# A Case-Based Filter for Diagnostic Belief Networks

*N.B. Peek*
Dept. of Law and Information Technology
University of Leiden
P.O. Box 9521
2300 RA Leiden
The Netherlands

*L.C. van der Gaag*
Dept. of Computer Science
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

**Abstract**
Special-case algorithms for Bayesian belief networks are designed to alleviate the computational burden of problem solving. These algorithms provide a case base for storing solutions for a small number of situations that are likely to be encountered during problem solving. This case base is employed as a *filter* for belief-network inference: for a problem under consideration, the network at hand is consulted only if the case base does not provide a solution for the problem. We present a new algorithm that further extends on the basic idea of special-case algorithms by exploiting knowledge about the way diagnostic problem solving with a belief network is shaped.

## 1 Introduction

From the early days of artificial intelligence research, the computational complexity of methods for automated problem solving is an issue of major concern: the quest for efficient methods is prevalent in most fields of modern AI research. The field that is concerned with automated reasoning in the presence of uncertain and incomplete information is no exception. Halfway through the 1980s, research in this field resulted in the framework of *(Bayesian) belief networks* [1]. This framework provides a powerful and intuitively appealing formalism for representing probabilistic information and in addition provides a set of algorithms for sound probabilistic inference. The belief network framework is suitable for various domains of application and for various types of task. Yet, over the last few years, it is becoming especially popular for building *diagnostic* knowledge-based systems [2, 3, 4]: experience with applying the belief network framework for diagnostic problem solving indicates that the framework is tailored to the task of diagnosis as it provides for reasoning with both causal and associative relationships in a domain.

Unfortunately, the basic algorithms of the belief network framework have a worst-case computational complexity that is exponential in the size of a network. In essence, this problem cannot be remedied as general probabilistic inference with belief networks is $\mathcal{NP}$-hard [5]. One way of enhancing the average-case complexity of problem solving with a belief network is the use of a *special-case algorithm* [6]. Special-case algorithms are based on the idea of shifting part of the burden of runtime computation to a pre-runtime computational phase: for a number of situations that are likely to be encountered during problem solving, solutions are precomputed and stored in a *case base*. At runtime, this case base is employed as a *filter* for belief-network inference: problem solving commences with a search in the case base and only if no precomputed solution is found is the belief network consulted. The use of a special-case algorithm has been reported to yield substantial computational savings for belief networks where a relatively small number of cases covers a large proportion of the likely uses of the network.

In this paper, we present a new special-case algorithm for supporting diagnostic problem solving. Our algorithm differs from other special-case algorithms for belief networks in that it explicitly makes use of knowledge about the way problem solving is shaped. The paper is organised as follows. In Section 2 the belief network framework is briefly reviewed. In Section 3 we describe diagnostic problem solving with a belief network; Section 4 then presents our special-case algorithm for supporting this type of problem solving. Section 5 addresses the precomputation of a case base to be used with our algorithm. The paper is rounded off with some conclusions and directions for further research in Section 6.

## 2 Preliminaries

A *(Bayesian) belief network* is a concise representation of a joint probability distribution on a set of variables. In a belief network, concision of representation is arrived at by explicit separation of information about the independencies holding among the variables in the distribution and the numerical quantities involved. A belief network therefore is composed of two parts: a *qualitative* part modelling independencies and a *quantitative* part specifying probabilities.

The qualitative part of a belief network takes the form of an acyclic directed graph. In this digraph, each vertex represents a variable that can take one of a finite set of values. The set of arcs of the digraph models the independencies among these variables. Informally speaking, we take an arc $V_i \rightarrow V_j$ in the digraph to represent a direct 'influential' or 'causal' relationship between the variables $V_i$ and $V_j$; the direction of the arc designates $V_j$ as the 'effect' or 'consequence' of the cause $V_i$. Absence of an arc between two vertices means that the corresponding variables do not influence each other directly and hence are (conditionally) independent. With each vertex of the digraph is associated a set of (conditional) probabilities describing the influence of the values of the vertex' predecessors on the probabilities of the values of the vertex itself. These probabilities with each other constitute the quantitative part of the belief network and, with the qualitative part, suffice for describing the joint probability distribution [1].

In the sequel, we will use Pr to denote a joint probability distribution under consideration. We will further restrict the discussion to *binary* variables taking one of the truth values *true* and *false*; the generalisation to variables with more than two discrete values, however, is straightforward. For abbreviation, we will use $v_i$ to denote the proposition that the variable $V_i$ takes the value *true*; $V_i = \mathit{false}$ will be denoted as $\neg v_i$. For any set of variables $V$, a conjunction $c_V$ of value assignments to the variables from $V$ is called a *configuration* of $V$. To avoid abundance of braces, we will write $c_{V_i}$ instead of $c_{\{V_i\}}$ for singleton sets $\{V_i\}$.

## 3 Diagnostic Problem Solving With Belief Networks

The objective of *diagnostic problem solving* is to identify a most likely explanation for a problem under consideration — this explanation is called the *diagnosis* of the problem. Establishing a diagnosis generally is supported by gathering information about the manifestations of the problem at hand. This information typically is obtained from applying *tests* to the problem. In most domains, it is not necessary to collect information on all possible manifestations before an accurate diagnosis is reached: information from only a few tests generally suffices. Moreover, it often is not desirable to apply all tests available as testing may be costly or damaging. In diagnostic problem solving therefore, tests should not be applied as a matter of course but selected carefully based on their expected usefulness. In this section, we take a closer look at diagnostic problem solving in the context of Bayesian belief networks.

### 3.1 Evidence Clusters

In most application domains, several tests are available for obtaining information about a problem under consideration. Examples of tests in a medical domain are laboratory tests and operative procedures. The information yielded upon applying a test typically pertains to one or more variables modelled in the belief network; these variables will be called *evidence variables*. With every available test $T_i$ we consider associated a set $\Phi_i$ of evidence variables for which applying the test will yield a value; this set $\Phi_i$ is called the *evidence cluster* of test $T_i$. The number of possible configurations that can be observed for an evidence cluster $\Phi_i$ of size $|\Phi_i| = m$ equals $2^m$; we will enumerate these configurations as $\phi_i^1, \ldots, \phi_i^{2^m}$. In the sequel, we will assume that $\Phi_i \neq \Phi_j$ for all $i \neq j$. We do *not* require, however, that $\Phi_i \cap \Phi_j = \varnothing$, $i \neq j$, that is, we allow for evidence clusters to share some variables. The union of the evidence clusters for all tests discerned constitutes the set of all evidence variables in the belief network and will be denoted as $E$.

For the purpose of selecting tests during problem solving, we distinguish between *surface evidence* and *deep evidence*. Surface evidence is evidence that in general is readily available, that is, without considerable cost; in a medical domain, an example is a patient's medical history. We assume that all variables modelling surface evidence are comprised in a single evidence cluster $\Phi_0$, called the *surface evidence cluster.* The phrase *deep evidence* is used to denote evidence that is hard or costly to obtain; in a medical domain, an example of deep evidence is information from an operative procedure. This type of evidence should only be sought after if strictly necessary for reaching a diagnosis.

## 3.2 Hypotheses

Most application domains involve various hypotheses that need be investigated during problem solving. In a medical domain, these hypotheses typically correspond with disorders. In the sequel, we will assume that each hypothesis is modelled as a separate variable in the belief network. These variables will be called *hypothesis variables*; the set of all hypothesis variables in the network will be denoted as $H$.

Each hypothesis discerned has some prior probability of being the diagnosis for a problem under consideration. The information that becomes available about the manifestations of the problem influences these probabilities: it may increase the probability of some of these hypotheses and decrease the probability of others. Now, if the probability of a hypothesis has increased so as to surpass a pre-defined threshold value, it becomes a likely candidate for the diagnosis; conversely, if the probability of a hypothesis has dropped below a pre-set threshold value, it very likely is not an explanation for the problem under consideration. More formally, we say that a hypothesis variable $H_j$ is a *confirmation candidate* given available evidence $c$ if $\Pr(h_j \mid c) \geq \varepsilon_1$, where $\varepsilon_1$ is a *confirmation threshold* with $0 < \varepsilon_1 \leq 1$; $H_j$ is called a *rejection candidate* given $c$ if $\Pr(h_j \mid c) \leq \varepsilon_2$, where $\varepsilon_2$ is a *rejection threshold* with $0 \leq \varepsilon_2 < \varepsilon_1$. A hypothesis variable is said to be *pending* given $c$ if it is neither a confirmation candidate nor a rejection candidate given $c$.

Now, consider a hypothesis whose probability after processing some evidence $c$ has surpassed the confirmation threshold. Although this hypothesis' variable is a confirmation candidate given $c$, it not necessarily will continue to be so as problem solving progresses: observing further evidence may very well decrease the hypothesis' probability. A hypothesis therefore is considered confirmed only if there is some guarantee that its probability will not decrease considerably in the future. More formally, we say that a confirmation candidate $H_j$ is *established* given evidence $c$ if for each configuration $\phi_i^k$ of every evidence cluster $\Phi_i$ with $\Pr(\phi_i^k \mid c) \geq \delta$, we have that $\Pr(h_j \mid c \wedge \phi_i^k) \geq \varepsilon_1$, where $\delta > 0$ is a pre-defined *guarantee threshold.* A similar observation holds for a hypothesis whose probability has dropped below the rejection threshold.

We would like to emphasize that the threshold values $\varepsilon_1, \varepsilon_2$, and $\delta$ are highly domain-dependent and should be chosen with care.

## 3.3 Test Planning

Diagnostic problem solving involves careful *planning* of tests to apply to a problem to minimise the *cost* of establishing a diagnosis. Test planning amounts to *selecting* the best tests to apply to the problem, and *evaluating* whether enough information has been obtained to establish a diagnosis with sufficient accuracy.

For selecting the best tests to apply to a problem, we make use of concepts from Bayesian decision theory [7]. The basic idea is to measure for each available test $T_i$ the expected usefulness of the information yielded upon application of the test. To this end, we assess for each possible configuration $\phi_i^k$ of its associated evidence cluster $\Phi_i$, the desirability of obtaining this information in the context of formerly obtained evidence $c$. This desirability is expressed as a numerical value $u(\phi_i^k, c)$, called the *utility* of $\phi_i^k$ given $c$. A utility may be based on probabilistic information only and not contain any other information about the domain at hand; a utility, however, may also involve non-probabilistic aspects from the domain such as the cost of obtaining the information [8]. In this paper, we use a very simple type of utility that is based on the changes in the probabilities of the various yet unestablished hypotheses incurred by the configuration at hand, following [9] in essence: we take the utility of a configuration $\phi_i^k$ given $c$ to be

$$u(\phi_i^k, c) \;=\; \textstyle\sum_{H_j \in H(c)} \Pr(h_j \mid c) \cdot |\Pr(h_j \mid c) - \Pr(h_j \mid c \wedge \phi_i^k)|$$

where $H(c)$ is the set of hypothesis variables that are not yet established given $c$. Now observe that before applying test $T_i$, it is not known *which* configuration of its associated evidence cluster will be found for the problem at hand: each configuration $\phi_i^k$ is observed with some probability $\Pr(\phi_i^k \mid c)$. The *expected utility* $\hat{u}(T_i, c)$ of applying test $T_i$ in the context of evidence $c$ therefore equals

$$\hat{u}(T_i, c) \;=\; \textstyle\sum_k \Pr(\phi_i^k \mid c) \cdot u(\phi_i^k, c)$$

The best test to apply now is the test with highest expected utility given $c$; in the sequel, we will use $\hat{T}(c)$ to denote this test. Note that all probabilities required for calculating expected utilities of tests can be computed from the belief network. Also note that evidence variables to acquire information on are selected *groupwise* by employing knowledge about the tests available in the domain; variables therefore are not selected one by one as in a fully myopic approach to evidence gathering [10].

After a test has been applied to a problem under consideration and the evidence yielded has been processed, it is evaluated whether a sufficiently accurate diagnosis can be reached on the basis of the available information. To this end, several different criteria can be used, depending on the domain at hand. In this paper, we assume that gathering evidence is pursued until every hypothesis variable modelled in the belief network is established; each confirmed hypothesis then is a likely explanation for the problem. Note that the threshold values $\varepsilon_1, \varepsilon_2$, and $\delta$ introduced in Section 3.2 determine the persistence with which evidence gathering is pursued.

Figure 1 summarises the structure of diagnostic problem solving. Initially, all surface evidence is acquired for a problem and processed. Based on the available evidence, the best test to apply next is selected. The user is asked to apply this test to the problem and enter the information yielded. This process is repeated until a sufficiently accurate diagnosis has been reached. We would like to note that diagnostic problem solving as outlined above may be computationally expensive: it is exponential in the number of variables comprised in the separate evidence clusters. We expect, however, that this number will be bounded by a (small) constant for most applications.

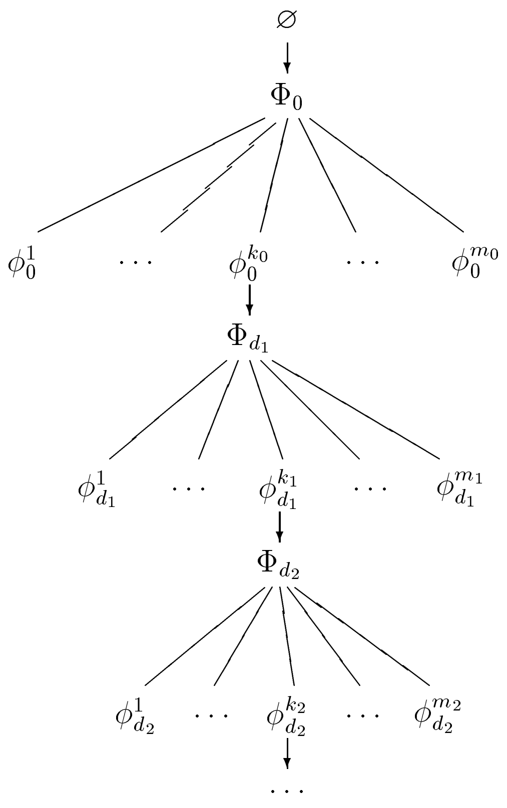

Figure 1: The Structure of Problem Solving.

## 4 A Case-Based Filter

Special-case algorithms for belief networks are designed to alleviate the computational burden of problem solving by exploiting knowledge about the likely uses of a network. These algorithms provide a *case base* for storing solutions for situations that are likely to be encountered during problem solving and a *retrieval algorithm* for searching this case base. In this section, we present a new special-case algorithm to support diagnostic problem solving as outlined in the foregoing. In our algorithm, the case base is organised as a tree closely resembling the structure of problem solving depicted in Figure 1; the associated retrieval algorithm basically is a tree traversal algorithm.

## 4.1 The Case Base

To allow for exploiting knowledge about the likely uses of a belief network, a case base for the network stores situations that can be encountered during problem solving and their associated solutions. These situations typically are descriptions of problem manifestations and will be referred to as *cases*.

In diagnostic problem solving as outlined before, the situations that can be encountered are fixed. Consider once more the problem-solving structure depicted in Figure 1. Initially, there is no information available about a problem under consideration. The case representing this situation is the configuration of the empty set, that is, the case *true*; in the sequel, we will refer to this case as the *null case*. Diagnostic problem solving commences with acquiring all surface evidence for the problem at hand. Every possible configuration of the surface evidence cluster therefore is a case representing a situation that can initially be encountered. As problem solving progresses, tests are selected and applied, yielding further evidence concerning the manifestations of the problem. The situations that can then be encountered reflect the selected tests and the cases representing these situations are built from configurations of successive evidence clusters. More formally, we have that a case $c$ is a configuration $c \equiv \phi_{i_1}^{k_1} \wedge \cdots \wedge \phi_{i_m}^{k_m}$, $m \geq 0$, such that either $c = true$, or $\phi_{i_1}^{k_1}$ is a configuration of the surface evidence cluster $\Phi_0$ and, for all $j = 1, \ldots, m-1$, we have that $T_{i_{j+1}} = \hat{T}(\phi_{i_1}^{k_1} \wedge \cdots \wedge \phi_{i_j}^{k_j})$; $m$ is called the *order* of case $c$, denoted $ord(c) = m$. Note that cases of equal order may differ in length as different evidence clusters may have different sizes. In the sequel, we will use $ev(c) = \Phi_{i_1} \cup \cdots \cup \Phi_{i_m}$ to denote the set of evidence variables that are assigned a value in case $c$.

Since cases are built from configurations of successive evidence clusters, we have that each case is subsumed by one or more cases of lower order. We say that a case $c'$ is a *subcase* of a case $c$ if $ev(c) \subseteq ev(c')$ and $c' \wedge c \equiv c$, that is, $c$ and $c'$ match with respect to the configuration of $ev(c)$; $c$ then is called a *supercase* of $c'$. The case $c'$ is called a *primary subcase* of $c$ if it is a subcase of $c$ and $ord(c') = ord(c) + 1$; $c$ then is called the *primary supercase* of $c'$. Note that each case has at most one primary supercase and may have various primary subcases. We say that a set of cases $S$ is *sound* if for every case $c \in S$, $c \neq true$, we have that $c' \in S$ where $c'$ is the primary supercase of $c$; $S$ is called *complete* if for every case $c \in S$ we have that $c' \in S$ for all subcases $c'$ of $c$.

We now organise a sound set of cases in a directed tree in which the nodes represent cases and the arcs represent the primary subcase-relationship among the cases; this directed tree is called a *case tree* and constitutes the case base for our special-case algorithm. More formally, the case tree for a sound set of cases $S$ is a directed tree

$T_S = (V(T_S), A(T_S))$ such that $V(T_S) = S$ and $(c, c') \in A(T_S)$ if and only if $c$ is the primary supercase of $c'$. If $S$ is both sound and complete, the case tree $T_S$ is called a *complete* case tree; otherwise, $T_S$ is called a *partial* case tree. Note that a complete case tree represents all possible situations that might be encountered during problem solving were evidence gathering pursued until all tests had been applied.

## 4.2 The Retrieval Algorithm

For searching a case base during problem solving with a belief network, a special-case algorithm provides a *retrieval algorithm.* The retrieval algorithm provided by our special-case algorithm basically is a *tree traversal* algorithm for traversing a partial case tree $T_S = (V(T_S), A(T_S))$. For a problem under consideration, the retrieval algorithm starts at the root of the case tree as it corresponds with the null case representing the initial situation where no information is available as yet. Now, suppose that a vertex $c \in V(T_S)$ is visited. Recall that this vertex models a situation that is encountered during problem solving. In this situation, a test is selected and applied to the problem at hand yielding a configuration of its associated evidence cluster. The newly arisen situation is represented by a primary subcase $c'$ of $c$. If this subcase $c'$ is comprised in the set $S$, then $(c, c') \in A(T_S)$ and the retrieval algorithm traverses this arc; otherwise, the traversal halts.

To fully exploit the presence of the case base, the retrieval algorithm should be *self-supporting* and not rely on belief-network inference. To circumvent the need for consulting the network during retrieval, additional information is stored with each case $c$ in the case tree:

- the *test* $\hat{T}(c)$ to be applied to the problem under consideration when the situation described by the case is encountered;
- for each hypothesis variable $H_j$, the *probability* $\Pr(h_j \mid c)$;
- for each hypothesis variable $H_j$, its *status* given $c$, that is, whether or not it is an established candidate given $c$.

The test $\hat{T}(c)$ stored with case $c$ in the case tree allows for the tree traversal to be self-supporting. When the traversal has halted, the information concerning the statuses of the hypothesis variables stored with the case provides for deciding on the necessity of further gathering of evidence. If every hypothesis variable is an established candidate given $c$, then a sufficiently accurate diagnosis has been reached and no further information is required; the diagnosis then is described by the probabilities for the various hypothesis variables stored with the case. If, on the other hand, there is at least one hypothesis variable that is not yet established, the case corresponds with a situation that needs further refining: all evidence observed so far is entered into the belief network and problem solving proceeds with belief-network inference.

## 5 The Precomputation Algorithm

In general, a case base to be used with a special-case algorithm is generated from a belief network by means of a *precomputation algorithm.* For our special-case algorithm, precomputation amounts to computing cases and building a partial case tree of suitable size. Before detailing our precomputation algorithm, we observe that for precomputing a case base several probabilities are required. Our precomputation algorithm makes use of a *simulation algorithm* for estimating these probabilities from the network at hand [11].

The basic idea of a simulation algorithm for a belief network is to generate a finite multiset of configurations of all variables involved, reflecting the joint probability distribution defined by the network; the separate elements from this multiset are called *samples.* From the generated multiset of samples, (conditional) probabilities of interest are estimated based on occurrence frequencies. Accuracy of these estimates can be obtained by generating a sufficiently large number of samples. More formally, if for the number of samples $N$ in a properly generated multiset $\Omega$ we have that

$$N > \frac{2}{\varepsilon_E^2} \ln \frac{1}{\delta_E}$$

then a probability estimate from $\Omega$ is within (absolute) error bound $\varepsilon_E$ with probability $1 - \delta_E$ [12]; conversely, a probability estimate from $\Omega$ is within the error bound

$$\varepsilon_E < \sqrt{\frac{2}{N} \ln \frac{1}{\delta_E}}$$

with probability $1 - \delta_E$. In the sequel, we assume that for the purpose of precomputing a case base a multiset $\Omega$ of $N$ samples has been generated.

### 5.1 Precomputing Cases

Precomputing a case for inclusion in the case base essentially boils down to computing the information to be stored with the case. We recall from Section 4.3 that with each case we store a test to be applied, the conditional probabilities of the various hypotheses discerned, and the statuses of all hypothesis variables. For a case $c$, this information is computed from the multiset $\Omega(c) = \{\omega \mid \omega \wedge c \equiv c, \omega \in \Omega\}$ of samples that match the case $c$; in the sequel, we will use $N(c)$ to denote the number of samples in $\Omega(c)$.

For each hypothesis variable $H_j$ in the network, the probability $\Pr(h_j \mid c)$ to be stored with case $c$ is estimated directly from the multiset of samples $\Omega(c)$: we take

$$\Pr(h_j \mid c) = \frac{n}{N(c)}$$

where $n$ is the number of samples $\omega \in \Omega(c)$ with $\omega \wedge h_j \equiv h_j$. The test $\hat{T}(c)$ to be applied to a problem under consideration in the context of available evidence $c$ is determined as outlined in Section 3.3. The probabilities required for computing the expected utilities of the various tests in the domain are estimated directly from appropriate multisets of samples from $\Omega(c)$, as before.

Now recall that we have defined a hypothesis variable $H_j$ to be a *confirmation candidate* given $c$ if the probability $\Pr(h_j \mid c)$ surpasses the pre-defined confirmation threshold $\varepsilon_1$. Since for the purpose of precomputation of cases we make use of estimates for probabilities of interest, confirmation candidacy cannot be determined with absolute certainty for a variable. We therefore allow for a small error to be made in assigning the status of confirmation candidate. Suppose that the probability $\Pr(h_j \mid c)$ for a hypothesis variable $H_j$ has been estimated to be $\frac{n}{N(c)}$. For this estimate, we have that

$$\frac{n}{N(c)} - \sqrt{\frac{2}{N(c)} \ln \frac{1}{\delta_E}} < \Pr(h_j \mid c) < \frac{n}{N(c)} + \sqrt{\frac{2}{N(c)} \ln \frac{1}{\delta_E}}$$

with probability $1 - \delta_E$. We now take $H_j$ to be a confirmation candidate given $c$ if

$$\frac{n}{N(c)} - \sqrt{\frac{2}{N(c)} \ln \frac{1}{\delta_E}} \geq \varepsilon_1$$

Note that by this inequality the hypothesis variable $H_j$ is assigned the status of confirmation candidate erroneously with a probability *smaller* than $\delta_E$. Also note that the inequality provides a conservative lower bound for confirmation candidacy. Similar observations apply for determining rejection candidacy for a variable and for determining whether or not a candidate is established; for further details, we refer to [13].

## 5.2 Building the Case Tree

Our precomputation algorithm builds a *partial* case tree for a belief network: the basic idea is that only cases with a high probability of being encountered during problem solving are included. Building the case tree therefore amounts to deciding for each case whether inclusion is justified by the expected frequency of its use.

A partial case tree for a belief network is generated in a breadth-first fashion, that is, a case of order $k+1$ is considered for inclusion in the case tree only after all cases of order $k$ have been considered. Now suppose that inclusion of subcases of a case $c$ that is already present in the case tree in the making, is considered. We recall that with case $c$ information is stored about the statuses of the various hypothesis variables. If this information reveals that every hypothesis variable is an established candidate, then case $c$ describes a situation for which a sufficiently accurate diagnosis can be established without further evidence gathering. During problem solving, therefore, never a situation corresponding with a subcase of $c$ is encountered. In the case tree, no subcases of case $c$ are included, rendering case $c$ a leaf of the tree. If, on the other hand, the statuses of the hypothesis variables stored with case $c$ reveal that the case needs further refining, then each of the primary subcases of $c$ is investigated separately. A primary subcase is included in the case tree only if the probability of encountering the situation represented by this subcase surpasses a pre-defined probability threshold.

To conclude, we would like to note that precomputation is computationally expensive. Our precomputation algorithm, however, has been designed to have an *anytime* property: precomputation will always yield a case tree, but the more time is given to the precomputation, the better its quality. We would further like to stress that the precomputation needs to be performed only once, prior to any problem solving; it therefore is best done at an off-peak moment, when considerable computing resources are available.

## 6 Conclusions

Belief networks by now have established their position of valuable representations of domain knowledge, as the increasing number of applications of the belief network framework demonstrate. Yet, as applications are growing more and more complex it becomes evident that the computational complexity of the basic algorithms associated with the framework resists efficient problem solving. The quest for efficient problem-solving methods for use with a belief network therefore is prevalent in modern belief-network research. The special-case algorithms that have emerged from this research enhance problem solving by exploiting knowledge about the likely uses of a network.

We have presented a new special-case algorithm for belief networks that exploits not only knowledge about the likely uses of a network but also knowledge about how problem solving with the network is shaped. In particular, our algorithm builds on the observation that the situations encountered during problem solving are fixed as a result of the evidence gathering strategy employed. We would like to note that although our algorithm has been designed to support diagnostic problem solving, it can be customised to other problem-solving tasks as well.

Our special-case algorithm has not been tested as yet on real-life belief-network applications. We feel, however, that our algorithm has potential for substantial speedup of problem solving as it further extends on the basic idea of special-case algorithms which have been reported to yield impressive results [6]. As for special-case algorithms in general we expect the best results for belief networks where a relatively small number of cases covers a large proportion of the likely uses of the network. Besides insight in the potential of our algorithm, other interesting issues such as the optimal size of the case base to be used and the effects of the various threshold values employed remain to be addressed. Another challenging issue is the use of learning methods for automated construction or adaptation of the case base. We plan to test our algorithm on real-life belief networks and hope to report on the results obtained in the future.

## References

[1] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers.

[2] S. Andreassen, M. Woldbye, B. Falck, and S.K. Andersen (1987). MUNIN - A causal probabilistic network for interpretation of electromyographic findings, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 366 – 372.

[3] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper (1991). Probabilistic diagnosis using a reformulation of the Internist-1/QMR knowledge base, *Methods of Information in Medicine*, vol. 30, pp. 241-255.

[4] D.E. Heckerman, E.J. Horvitz, and B. Nathwani (1992). Towards normative expert systems: part I, the Pathfinder project, *Methods of Information in Medicine*, vol. 31, pp. 90-105.

[5] G.F. Cooper (1990). The computational complexity of probabilistic inference using Bayesian belief networks, *Artificial Intelligence*, vol. 42, pp. 393-405.

[6] E.H. Herskovitz and G.F. Cooper (1991). Algorithms for Bayesian belief-network precomputation, *Methods of Information in Medicine*, vol. 30, pp. 81-89.

[7] J.Q. Smith (1989). *Decision Analysis: a Bayesian Approach*, Chapman and Hall.

[8] P. Glasziou and J. Hilden (1989). Test selection measures, *Medical Decision Making*, vol. 9, pp. 133 – 141.

[9] L.C. van der Gaag and M.L. Wessels (1994). Selective evidence gathering for diagnostic belief networks, *AISB Quarterly*, No. 86, pp. 23-34.

[10] G.A. Gorry and G.O. Barnett (1968). Experience with a model of sequential diagnosis, *Computers and Biomedical Research*, vol. 1, pp. 490 – 507.

[11] S.B. Cousins, W. Chen, and M.E. Frisse (1993). A tutorial to stochastic simulation algorithms for belief networks, *Artificial Intelligence in Medicine*, vol. 5, pp. 315-340.

[12] R. Karp, M. Luby, and N. Madras (1989). Monte-Carlo approximation algorithms for enumeration problems, *Journal of Algorithms*, vol. 10, pp. 429-448.

[13] N.B. Peek (1994). *A Filter for Belief Networks*, M.Sc. thesis INF/SCR-94-20, Utrecht University.