The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks

Ingo A. Beinlich, M.D., H. J. Suermondt, R. Martin Chavez, Gregory F. Cooper, M.D., Ph.D.

Section on Medical Informatics, Stanford University School of Medicine, Stanford, California, USA

Abstract ALARM (A Logical Alarm Reduction Mechanism) is a diagnostic application used to explore probabilistic reasoning techniques in belief networks. ALARM implements an alarm message system for patient monitoring; it calculates probabilities for a differential diagnosis based on available evidence. The medical knowledge is encoded in a graphical structure connecting 8 diagnoses, 16 findings and 13 intermediate variables. Two algorithms were applied to this belief network: (1) a message-passing algorithm by Pearl for probability updating in multiply connected networks using the method of conditioning; and (2) the Lauritzen-Spiegelhalter algorithm for local probability computations on graphical structures. The characteristics of both algorithms are analyzed and their specific applications and time complexities are shown.

Introduction

The goal of the ALARM monitoring system is to provide specific text messages advising the user of possible problems. This is a diagnostic task, and we have chosen to represent the relevant knowledge in the language of a belief network *(Fig.1)*. This graphical representation [Pearl 86b] facilitates the integration of qualitative and quantitative knowledge, the assessment of multiple faults, as required by our domain, and nonmonotonic and bidirectional reasoning.

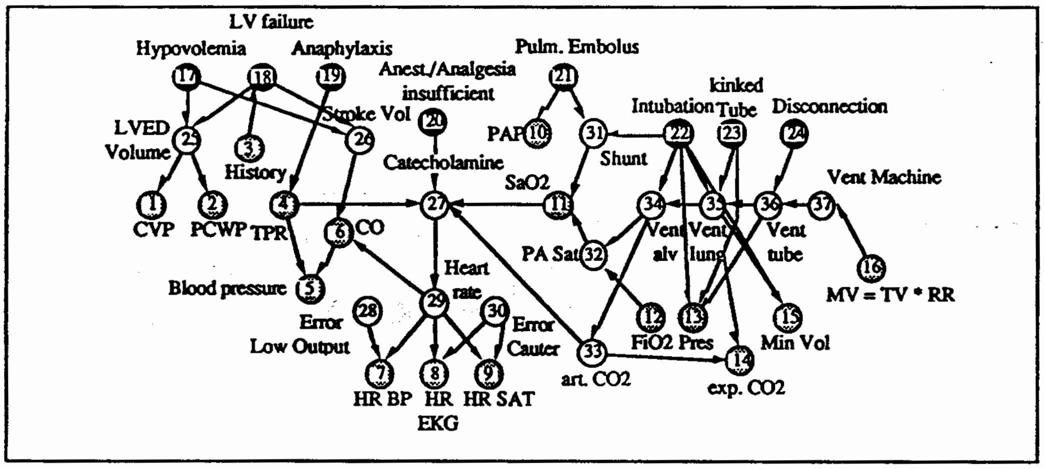


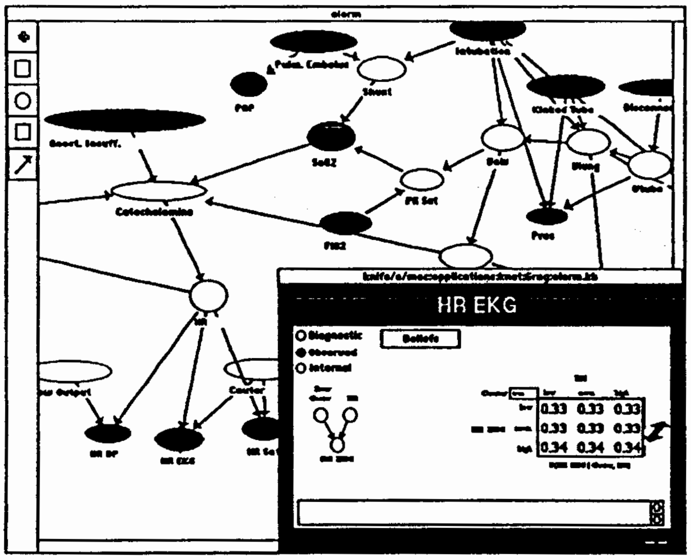
Fig. 1 The ALARM network representing causal relationships is shown with diagnostic (●), intermediate (○) and measurement (◉) nodes. CO: cardiac output, CVP: central venous pressure, LVED volume: left ventricular end-diastolic volume, LV failure: left ventricular failure, MV: minute ventilation, PA Sat: pulmonary artery oxygen saturation, PAP: pulmonary artery pressure, PCWP: pulmonary capillary wedge pressure, Pres: breathing pressure, RR: respiratory rate, TPR: total peripheral resistance, TV: tidal volume

A belief network is a directed, acyclic graph in which nodes represent domain variables and arcs show important dependencies among those variables. Probabilities are attached to nodes and to arcs or local groups of arcs. In particular, for each node N_i without direct predecessors, there is a prior probability function $p(N_i)$; for a node N_j with one or more predecessors P_k there is a conditional probability function $p(N_j \mid P_k)$. These probability functions capture for example our assessment of the chance of having hypovolemia in a patient or that bloodpressure is dependent on cardiac output (CO) and total peripheral resistance (TPR).

Knowledge Acquisition

Three types of variables are represented in ALARM. *Diagnoses* and other qualitative information are at the top level of the network. These variables have no predecessors and they are assumed to be mutually independent a priori. All nodes are associated with a set of mutually exclusive and exhaustive values representing the presence or absence or the severity of a particular disease. *Measurements* represent any available quantitative information. All continuous variables are represented catagorically with sets of discrete intervals dividing the value range. Depending on the necessary level of detail, three to five categories are used per node. *Intermediate variables* are inferred entities that cannot be measured directly.

The dependencies in the network were assessed from diagnosis to evidence. There are three reasons for this approach: First, our domain involves some well-known causal interactions. Second, most textbooks of medicine use this form of description. Third, the representation is less complicated in the causal direction, as more independence assumptions can be made [Shachter 87]. The resulting structure was later edited using KNET, an object-oriented environment developed in our laboratory for drawing and manipulating decision networks *(Fig.2)*.



	[illegible]	[illegible]	[illegible]
[illegible]	0.33	0.33	0.33
[illegible]	0.33	0.33	0.33
[illegible]	0.34	0.34	0.34

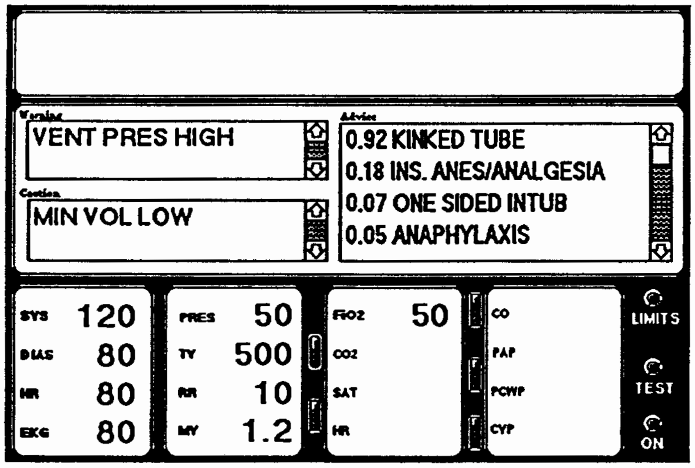
Fig. 2 Editing the belief network in KNET: The object oriented-graphics editor adds a user interface for knowledge acquisition and consultation. The structure of the network is defined first with a set of drawing tools. Probabilities are then entered through an editor specialized for this application (inset).

KNET differs from other tools for expert-system construction in that it combines a direct-manipulation visual interface with the probabilistic inference schemes described in this paper. The KNET architecture defines a complete separation between the design of a user interface on the one hand, and the representation and management of expert opinion on the other [Chavez 88].

The probabilities in a belief network can represent objective as well as subjective knowledge. ALARM contains statistical data on prior probabilities, logical conditional probabilities computed from equations relating variables, and a number of subjective assessments. It is necessary to obtain conditional probabilities for the states of a node, given all different states of the parent nodes. An example of this shown in the inset of the *figure 2:* here the heart rate displayed by an EKG-monitor is conditioned on the heart rate of the patient and whether or not an electrocauter is in use. If the electrocauter is in use, the EKG monitor is equally likely to show a low, normal, or high reading independent of the patient's real heart rate (p = 0.33 for all states); the corresponding matrix of conditional probabilities is displayed on the ALARM probability editor *(Fig. 2)*. If the electrocauter is not in use, then the EKG reading is very likely to correspond to the patient's heartrate (p = 0.98 for identical categories, p = 0.01 otherwise); this probability matrix would be accessed by pointing to one of the arrows on the editor display using a mouse.

A sample consultation

ALARM is a data-driven system. Simulating an anesthesia monitor, ALARM accepts a set of physiologic measurements. An example would be as follows: blood pressure 120/80 mmHg, heart rate 80/min, inspired oxygen concentration 50%, tidal volume 500 ml, respiratory rate 10/min, breathing pressure 50 mbar, and measured minute ventilation 1.2 l/min. These measurements are categorized into 'low', 'normal', 'high', etc. and text messages are generated when measurements are outside of their normal range. These messages will then appear in the *Warning* and *Caution* fields of the monitor depending on their importance *(Fig. 3)*. In the given example, the high breathing pressure of 50 mbar imposes a direct danger to the patient and a warning is issued. The low minute ventilation is less immediate and is displayed as a caution only.



Warning: VENT PRES HIGH

Caution: MIN VOL LOW

Advice:
0.92 KINKED TUBE
0.18 INS. ANES/ANALGESIA
0.07 ONE SIDED INTUB
0.05 ANAPHYLAXIS

SYS 120, DIAS 80, HR 80, EKG 80, PRES 50, TV 500, RR 10, MV 1.2, FIO2 50

Fig. 3

ALARM simulates an anesthesia monitor. It takes patient measurements, displays warning and caution messages, and lists a differential diagnosis.

The categorized information is then passed to the measurement nodes and the network is updated using either the Pearl or Lauritzen–Spiegelhalter algorithm. The posterior probabilities on all diagnostic nodes are then presented as a differential diagnosis on the monitor.

Pearl's Algorithm

Pearl's algorithm implements a local message-passing system for probability updates in a belief network [Pearl 86b]. Each node receives messages on prior probabilities from its parents and on likelihoods from its children. These messages are combined to form a belief in the proposition represented by a node. This belief is then sent to neighboring nodes until an update of the complete network is obtained.

Given a singly connected network with only one path between any two nodes, this algorithm has a running time proportional to the size of the network for a single measurement. In the worst case, messages have to travel from the node containing the measurement to all other nodes in the network; thus, the running time is proportional to the length of all possible paths in the network.

Fortunately, message passing stops under a number of *blocking conditions (Fig. 4): (a)* an observed node does not send information from its children to its parents or from its parents to its children – once, for example, the left ventricular end-diastolic volume (LVED volume) is known, information on left–ventricular failure (LV failure) will not influence the beliefs in central venous pressure (CVP) or pulmonary capillary wedge pressure (PCWP) and vice versa; *(b)* an observed node does not send information from its child to any other children – a measurement of CVP will not influence the belief in PCWP once the LVED volume is known; and *(c)* a node that has not been observed and that does not have any descendents that have been observed does not send information from one parent to any other parents - this states for example the fact that left–ventricular failure and hypovolemia are independent of each other if neither the LVED volume nor CVP or PCWP are known. Thus, depending on the location of the measurement nodes, messages and updates will be necessary for only parts of the network.

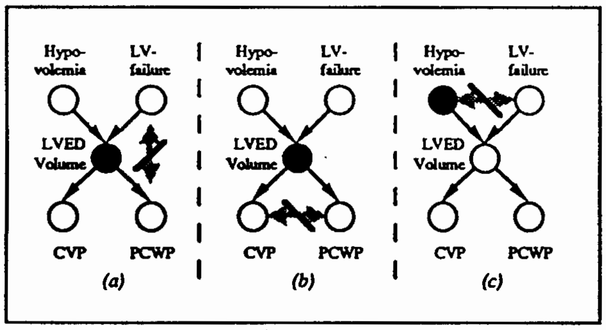


Fig. 4

Blocking conditions: (a) *an observed node does not send information from its children to its parents or vice versa,* and (b) *does not send information from its child to any other children;* (c) *a node that has not been observed and that does not have any observed descendents does not send information from one parent to any other parents*

In the ALARM network, the observation of a 'central venous pressure' (node 1 in *Fig.1*) will affect only nodes up to five links away, if no other information is available. Message propagation will stop at the node for 'blood pressure' if there is no blood pressure measurement available (blocking condition c). On the other hand, information on 'heart rate' (node 8) will change beliefs in almost the entire network *(Fig. 5)*. Information will traverse all of the network for only those combinations of evidence that override the blocking conditions.

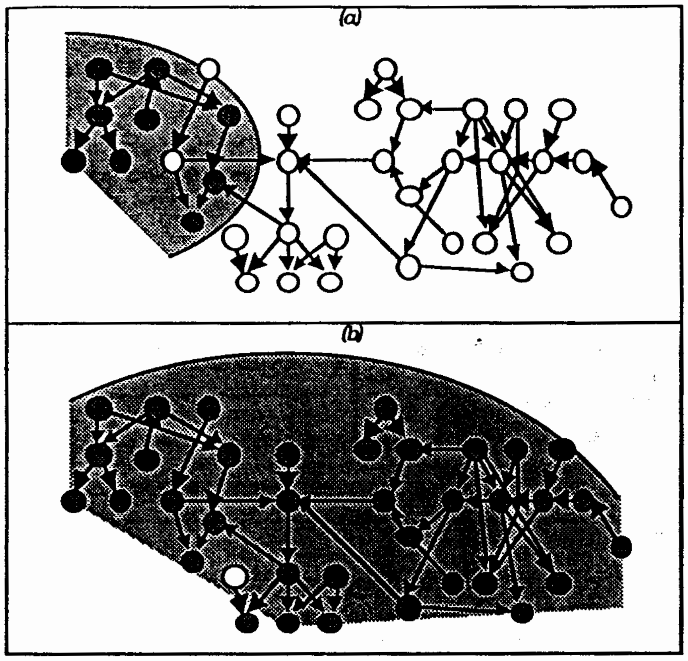


Fig. 5 Propagation of single pieces of evidence with Pearl's algorithm.

(a) information on 'central venous pressure' updates only a part of the network.

(b) On the other hand, information on 'heart rate' propagates through almost the entire network

Pearl's original algorithm works for singly-connected networks only, because multiple paths between nodes, also called loops, can cause messages to cycle indefinitely in the network. Pearl suggest three solutions to this problem: (1) clustering, (2) conditioning, and (3) stochastic simulation [Pearl 86a; 87]. Clustering is the formation of a supernode from nodes that form a loop. As all possible combinations of propositions from the individual nodes must be represented in the supernode, only very small local loops can be handled by this method. Stochastic simulation yields a heuristic approximation of the probability distributions by sampling random cases. The method of conditioning (explained below), or reasoning by assumptions, provides an alternative yielding exact results; it works moderately quickly with a small number of loops.

Conditioning breaks the message-passing cycle by assuming one node in a loop as being observed. When properly chosen, this will prevent message cycling by the blocking conditions described previously. In a network with multiple loops, each loop must be interrupted in such a fashion. The result is a set of nodes, the loop-cutset, that renders a network singly connected when assuming the cutset-nodes are observed.

A multiply connected network must now be updated for all possible instantiations of the cutset-nodes. The algorithm then combines the resulting probabilities, weighting each by the probability of the particular instantiation that was used to generate it. The number of instantiations is equal to the number of all possible combinations of cutset-node propositions, and thus is exponential in the size of the loop-cutset and its propositions.

One of us has recently developed a heuristic technique for finding a cutset for any arbitrary network [Suermondt 88]. Although finding the minimal cutset is NPhard, this algorithm will find a small cutset that is often, but not always, minimal. For the ALARM network, which contains a number of loops, a cutset of five nodes was found *(Fig. 6)*. This set of nodes has 144 possible instantiations, each corresponding to a complete update of the network.

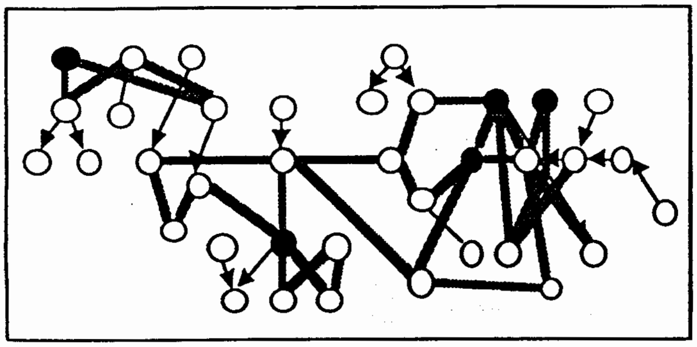


Fig. 6

Loops (▬) and loop-cutset nodes (●) in the ALARM network

In the ALARM system, a collection of data usually becomes available at the same time. That is, the network must be updated simultaneously with information on the blood pressure, the pulse rate, the oxygen saturation, and so on of a patient. In the absence of a special scheduling mechanism for messages, Pearl's algorithm can handle this set of measurements only by sequential updating. First, the information for the patient's blood pressure is incorporated; then the information on the pulse rate is added, and so forth. A complete update must be performed separately for each piece of information .

The time complexity of this algorithm is therefore proportional to the product of the size of the network, the number of cutset instantiations and the size of the measurement set. In our current implementation and computer architecture (Macintosh II) this complexity corresponds to an average time of 8 minutes to update all 8 diagnostic nodes for each set of measurements.

The Lauritzen-Spiegelhalter Algorithm

The Lauritzen-Spiegelhalter algorithm rearranges a network into a tree by forming clusters of nodes [Lauritzen-Spiegelhalter 88]. First, arcs are added between each pair of parents of each node in the graph. The authors of the algorithm call this 'marrying parents' and the result is a 'moralized' graph. Moralizing makes the next step, the triangulation of the graph, more efficient, as this procedure already triangulates parts of the graph. A triangulated graph contains no cycles of length 4 or more without a short-cut [Gavril 72; Lauritzen 84]. Next, the network is converted to an undirected graph, and more edges are added to completely triangulate the graph using the maximum cardinality search algorithm by Tarjan [Tarjan 84]. Darroch has shown that a triangulated graph is decomposable into clusters of nodes, called cliques, for which a joint probability and marginal probabilities are relatively easy to compute [Darroch 80, Spiegelhalter 87]. A clique is defined as a set of nodes such that each node in the set has an arc to all other nodes in the set *(Fig. 7)*.

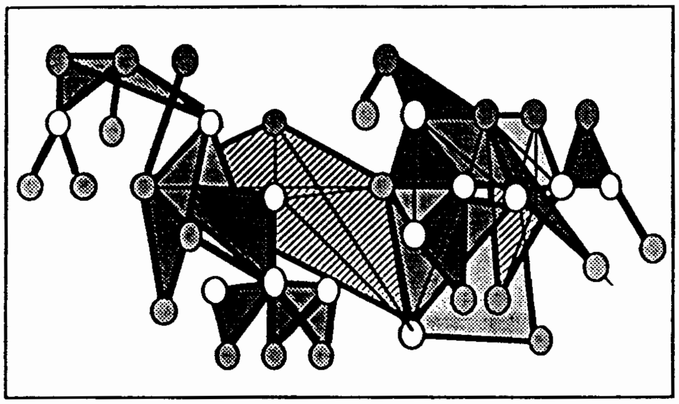


Fig. 7

Spiegelhalter's algorithm re-arranges the ALARM network by triangulation and clique formation. The cliques are shaded differently to make them visible.

Tarjan and Yannakakis have shown that these cliques satisfy the running intersection property: There exists an ordering of the cliques such that each clique can be assigned a unique parent clique, which summarizes all information from cliques preceding in the ordering [Tarjan 84; Leimer 85]. Thus, all cliques can be rearranged in a clique tree where each clique is linked only to its parent clique, and to a number of children *(Fig.8)*. This treeformation allows evidence propagation through local only operations on the cliques.

The initial clique tree is simplified further once evidence becomes available. Evidence nodes contribute their information to the cliques in which they are contained. The nodes are then removed, reducing the size of the respective cliques or eliminating a clique entirely if it becomes a subset of another clique. Thus, observing evidence simplifies the network and makes inference faster *(Fig. 9)*.

The complexity of evidence propagation using this algorithm is linear in the number of cliques, and is exponential in the size of the largest clique in the network. Evidence propagates to all cliques, going from one clique to the next by looping through all possible combinations of values of nodes in a clique.

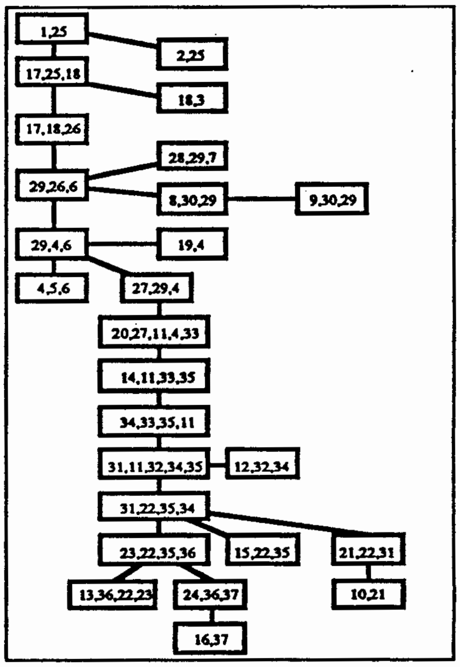


Fig. 8

The ALARM network converted to a clique tree: Each clique is shown as a set of node numbers

Since the number of these combinations is the product of the number of nodes in a clique, propagation from one clique to the next is exponential with respect to clique size. The clique size is in turn dependent on the connectivity of the network: Nodes with a large number of parents will generate large cliques.

The size of the largest clique is dominant in determining the performance of the algorithm. In the ALARM network, the largest cliques contains five nodes and there is a total of 26 cliques. This small maximum clique size makes the network particularly suitable for inference with the Lauritzen-Spiegelhalter algorithm; the average propagation time for a set of measurements is 3 seconds.

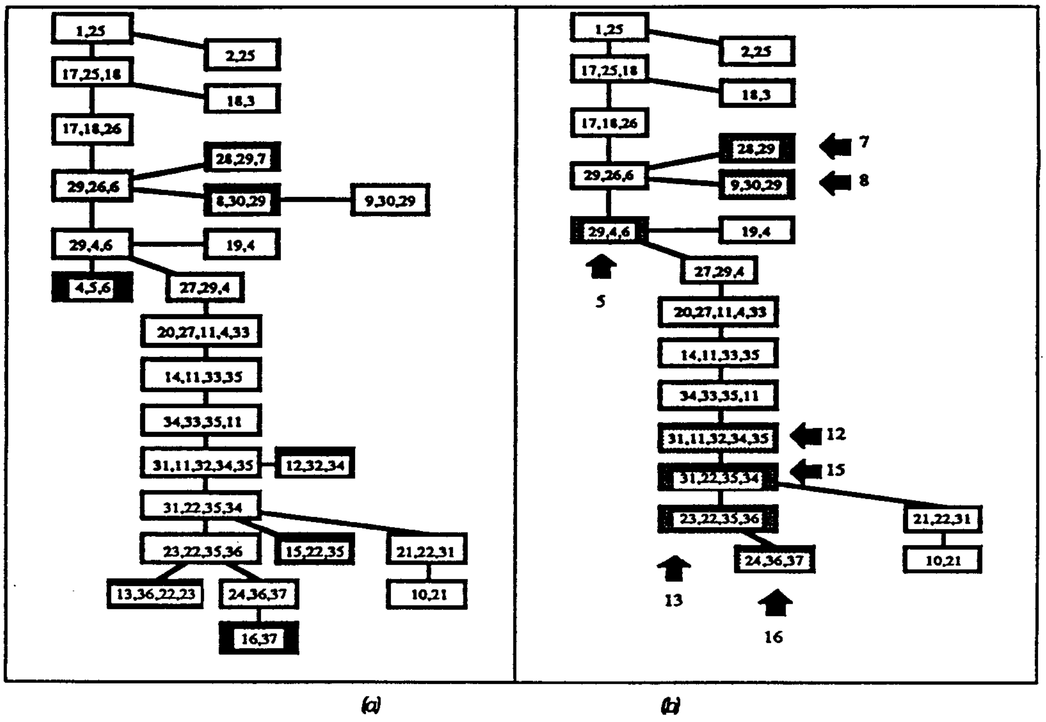


Fig. 9. (a) Evidence for nodes 5, 7, 8, 12, 13, 15 and 16 becomes available. (b) This information is absorbed, and the clique tree is simplified before updating. As a result, the algorithm becomes faster as the set of evidence grows.

Discussion

Belief networks are an increasingly prevalent framework for representing probabilistic knowledge. They have been used successfully to calculate the probability of genetic diseases [Spiegelhalter 88] or to interpret clinical tests [Andreassen 87]. One criticism of them, however, is the prohibitive growth in inference complexity as the size of the network increases. The ALARM network provided us with a case to illustrate these problems in a real-world domain.

The differences among algorithms are significant. Updating the network with Pearl's algorithm takes approximately 8 minutes for a typical set of measurements.

On the other hand, evidence propagation with the Lauritzen–Spiegelhalter algorithm is completed in 3 seconds for the same set of findings. Characteristics of the network topology and the handling of loops determine this behavior.

The performance of Pearl's algorithm on the ALARM network can be explained by the size of the cutset, the large number of measurements available at any time, and the peripheral location of the evidence nodes. The high number of dependencies in this network mandates multiple connections between nodes. To render the network singly connected by the method of conditioning, the algorithm must break five loops generating a cutset that is impractically large.

In this monitoring domain, whole sets of data, typically nine or more measurements, become available at any point in time. This evidence must be propagated sequentially in the absence of special scheduling techniques. In addition, most measurement nodes in the ALARM network are located peripherally. Their instantiation will not simplify subsequent updating. Instead, these nodes start passing messages from parent to parent after being observed, making evidence propagation longer. The peripheral location of evidence nodes also makes it impossible to use the commonly observed ones as cutset members. We could eliminate a large number of cutset instantiations were that possible.

Ironically, a large set of evidence has the opposite effect for the Lauritzen–Spiegelhalter procedure: Evidence simplifies the clique tree and increases update speeds. The high connectivity of the ALARM network, a problem for the Pearl mechanism, makes the cliques smaller in the case of the Lauritzen–Spiegelhalter algorithm. As there is only a small number of parents per node, the maximum clique size stays within reasonable limits. Higher–order dependencies leading to many parents per node are unlikely as they are only rarely known or assessed; in general, new concepts are introduced to avoid such dependencies.

The choice of an inference mechanism corresponding to the topology of a belief network is crucial for any real-world application. It is important to match the given domain with the inference algorithm that will be most efficient given the specific properties of the domain model.

Acknowledgments

We thank Maria Rivero and David Heckerman for their contribution and feedback. Part of this work was supported by grant NSF IRI-8703710, National Library of Medicine grant LM-04136, US Army Research Office grant P-25514-EL and a gift from the Hewlett Packard Corporation. Computing facilities were provided by SUMEX-AIM resource under National Institutes of Health grant RR-00785.

References

[Andreassen 87] Andreassen, S. K., Woldbye, M., Falck, B., Andersen, S. K., MUNIN - a causal probabilistic network for interpretation of electro-myographic findings. In *Proc. 10th International Joint Conference on Artificial Intelligence, Milan*, pp. 366-372, 1987.

[Chavez 88] Chavez, R. M., and Cooper, G. F., KNET: Integrating hypermedia and normative Bayesian modeling. *Proceedings of the Fourth AAAI Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN, pp. 49-54, 1988.

[Darroch 80] Darroch, J. N., Lauritzen, S. L., and Speed, T. P., Markov fields and log-linear models for contingency tables, *Annals of Statistics*, 8, pp. 522-539, 1980.

[Gavril 72] Gavril, T., Algorithms for minimum coloring, maximum clique, minimum coloring by cliques and maximum independent set of a choral graph. *SIAM J Comput*, 1, pp. 180-187, 1972.

[Howard 84] Howard, R. A., and Matheson, J. E., *Readings on the Principles and Applications of Decision Analysis*, 2nd edition. Strategic Decision Group, Menlo Park, CA, 1984.

[Lauritzen/Spiegelhalter 88] Lauritzen, S. L., and Spiegelhalter, D. J., Local Computations with probabilities on graphical structures and their application to expert systems. *J R Statist Soc B*, 50:2, pp. 157-224, 1988.

[Leimer 85] Leimer, H. G., Strongly decomposable graphs and hypergraphs. *Thesis 85-1*, Bereich für Stochastik und verwandte Gebiete., Univ. of Mainz, 1985.

[Pearl 86a] Pearl, J., A constraint-propagation approach to probabilistic reasoning. In: Kanal, L.N. and Lemmers, J.F., eds., *Uncertainty in Artificial Intelligence*, Elsevier Science Publishers, Amsterdam, The Netherlands, pp. 357-369, 1986.

[Pearl 86b] Pearl, J., Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29, pp.241-288, 1986.

[Pearl 87] Pearl, J., Distributed revision of composite beliefs. *Artificial Intelligence*, 33, pp. 173-215, 1987.

[Shachter 86] Shachter, R. D., Evaluating influence diagrams. *Operations Research*, 34, pp. 871-882, 1986.

[Shachter 87] Shachter, R. D., and Heckerman, D. E., Thinking backward for knowledge acquisition. *AI Magazine*, Fall 1987, pp. 55-61, 1987.

[Spiegelhalter 87] Spiegelhalter, D. J., Coherent evidence propagation in expert systems, *The Statistician*, 36, pp. 201-210, 1987.

[Spiegelhalter 88] Spiegelhalter, D. J., Fast algorithms for probabilistic reasoning in influence diagrams, with applications in genetics and expert systems, *Proceedings of the Conference on Influence Diagrams for Decision Analysis, Inference, and Prediction*, Berkeley,CA, May, 1988.

[Suermondt 88] Suermondt, H. J., and Cooper, G. F., Updating probabilities in multiply-connected belief networks. *Proceedings of the Fourth AAAI Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN, pp. 335-343, 1988.

[Tarjan 84] Tarjan, R.E., and Yannakakis, M., Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce hypergraphs. *SIAM J Comput*, 13, pp. 566-579, 1984