Methods of Information in Medicine

E. H. Herskovits, G. F. Cooper

Algorithms for Bayesian Belief-Network Precomputation

Section on Medical Informatics,
Stanford University,
Stanford CA, USA

Abstract: Bayesian belief networks provide an intuitive and concise means of representing probabilistic relationships among the variables in expert systems. A major drawback to this methodology is its computational complexity. We present an introduction to belief networks, and describe methods for precomputing, or caching, part of a belief network based on metrics of probability and expected utility. These algorithms are examples of a general method for decreasing expected running time for probabilistic inference.
We first present the necessary background, and then present algorithms for producing caches based on metrics of expected probability and expected utility. We show how these algorithms can be applied to a moderately complex belief network, and present directions for future research.

Key-Words: Expert Systems, Algorithms, Computer-Assisted Diagnosis, Probability Theory

1. Introduction

A particular graphical representation of probabilistic relationships among variables, called a *Bayesian belief network,* has been independently defined and explored by several researchers. This independence is manifested in the plethora of synonyms for Bayesian belief networks [1], such as *causal nets* [2, 3], *probabilistic causal networks* [4], *influence diagrams*[1] [5, 6], and *causal networks* [7]. We shall use the term *belief networks* in this paper.

Belief networks show promise as a powerful representational framework for constructing expert systems that reason under uncertainty [8, 9]; they provide platforms for knowledge acquisition [10–12] and for probabilistic inference [7, 13, 14]. This representational power is particularly important in domains in which conclusions, intermediate states, and evidence are related by complex interactions, making knowledge acquisition and knowledge-base maintenance difficult.

Algorithms for belief-network inference, such as those developed by Pearl [13, 14] and Lauritzen and Spiegelhalter [7], allow the user to set nodes to specific values, after which these algorithms compute the posterior distributions over the remaining nodes, and thus provide a simple yet general mechanism whereby the user may enter evidence and determine the ensuing implications. Despite the intuitive appeal of this inference paradigm, the run-time complexity of general belief-network inference may be too great for solving many complex problems in a practical amount of time; in fact, Cooper has shown that probabilistic inference using belief networks is NP-hard [15].

In an attempt to increase the tractability of belief-network inference, researchers have focused their attention on developing approximate and special-case algorithms for belief-network inference [1, 4, 10, 16, 17]. A special-case algorithm might prove useful, for example, where a relatively small subset of possible values of evidence nodes in a belief network covers a large proportion of all possible values typically encountered when using the network.

One special-case algorithm, that of precomputing results for future use, has been proposed as a foundation for planning systems [18]. For belief networks that represent a highly skewed distribution of joint probabilities of evidence, storing a small number of precomputed cases to capture a large proportion of the likely uses of the network may lead to a significant increase in the speed of inference in many cases. We describe here the foundations of belief-network inference, and report the results of a set of algorithms that cache (precompute and store) a small subset of a belief-network cases to decrease the expected running time for inference.

2. Belief Networks

A belief-network structure is a finite directed acyclic graph in which nodes represent the variables of interest, and arcs from parent nodes to child nodes represent a probabilistic

[1] A belief network is a special case of an influence diagram. An influence diagram represents decision alternatives and outcome values in addition to the probabilistic associations among variables found in a belief network.

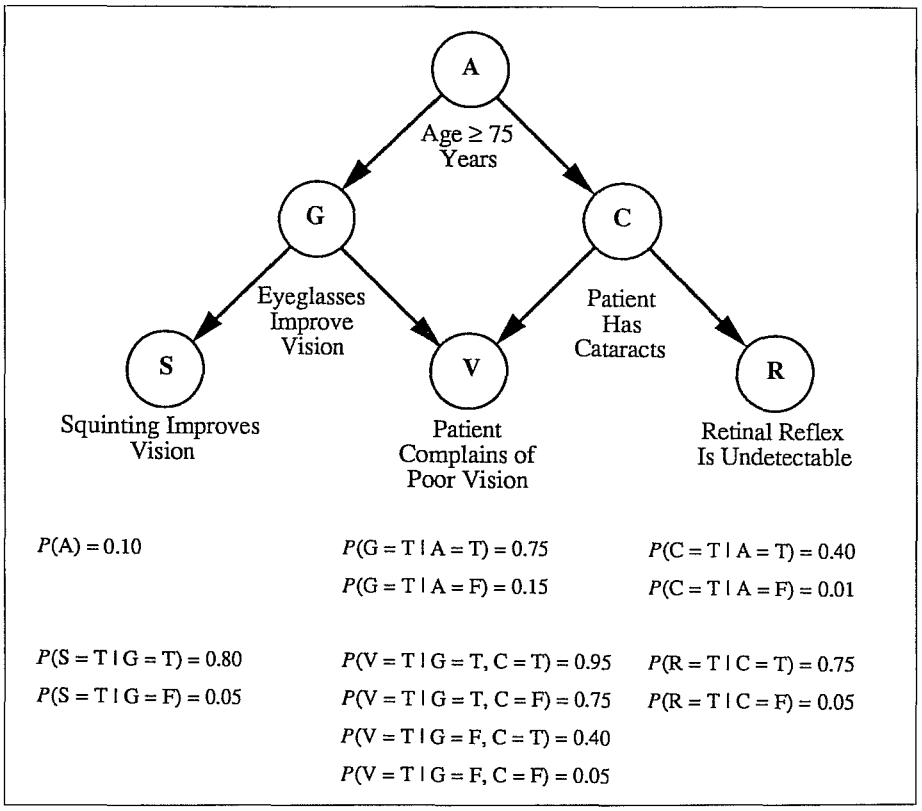

Fig. 1 An example of a belief network.

association between the child and its parents. Probabilities are attached to arcs in a belief-network structure; these probabilities capture the uncertainty inherent to the relationships among the variables. In particular, for each node x_i with a set of parents π_i, there is a conditional probability distribution $P(x_i \mid \pi_i)$; for each x_i without parents, there is a prior probability distribution $P(x_i)$.

Fig. 1 contains a belief network from the domain of visual disturbances; it has been greatly simplified for the purpose of illustration. For conciseness, some probabilities are not shown in the Figure, but all can be computed readily from those presented. For example,

$P(G = \mathrm{F} \mid A = \mathrm{T}) = 1 - P(G = \mathrm{T} \mid A = \mathrm{T}) = 1 - 0.75 = 0.25.$

Belief-network semantics dictate that *patient complains of poor vision* and *retinal reflex is undetectable* be conditionally independent given *patient has cataracts;* that is, once you know whether the patient has cataracts, knowing whether a retinal reflex is undetectable tells you nothing about whether the patient complains of poor vision. Were this independence assumption invalid, other interrelationships among the variables would require representation by additional arcs in the belief network. In the example, *patient complains of poor vision* is associated with *eyeglasses improve vision* and *patient has cataracts,* and these associations are symbolized by the arcs from *eyeglasses improve vision* and from *patient has cataracts* to *patient complains of poor vision.*

If a variable is associated with several other variables (shown as parent nodes in the graph), the person constructing the belief network must specify the probabilities of each variable's values given each possible combination of that variable's parents' values. For example, in Fig. 1, the probabilities for $V = \mathrm{T}$ are specified for each possible combination of values for this node's parents, G and C. Although tables or lists are usually used to express probabilistic relationships among variables, closed-form functional probabilistic relationships can also be used.

Conditional probabilities in belief networks can be used as "if–then" rules in the construction of probabilistic expert systems. In this interpretation, a rule in a belief network is a conditional probability of the form $P(x_i \mid \gamma_1, \gamma_2, \ldots \gamma_n)$, where x_i and $\gamma_1, \gamma_2, \ldots \gamma_n$ are variables with assigned values. Each variable has an associated *rule family,* which is the collection of conditional probabilities for each possible combination of values that this variable and its parents can assume. For example, in Fig. 1, the probability

$P(V = \mathrm{T} \mid G = \mathrm{T}, C = \mathrm{T}) = 0.95$

corresponds to the rule "If $G = \mathrm{T}$ and $C = \mathrm{T}$ then $V = \mathrm{T}$ (with probability 0.95)." There are three other rules in this rule family corresponding to the three other probabilities below

$P(V = \mathrm{T} \mid G = \mathrm{T}, C = \mathrm{T})$

in Fig. 1. In fact, there are a total of eight rules in this family, but half of them are redundant, in that they can be derived from those rules shown in the Figure.

The prior and conditional probabilities explicitly represented in a belief network are sufficient for computing any probability of the form $P(\Theta \mid \Phi)$, where Θ and Φ are members of the power set of this belief network's variables. The key feature of the belief-network paradigm is its explicit delineation of conditional independence among variables, which in turn decreases the number of probabilities required to capture the full joint distribution. We shall now formally define the conditional independence that is expressed in the structure of a belief network. We need the following auxiliary definitions:

- U is the set of all n nodes in a given belief network.
- A node X_j is a direct predecessor of node X_i if there is an arc form X_j to X_i.
- π_i is the set of all direct predecessors of X_i.
- A node X_k is a successor of node X_i if there is a directed path from X_i to X_k.
- S_i is the set of all successors of X_i.
- S_i' is the complement of S_i, excluding X_i itself. Thus, $S_i' = (U \setminus S_i) \setminus \{X_i\}$, where $\setminus$ is the set-difference operator.
- Q_i is the power set of S_i'.

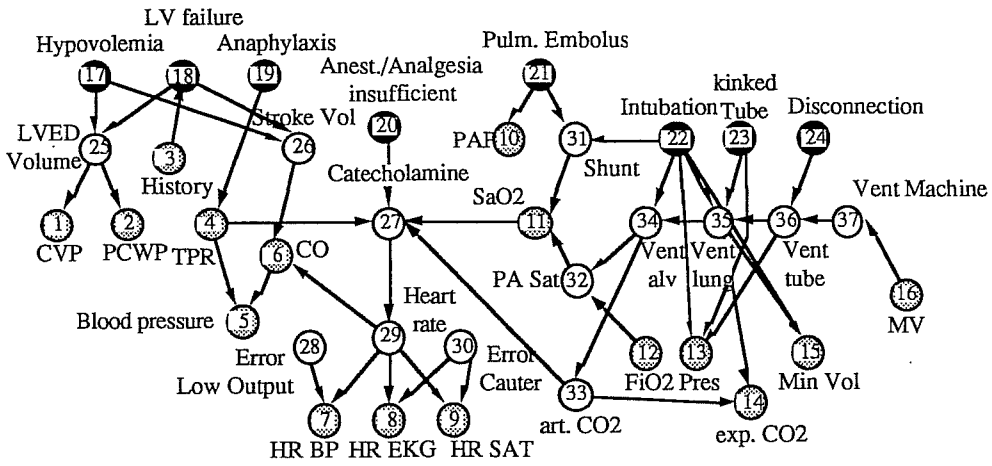

Fig. 2 The ALARM network representing causal relationships, with diagnostic (●), intermediate (○) and evidence (◎) nodes. Abbreviations are explained in Table 3.

As an example, suppose that $X_i = G$ in Fig. 1. In this case,

$U = \{A, G, C, V, S, R\}$,
$\pi_i = \{A\}$,
$S_i' = \{A, S\}$,
$S_i' = \{A, C, R\}$, and
$Q_i = \{\{\}, \{A\}, \{C\}, \{R\}, \{A, C\}, \{A, R\}, \{C, R\}, \{A, C, R\}\}$.

In general, for a given node value x_i, a belief network expresses the following conditional independence relation:

$$\forall q \in Q_i : P(x_i \mid \pi_i \cup q) = P(x_i \mid \pi_i) \quad (1)$$

This relation states that, if the values of the direct predecessors of X_i are known with certainty, then the probability of each value of X_i is conditionally independent of any subset of the nodes in S_i' – that is, of any subset of the nodes that are not successors of X_i. For instance, in the previous example, if we know just the value of A then we can determine the probability that $G = T$, regardless of the values of C or R. As another example of conditional independence, the absence of an arc from node A to node V is a statement that the value of node A is conditionally independent of the value of node V, given the values of nodes G and C. In general, the absence of an arc from a node X_j in S_i' to node X_i indicates that X_i is conditionally independent of X_j, given just the values of the nodes in π_i.

Shachter [19] has shown that, given the conditional-independence constraints expressed in equation (1), a belief network represents the full joint-probability space over the n variables in the network by way of the following equation:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \pi_i) \quad (2)$$

Therefore, we can compute the joint probability of any instantiation[2] of all the variables in a belief network as the product of exactly n probabilities.

For example, consider the following calculation of the joint probability of all the variables in Fig. 1 with values instantiated to true (T):

$$P(A = T, G = T, C = T, S = T, V = T, R = T) = P(A = T) \times P(G = T \mid A = T) \times P(C = T \mid A = T) \times P(S = T \mid G = T) \times P(V = T \mid G = T, C = T) \times P(R = T \mid C = T) = 0.10 \times 0.75 \times 0.40 \times 0.80 \times 0.95 \times 0.75 = 1.7 \times 10^{-2} \quad (3)$$

We can recover the complete joint-probability space from the belief-network representation by calculating the joint probabilities that result from every possible instantiation of the n variables in the network. Instead of representing all 2^n probabilities in the joint space[3], however, we need represent only $P(x_i \mid \pi_i)$ for each node value x_i in the network; this representation may require a total of many fewer than 2^n probabilities. For example, in Fig. 1, only 13 probabilities (rather than 64) are needed to represent the six-node belief network. Algorithms have been developed that directly manipulate the probabilities in a belief network to perform inference, obviating the need for explicit reconstruction of the underlying joint-probability space.

In summary, a belief network graphically represents the probabilistic relationships among concepts in a knowledge base. This representation usually reduces the number of probabilities needed (relative to the cardinality of the full joint-probability space), and may also reduce the computational time complexity of probabilistic inference for some networks. Nonetheless, for belief networks with densely interconnected nodes, inference time can be prohibitively long. We now describe the implementation and results of a method for precomputing part of a belief network to increase the expected speed of inference.

[2] The term *instantiated variable* denotes a variable with a known, assigned value. For example, an instantiated propositional variable would have an assigned value of either true (T) or false (F).

[3] In fact, only $2^n - 1$ probabilities are necessary for a binary network, as they determine the value of the remaining probability.

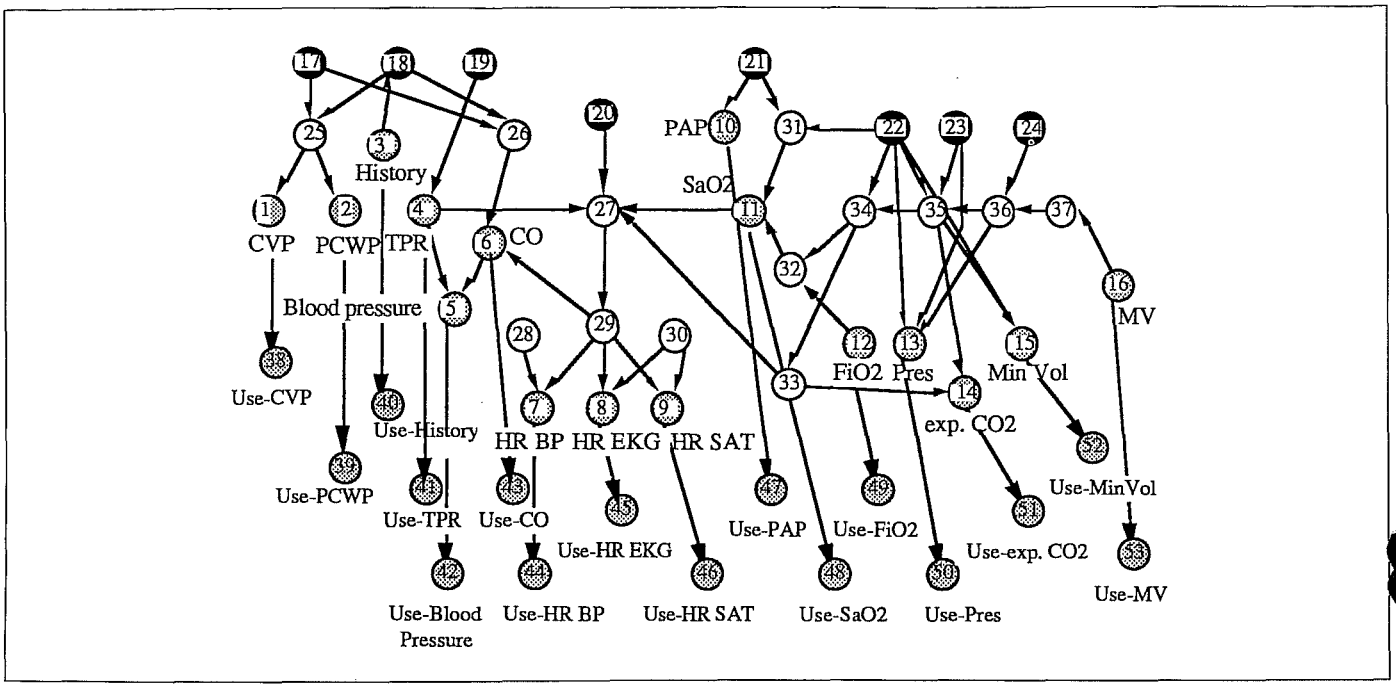

Fig. 3 The NetUse network representing causal relationships, with diagnostic (●), intermediate (○) and evidence (◎) nodes. Numbering for nodes 1–37 is identical to that in Figure 2; labels for non-evidence nodes have been omitted for clarity. See Table 3 for labels and for explanations of abbreviations. The prefix Use before a node indicates that the node models the probability that a user will report a manifestation given that that manifestation has been observed. (After [19], with permission.)

3. Methods

3.1 The Domain

Precomputations performed on the ALARM belief network [20] served as our benchmarks for evaluations, although the algorithm presented here could be applied to any belief network. The ALARM network, shown in Fig. 2, is a prototype knowledge base that models a small subset of the medical conditions that anesthesiologists encounter in an operating room; the corresponding graph has 37 nodes, 46 arcs, and approximately 10^{17} elements in its joint distribution. Eight of the nodes represent a diagnosis, such as *pulmonary embolus* (a blood clot obstructing the circulatory system of the lungs). All these eight nodes are roots of the network; that is, they have no parents. Sixteen of the nodes represent evidence, such as *blood pressure*, acquired during patient evaluation; these nodes have no children. The remaining 13 nodes represent intermediate variables, such as the *catecholamine* level (a measure that reflects the degree of physiologic and metabolic stress the patient is experiencing); these nodes have both parents and children.

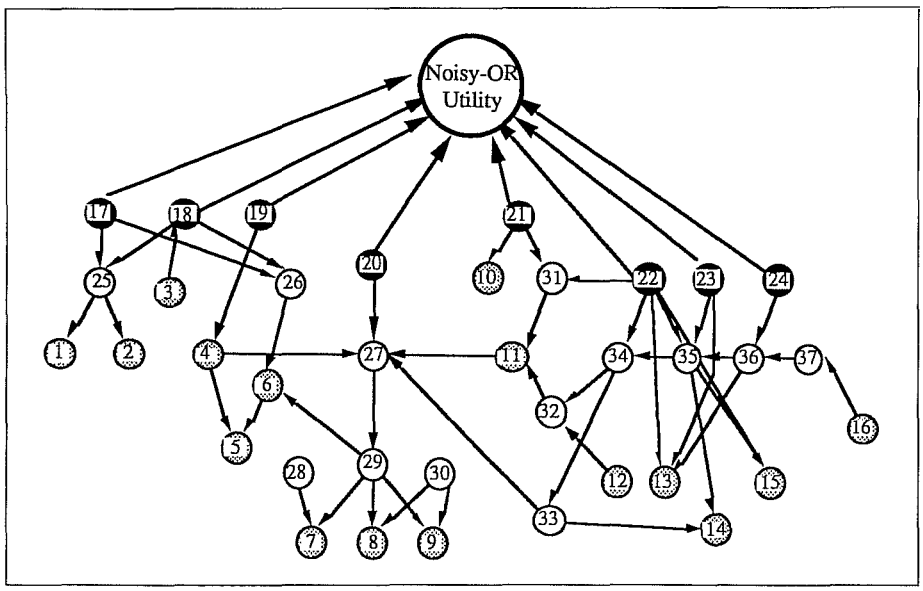

Fig. 4 The CaseUtility network representing causal relationships, with diagnostic (●), intermediate (○) and evidence (◎) nodes, and the noisy-OR-utility node associated with the diagnoses. Numbering and labels for nodes 1–37 are identical to those in Figure 2; the labels have been omitted for clarity. See Table 3 for labels and for explanations of abbreviations. (After [19], with permission.)

3.2 Cache-Construction Algorithms

Three caching methods were implemented: The BASIC, NETUSE, and CASEUTILITY algorithms[4]. In con-

[4] We implemented the algorithms on an Apple® Macintosh II™ with 2 megabytes of random-access memory, a 20-megabyte hard disk, and a Motorola® 68881 mathematics coprocessor. We used Lightspeed Pascal™, version 2.0, for software development.

structing the caches, we assumed that the developer would have access to ample computer time; thus caches would be limited strictly by available memory. We further assumed that inference would be performed in settings in which memory is not limited, but in which the time to completion of inference is critical. Such would be the situation in an acute-care setting, where timely diagnoses may make the difference between effective treatment and morbidity.

In implementing these algorithms, we further assumed that the user would be interested only in instantiating evidence nodes and in observing diagnosis nodes, a common application of diagnostic expert systems. In this context, we defined a *case* as a set of evidence and diagnosis, and considered the joint probability of an evidence set to be a proxy for the relative likelihood that a user would enter this evidence set during a consultation; this requirement was relaxed in the NETUSE version of the algorithm. These algorithms have in common three modules: a case generator, a cache builder, and a network evaluator, which we will present as the components of the BASIC algorithm.

The ideal case generator would run without user intervention and would generate cases in order of likelihood. One readily apparent candidate algorithm is based on the exhaustive enumeration of all possible cases and then sorting on the joint probability of the evidence set. For a belief network with a large number of possible evidence sets, exhaustive enumeration of the evidence feature space is impractical; a heuristic method must be employed to decrease the size of the search space. One such method is Henrion's logic-sampling algorithm [16], which generates cases by instantiating root nodes based on the nodes' prior probabilities, and then instantiating nodes whose parents have all been instantiated, until all the nodes have been instantiated. For example, in the belief network in Fig. 1, the algorithm would instantiate node A first; there is a 10% chance that this node would be set to True. Next, nodes G and C would be instantiated; assuming A were set to False, G and C would be set to True with probabilities 0.15 and 0.01, respectively. Finally, nodes V, S and R would be instantiated stochastically, based on the values of G and C. Although this method represents a stochastic sampling of the space of possible cases, it has two desirable properties. First, it has a running time that is polynomial in the size of the belief network and in the number of cases generated. Second, it is expected to generate evidence sets that have relatively high joint probabilities. In addition, the more likely cases tend to be generated earlier, so this algorithm performs an approximate sort on cases by their respective probabilities of occurring.

The second module constructs the cache by comparing each generated case to those already in the cache. If the case is found to be new, this algorithm determines the joint probability of the case's evidence using the third module, an implementation by Suermondt of the Lauritzen-Spiegelhalter algorithm [7]. The marginal posterior-probability distributions over the diagnosis nodes are also determined with the Lauritzen-Spiegelhalter algorithm, and the values of the evidence nodes, the joint probability of the evidence set, and the marginal posterior-probability distributions for the diagnosis nodes are stored in the cache.

There are several possible termination criteria for the BASIC algorithm, including the number of cases generated, and the time spent on cache generation. We incorporated both of these criteria in our implementation.

The NETUSE and BASIC algorithms are identical except for the structure of the belief network; to construct the NETUSE version of ALARM, we defined additional nodes and acquired from our expert the corresponding conditional probabilities that answered the question: "Given that the patient has evidence x, what is the probability that the user will instantiate the belief-network node corresponding to x?" For example, x could be the evidence that *squinting improves vision*, we would then acquire from our expert the probabilities that a user would instantiate the variable *squinting improves vision*, conditioned on whether the patient's vision improves with squinting. The NETUSE version of the ALARM network is shown in Fig. 3. Given that the *blood pressure* is low, for example, this node will be instantiated as low, with probability 1.0, since we assume that, in the operating room, where the ALARM network would be used, the user always has access to (and always reports) the patient's blood pressure.

In contrast to the NETUSE algorithm, which models expected use of a belief network and gives an estimate of how a user will instantiate the nodes, the CASEUTILITY algorithm supports utility-driven case generation: It generates those cases for which the user places a high utility on rapid diagnosis. The CASEUTILITY algorithm thus precomputes cases on the basis of their expected utilities rather than of their expected likelihoods. An example of the ALARM network enhanced with a utility node is shown in Fig. 4. To simplify implementation and utility assessment, we implemented a noisy-OR utility model [21], which is an analogue of the noisy-OR gate used to simplify probabilistic knowledge acquisition and inference [22]. Let $P(d_i \mid C)$ be the probability of the ith diagnosis being true, given the current case C, and let t be the amount of time required to compute $P(d_i \mid C)$ for all diagnoses. We assessed one utility for each diagnosis in the network by asking the question: "What is the probability that a patient will die as a direct result of his physician learning of the system's probability distribution for each d_i only after t minutes?" These utilities are assessed only for the presence of conditions, not for the absence of conditions, since the probability of dying from d_i, given that d_i is false, is assumed to be 0 (that is, $U_t(d_i = False) = 0$ for all d_i and for all t within the time horizon of the system's application). Furthermore, we assume that:

- Diagnoses occur at discrete points in time.
- For any case, the availability of the system's probability distribution over the diagnoses always results in the patient's physicians making the correct diagnosis.

Table 1 Utilities for diagnosis nodes, t = 0 and 2 min*

Diagnosis (D_i)	Utility $U_0(D_i)$	$U_2(D_i)$
Hypovolemia	0.0020	0.0020
LV failure	0.0050	0.0050
Anaphylaxis	0.0050	0.1000
A/A insufficient	0.0001	0.0010
Pulmonary embolus	0.7000	0.7500
Intubation		
Extubated	0.0010	0.0050
One-sided	0.0000	0.0010
Kinked tube	0.0001	0.0005
Disconnection	0.0001	0.0005

* LV failure: left-ventricular failure; A/A insufficient: insufficient anesthesia or analgesia.

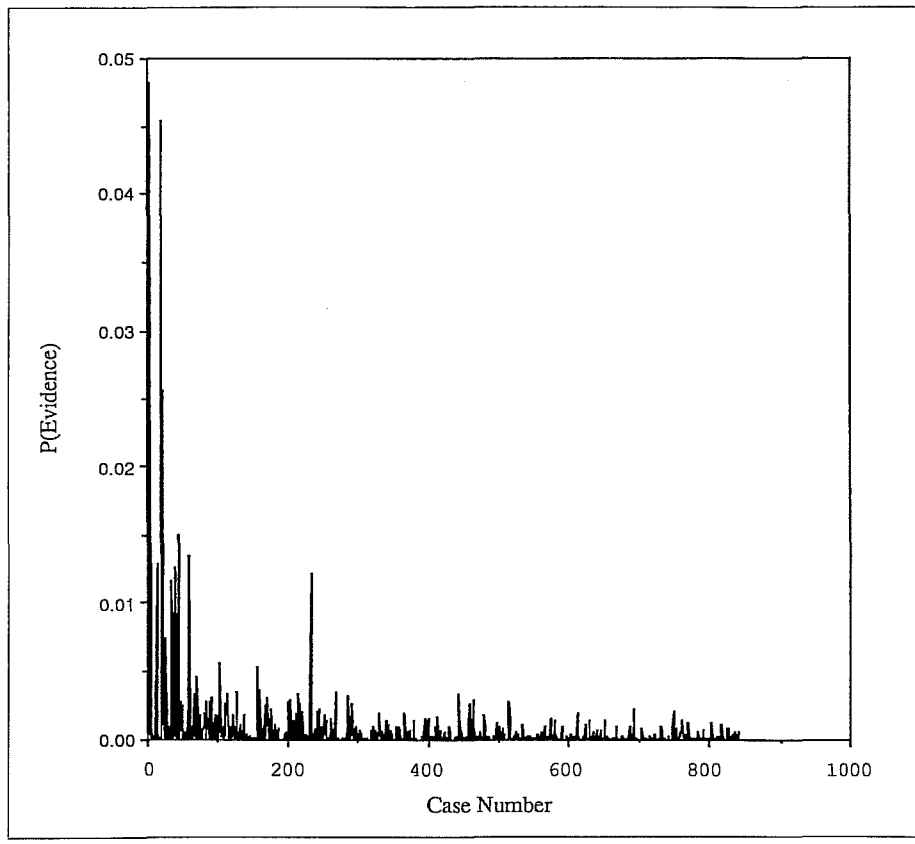

Fig. 5 Joint probability of the evidence: Basic algorithm.

- The physician will immediately render optimal treatment once the correct diagnosis is known.
- For any case, the unavailability of the system's probability distribution over the diagnoses results in the physician treating the patient based on evidence alone.
- Once one or more diagnoses occur, those diagnoses, and only those diagnoses, persist until they have been appropriately treated.
- Utility is a function only of diagnosis-related mortality, and not of diagnosis-related morbidity.

Clearly, these assumptions are strong; we can relax them by additionally modeling disease evolution, the expected decision-making behavior of the physician given the presence or absence of particular system recommendations in the context of particular sets of evidence, and the expected outcome of physician actions on long- and short-term morbidity and mortality given a diagnosis set. Although we acknowledge that many refinements in the utility model are both possible and desirable, our primary purpose here is to demonstrate the fundamental principles of caching cases on the basis of expected-utility considerations.

Generally, the time required to retrieve a case that is stored in the cache is negligible, because it is a simple table-lookup operation; therefore, we assume it to be instantaneous. Let $U_t(d_i)$ represent the utility of informing the physician of the probability of the ith diagnosis t time units after the onset of d_i. Let $U_t(D)$ represent the utility of informing the physician of the probability of all d_i, t time units after the beginning of one or more diagnoses. We calculate $Ut(D)$ by:

$$U_t(D) = 1 - \prod_{d_i \in D} (1 - U_t(d_i)) \quad (4)$$

In this model, which has been called the noisy-OR utility [21], the probability that a patient dies by time t is just the probability that at least one disease will cause him to die by time t. Let $U(D) = U_0(D) - U_t(D)$ represent the net utility of caching a case with the diagnosis set represented by D, and let $P(D \mid E)$ represent the probability of observing the diagnosis set D given the evidence E.

The CaseUtility algorithm randomly samples and caches cases on the basis of their expected net utilities, which are expressed as $U(D)\ P(D \mid E)$. First, each diagnosis is instantiated based on its prior probability, and $U_t(D)$ is calculated using equation (4) for a given t. Next, a random number is sampled from the uniform distribution on [0, 1]; if it is less than $U(D)$, the evidence nodes E are instantiated using the Basic algorithm starting with diagnosis set D; otherwise, another diagnosis set D is generated. This procedure biases the prior probability of the diagnosis set (used in the first step of the Basic algorithm) by the importance of caching cases that are associated with that diagnosis set.

Table 1 shows our subjective estimates of a patient's probability of dying as a direct result of each condition remaining undiagnosed for 0 minutes and for 2 minutes. For example, the probability of a patient dying as a direct result of anaphylaxis (an allergic reaction to a stimulus, in which the patient's blood pressure may drop to a fatally low level) is 0.100 if there is a 2-minute delay in the diagnosis of anaphylaxis (even though the patient may be treated based on his symptoms within this 2-minute period); this risk is reduced to 0.005 if anaphylaxis is diagnosed immediately. In contrast, early diagnosis and intervention do not have much effect on the mortality caused by hypovolemia

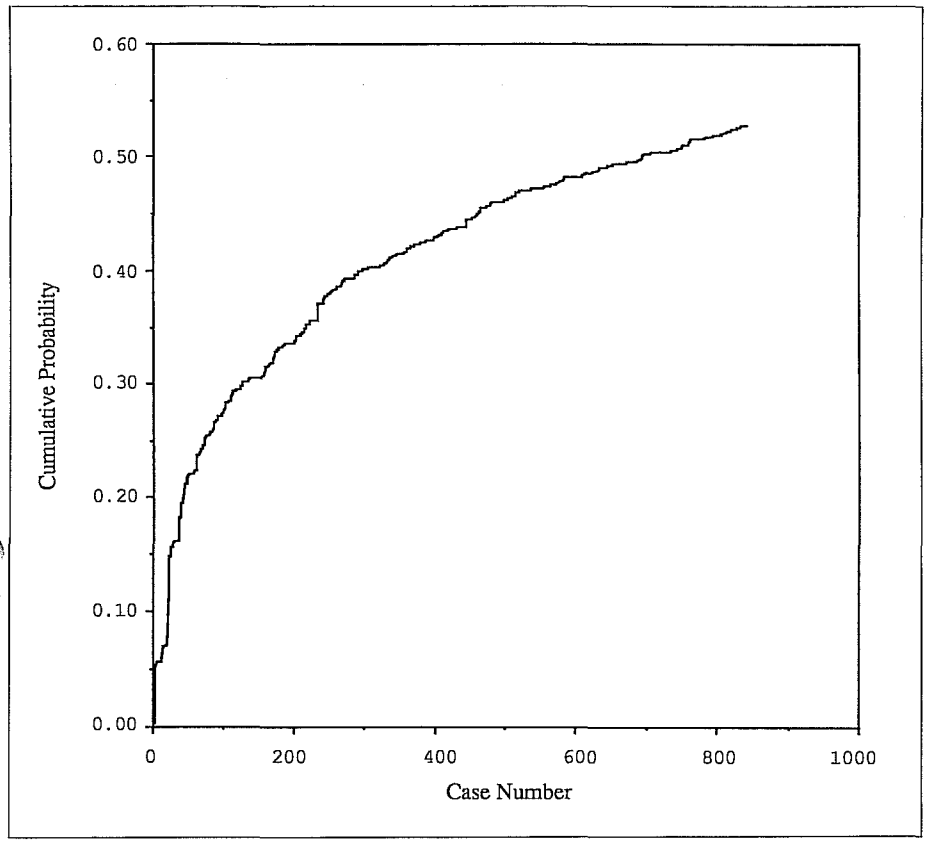

Fig. 6 Cumulative joint probability of the evidence: BASIC algorithm.

(a decrease in the patient's blood volume). We emphasize that the utilities in Table 1 are rough approximations that are intended to demonstrate the concepts underlying utility-based caching, and are not intended to be used in a system that recommends therapy.

3.3 Inference

Cache-based inference employs a daemon that intercepts cases bound for a conventional belief-network evaluation algorithm. The daemon compares the user-supplied case with those in the cache; if a match is found, the marginal posterior probability for each disease is returned immediately. If the case is not found in the cache, the daemon passes the case to the network-evaluation module, and inference proceeds as it would without the cache. The use of a daemon is apparent to the user only because some cases are evaluated much more rapidly than they would have been had the cache not been present.

Table 2 Frequencies and prior probabilities for diagnosis nodes: All algorithms*

Diagnosis	Fraction of Generated Diagnoses			Prior probability
	BASIC	NETUSE	CASEUTILITY	
Hypovolemia	0.125	0.135	0.125	0.10
LV failure	0.076	0.092	0.076	0.05
Anaphylaxis	0.010	0.010	0.019	0.01
A/A insufficient	0.204	0.204	0.201	0.20
Pulmonary embolus	0.015	0.010	0.101	0.01
Intubation				
Normal	0.898	0.903	0.854	0.92
Extubated	0.051	0.041	0.074	0.03
One-sided	0.051	0.055	0.072	0.05
Kinked tube	0.051	0.065	0.068	0.04
Disconnection	0.116	0.107	0.125	0.10

* LV failure: left-ventricular failure; A/A insufficient: insufficient anesthesia or analgesia.

4. Results

Figs. 5 and 6 show the probability mass function (PMF) and cumulative distribution function (CDF), respectively, for the ALARM cache generated by the BASIC algorithm; this cache was generated in approximately 19 hours. Of the 98,304 possible evidence sets in this network, the 841 cases generated (0.85 percent) have a total probability of approximately 0.53. Thus, a user instantiating every evidence node, and no other nodes, will receive results immediately in slightly more than 50 percent of the cases, assuming that the evidence nodes are instantiated in accordance with probabilities in the belief network. The average marginal probabilities for each of the diagnosis nodes are presented in Table 2; these numbers give an estimate of which diagnoses were most common in the generated cases. As expected, BASIC diagnoses were generated roughly in proportion to their prior probabilities.

In contrast to the cache generated by the BASIC algorithm, that resulting from the NETUSE algorithm has many small evidence sets with relatively high probabilities (Fig. 7), reflecting the fact that the user will almost never instantiate values for all the evidence nodes. Note that these evidence sets are unique but not mutually exclusive; for example, both the evidence sets (*CVP* = Low) and (*CVP* = Low, *BP* = High) may be generated as part of this cache. Clearly, the sum of the probabilities of these evidence sets will exceed 1.0 if enough sets are generated. The average marginal probabilities for each of the NETUSE diagnosis nodes (Table 2) reveal approximately the same distribution of diagnoses as in the BASIC cache; only the likelihood of evidence being presented has changed.

Table 2 also shows the average marginal probabilities for each of the diagnosis nodes for the cases in the CASEUTILITY cache. As expected, the diagnoses associated with the greatest changes in utility of using the cache are most likely to be included in cases constituting the cache. In particular, the average marginal probabilities of

Table 3 Explanation of abbreviations used in Figs. 2–4. (From [19], with permission.)

Abbreviation	Number	Meaning
Anest/Analgesia insufficient	20	insufficient anesthesia or analgesia
art. CO2	33	arterial carbon-dioxide content
CO	6	cardiac output
CVP	1	central-venous pressure
Error Cauter	30	error in HR reading due to electrocautery device
Error Low Output	28	error in HR reading due to low cardiac output
exp. CO2	14	carbon-dioxide content of expired gas
FiO2	12	fraction of oxygen in inspired gas
History	3	history of left-ventricular failure
HR BP	7	heart rate obtained from a blood-pressure monitor
HR EKG	8	heart rate obtained from an electrocardiogram
HR SAT	9	heart rate obtained from an oximeter
LVED Volume	25	left-ventricular end-diastolic volume
LV failure	18	left-ventricular failure
Min Vol	15	minute volume, measured
MV	16	minute volume, calculated
PA Sat	32	pulmonary-artery oxygen saturation
PAP	10	pulmonary-artery pressure
PCWP	2	pulmonary-capillary wedge pressure
Pres	13	breathing pressure
Pulm. Embolus	21	pulmonary embolus
SaO2	11	arterial-blood oxygen saturation
Stroke vol.	26	stroke volume
TPR	4	total peripheral resistance
Vent. alv	34	alveolar ventilation
Vent lung	35	pulmonary ventilation
Vent Machine	37	minute ventilation measured at the ventilator
Vent tube	36	ventilation measured at the endotracheal tube

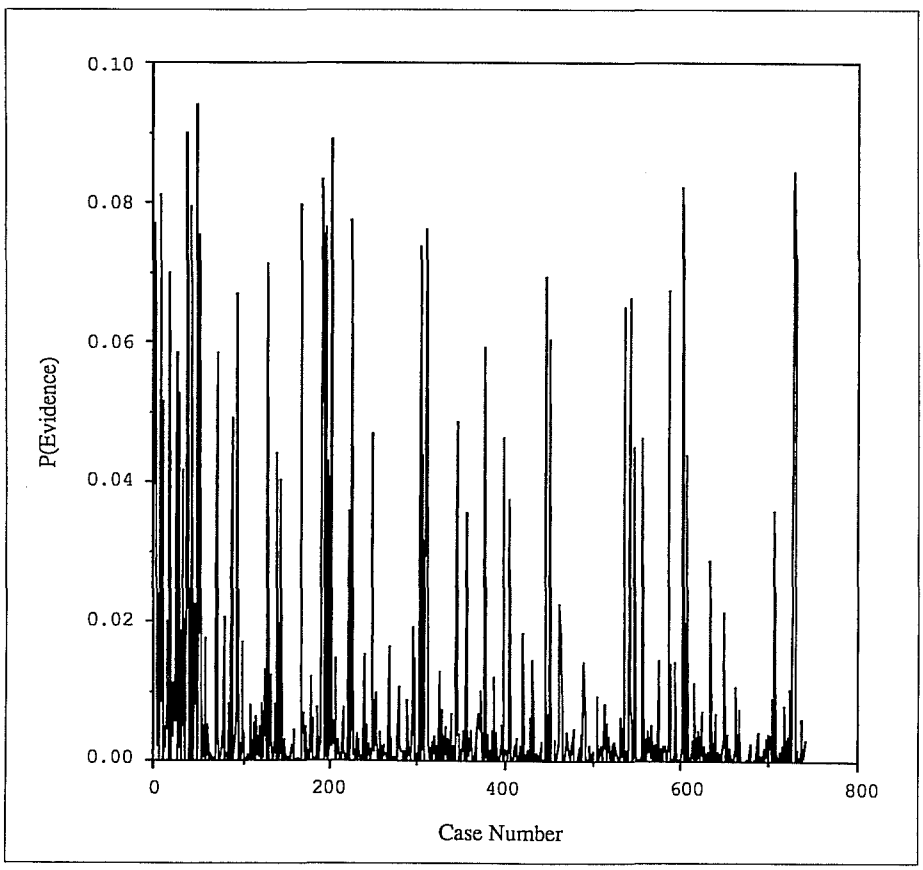

Fig. 7 Joint probability of the evidence: NETUSE algorithm.

anaphylaxis and *pulmonary embolus* are 2 and 10 times higher than their respective prior probabilities.

5. Discussion

The results of the BASIC algorithm are consistent with the hypotheses that a relatively small number of evidence sets in the ALARM belief network span a significant proportion of the total evidence space. In the case of the ALARM belief network, our results indicate that, regardless of the inference algorithm and the hardware platform, a relatively small cache can provide immediate results in more than half of the cases, yielding a 50% speedup in inference. In modeling actual use of the network with the NETUSE algorithm, we find that we can readily relax the strong requirement that the user instantiate all the evidence nodes during a consultation, yet we still generated likely cases based on the properties of the belief network only. The CASEUTILITY algorithm tends to generate diagnosis sets in proportion to the net change in utility of using the cache to diagnose them. The three algorithms have in common the emphasis on improving *average-case* performance; that is, they optimize speed of inference based on expected-probability or expected-value metrics.

This introduction to the potential of the caching paradigm points to several areas for further research. A combination of the CASEUTILITY and NETUSE approaches could be implemented readily, leading to simultaneous modeling of expected use of a belief network (which node values are likely to be instantiated?) and expected need to use the belief network (which cases are important to diagnose early?). Different case generators might be based on daemons that silently sample input as users present cases to a belief network. This approach might be natural for a heavily used belief network, in particular for one instantiated automatically by sensors (for example, a blood-pressure monitor). The subset of nodes cached in the belief network does not need to be

restricted to any of the sets presented here; caching based on subparts of the total belief network (e.g., Markov blankets [22]) might prove to be a powerful technique for separating frequently instantiated subsets of a belief network from the other nodes, thus substantially reducing running time. Belief networks might then be decomposed into modules, each independently cached, leaving only the rarely used subnetworks to be evaluated by the standard algorithms.

Although caching is effective for the ALARM network, its usefulness must be verified for a large variety of other networks. Ultimately, the appropriateness of caching for probabilistic inference must be evaluated empirically; networks that are evaluated rapidly enough using current technology, or those that cannot be evaluated at all using existing algorithms, clearly would not be suitable for caching, since cases could not be generated and evaluated. The networks in which an additional order of magnitude in performance improvement would make the network useful are those that are most likely to benefit from caching the results of probabilistic inference.

We anticipate further work with networks in which exact inference is impractical; for these networks, caching the results of simulation algorithms might yield substantial improvements in expected time for performing inference. Caching algorithms might also be appropriate subjects of metareasoning algorithms that determine the tradeoff between precomputation and inference time [23]. The results presented here lead us to believe that cache-based inference can play an important role in enhancing the performance of real-time probabilistic expert systems in medicine.

Acknowledgements

We thank H. Jacques Suermondt for providing us with his implementation of the Lauritzen-Spiegelhalter algorithm. We gratefully acknowledge the assistance of Ingo Beinlich, who supplied us with the ALARM belief network and provided insightful criticism of earlier versions of this paper. We thank Harold Lehmann for his comments regarding the theoretical basis of our algorithms. Lyn Dupré provided excellent editorial advice. This work is supported by training grant LM-07033 from the National Library of Medicine, by grant IRI-8703710 from the National Science Foundation, and by grant P-25514-EL from the U.S. Army Research office. Computing resources were provided by grant RR-00785 from the Division of Research Resources of the National Institutes of Health.

REFERENCES

1. Pearl J. Evidential reasoning using stochastic simulation of causal models. Artif Intell 1987; 32: 245–57.
2. Good IJ. A causal calculus (I). Brit J Philos of Science 1961; 11: 305–18.
3. Good IJ. A causal calculus (II). Brit J Philos of Science 1961; 12: 43–51.
4. Cooper GF. NESTOR: A Computer-Based Medical Diagnostic Aid that Integrates Causal and Probabilistic Knowledge (Ph. D. dissertation). Department of Medical Information Sciences. Stanford CA: Stanford University, 1984.
5. Shachter RD. Evaluating influence diagrams. Operations Res 1986; 34: 871–82.
6. Howard RA, Matheson JE. Readings on the Principles and Applications of Decision Analysis. Menlo Park CA: Strategic Decisions Group, 1984.
7. Lauritzen SL, Spiegelhalter DJ. Local computations with probabilities on graphical structures and their applications to expert systems. J Royal Statist Soc 1988; 50: 157–224.
8. Cooper GF. Current research directions in the development of expert systems based on belief networks. App Stoch Models and Data Anal 1989; 5: 39–52.
9. Horvitz EJ, Breese JS, Henrion M. Decision theory in expert systems and artificial intelligence. J Approx Reas 1988; 2: 247–302.
10. Chavez RM, Cooper GF. A randomized approximation algorithm for probabilistic inference on Bayesian belief networks. Networks 1990; 20: 661–85.
11. Heckerman DE, Horvitz EJ, Nathwani BN. Update on the Pathfinder project. In: Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care. Washington DC: IEEE Computer Society Press, 1989, 203–7.
12. Lehmann HP. Knowledge acquisition for probabilistic expert systems. In: Proceedings of the Twelfth Symposium on Computer Applications in Medical Care. Washington DC: IEEE Computer Society Press, 1988, 73–7.
13. Pearl J. Fusion, propagation, and structuring in belief networks. Artif Intell 1986; 29: 241–88.
14. Suermondt HJ. Updating probabilities in multiply connected belief networks. In: Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence. Minneapolis MIN: American Association for Artificial Intelligence, 1988, 335–43.
15. Cooper GF. The computational complexity of probabilistic inference using Bayesian belief networks. Artif Intell 1990; 42: 393–405.
16. Henrion M. Propagation of uncertainty by probabilistic logic sampling in Bayes' networks. In: Kanal LN, Lemmer JF (eds). Uncertainty in Artificial Intelligence, 2. Amsterdam: Elsevier Science Publ, 1988: 149–63.
17. Pearl J. Addendum: Evidential reasoning using stochastic simulation of causal models. Artif Intell 1987; 33: 131–2.
18. Schoppers MJ. Universal plans for reactive robots in unpredictable environments. In: Proceedings of the Tenth International Joint Conference on Artificial Intelligence. Milan Italy: International Joint Conferences on Artificial Intelligence, 1987, 1039–46.
19. Shachter RD. Intelligent probabilistic inference. In: Kanal LN, Lemmer JF (eds). Uncertainty in Artificial Intelligence. Amsterdam: Elsevier Science Publ, 1986: 371–82.
20. Beinlich IA, Suermonth HJ, Chavez RM, Cooper GF. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medicine. London: Springer Verlag, 1989; 38: 247–56.
21. Heckerman DE, Horvitz EJ. Personal communication, 1989.
22. Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo CA: Morgan Kaufmann Publ, 1988.
23. Breese JS, Horvitz EJ. Ideal reformulation of belief networks. In: Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence. Cambridge MA: Association for Uncertainty in Artificial Intelligence, 1990, 64–9.

Address of the authors:
E. H. Herskovits and G. F. Cooper,
Section on Medical Informatics,
Stanford University,
MSOB, X-215,
Stanford, CA 94305, USA