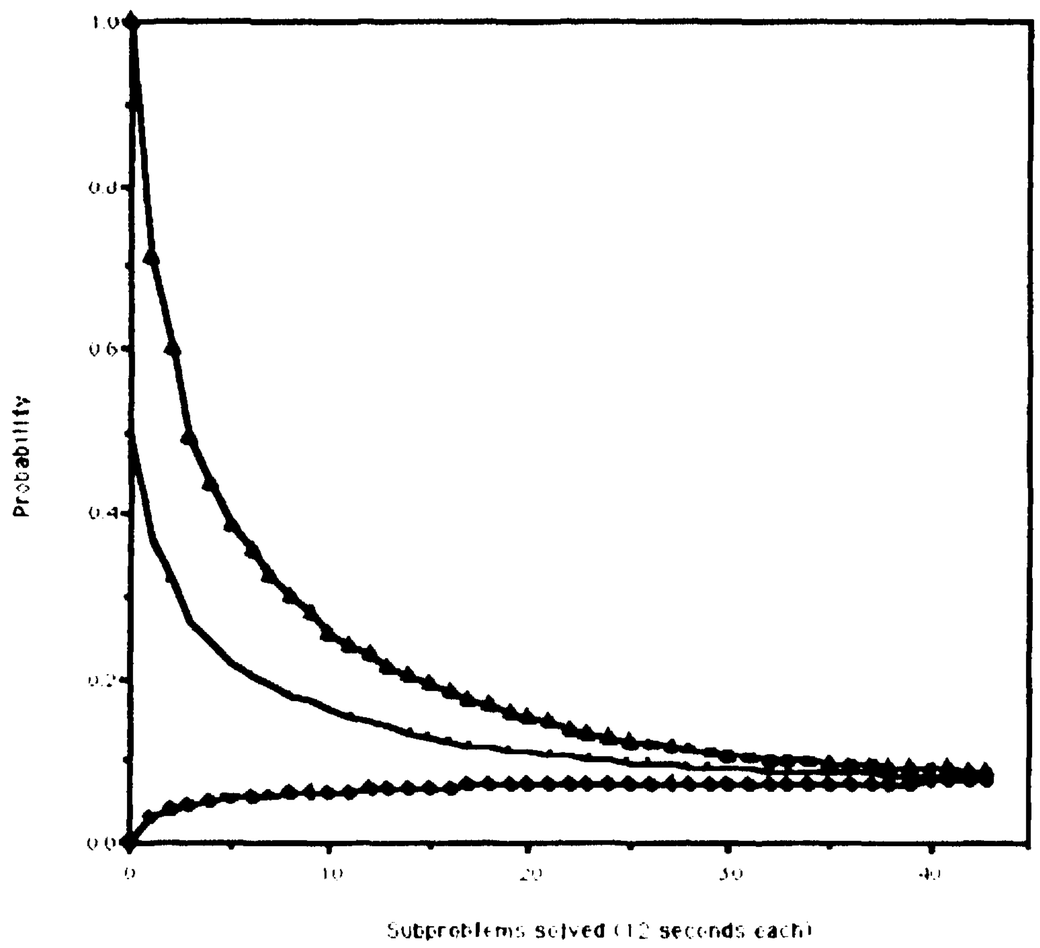


Figure 3: The application of the bounded-conditioning method to the intensive-care unit belief network problem. The graph shows the convergence of the upper and lower bounds, and the mean of the approximation (center curve), to a probability of interest as additional inference subproblems are solved.

5.2 Acquisition of Control Knowledge

Our formalism for the calculating the value of probabilistic bounding operates on knowledge about convergence on bounds. We have performed theoretical analysis of worst-case performance of bounded conditioning. We have also recorded empirically derived partial-characterization information. Clearly, an agent could benefit by continually bolstering its knowledge about partial characterizations with extensive empirical study of problem-solving trajectories during idle-time. A component of our research focuses on an offline analysis of the performance of reasoning strategies of different networks. The analyses are aimed at capturing useful partial characterization of the expected performance of different strategies by performing Monte Carlo simulation to generate plausible patterns of evidence, and summarizing and storing a set of performance indices. For example, we are interested in the convergence of bounds in response to a state of evidence. This information can be extremely useful to a control reasoner that is attempting to valuate the EVC for a set of competing solution strategies. We are researching the automated acquisition of partial characterizations of stategy performance within the intensive-care unit application area.