Applied Intelligence 21, 81–97, 2004

A Prototypical System for Soft Evidential Update

YOUNG-GYUN KIM
Department of Mathematics and Computer Science, South Carolina State University, Orangeburg, SC 29117, USA

MARCO VALTORTA
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA
mgv@cse.sc.edu

JIŘÍ VOMLEL
Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, 182 08 Prague 8, Czech Republic

Abstract. Autonomous agents that communicate using probabilistic information and use Bayesian networks for knowledge representation need an update mechanism that goes beyond conditioning on the basis of evidence. In a related paper (M. Valtorta, Y.G. Kim, and J. Vomlel, *International Journal of Approximate Reasoning*, vol. 29, no. 1, pp. 71–106, 2002), we describe this mechanism, which we call *soft evidential update*, its properties, and algorithms to realize it. Here, we describe an implementation of the most promising such algorithm, the *big clique algorithm*, together with examples of its use.

Keywords: Bayesian networks, multiagent systems, evidential updating, iterative proportional fitting procedure (IPFP), soft evidence

1. Introduction and Motivation

The problem of updating a probability distribution represented by a Bayesian net upon the presentation of soft evidence is called the problem of soft evidential update. In this paper, we describe *BC-Hugin*,[1] a program that implements soft evidential update (for Bayesian networks). A companion article by the authors [1] contains theoretical and methodological preliminaries and should be read in conjunction with this paper.[2]

The motivation for this work is our desire to let agents that use probabilistic models (and especially Bayesian nets) communicate with each other by exchanging beliefs.

While this is not the focus of this paper, we need to describe briefly our agent model, which is called the Agent-Encapsulated Bayesian Network (AEBN) model, originally due to Bloemeke [2]. Each agent in an AEBN model uses as its model of the world a single Bayesian network (which we also call an AEBN). The agents communicate via message passing. Each message is a distribution on variables shared between the individual networks.

The variables of each AEBN are divided into three groups: those about which other agents have better knowledge (*input set*), those that are only used within the agent (*local set*), and those of which the agent has the best knowledge, and which other agents may want (*output set*). The variables in the input set and the output set are shared, while those in the local set are not. An agent consumes (or subscribes to) zero or more variables in the input set and produces (or publishes) zero or more variables in the output set.

The mechanism for integrating the view of the other agents on a shared variable is to replace the agent's current belief in that variable with that of the communicating agent. When an agent receives a message from

a publisher, it modifies the probabilities in its internal model, so that its local distribution either becomes consistent with the other agent's view or is inconsistent with it.

Therefore, except for a zero-probability situation,[3] after updating using all evidence, we still require that all appropriate marginals of the updated distribution be equal to the evidence entered. The deservedly celebrated junction tree algorithm for probability update [3–6] was not designed to satisfy this requirement, and in fact it does not, as we will show in Section 3.1.

When a publisher makes a new observation, it sends a message to its subscribers. In turn, the subscribers adjust their internal view of the world and send their published values to their subscribers. Assuming that the graph of agent communication (which we simply call *agent graph* as in [2]) is a DAG, equilibrium is reached, and a kind of global consistency is assured, because the belief in each shared variable is the same in every agent.

The restriction that one of the agents has oracular knowledge of a variable may seem excessive. However, it is permissible to have multiple views of a common variable. For example, in a multiagent system for interpretation, two agents may issue a report that corresponds to the same (unknown) physical quantity. Nothing prevents another agent from integrating the reports of these agents and effectively obtain a new (and possibly more accurate) view of the same underlying quantity. As another example, it is possible for a subscriber agent to model known reporting errors or biases of the publisher, as we will show in Section 2.2.

In the special case in which the agent graph is a tree, there is at most only one directed path from one agent to another. In the general case, the agent graph is not a tree, and there may be multiple directed paths from one agent to another. Such multiple paths lead to possible double counting of information, which is often known as the rumor problem. We do not address this important problem in this paper, but cf. [2]. It is also possible to consider directed cycles (and in particular, the tight cycles resulting from bi-directional communication in agent graphs), by appropriately sequencing messages between agents. We do not address this extension further in this paper, but the reader must be made aware of the very interesting and important work by Xiang on Multiply Sectioned Bayesian networks for related results [7, 8].

The rest of paper is organized as follows. In Section 2, we explain the notion of soft evidence with an example that also illustrates the AEBN model (in Section 2.2). Section 3.1 is devoted to a discussion of why the junction tree method does not support soft evidential update. We than present a modification of the junction tree algorithm, the *big clique algorithm*, that supports soft evidential update. In Section 4, we describe the big clique algorithm and its implementation. In Section 5, we test the implementation on a suite of problems, verify that it works correctly, and discuss the results obtained. Section 6 contains a summary and evaluation of our work and suggestions for future work.

2. Soft Evidence

2.1. Evidence

Evidence is a collection of findings on variables. A finding may be hard or soft. A *hard finding* specifies the value of a variable. A *soft finding* specifies the probability distribution of a variable. *Hard evidence* is a collection of hard findings. *Soft evidence* is a collection of soft findings.

The correct processing of soft evidential update requires the introduction of special *observation variables*. The definition of soft evidence could be generalized in three ways. Firstly, we may extend the definition of finding to allow conditional distributions. Secondly, we may allow joint (and possibly, conditional) distributions on a collection of variables. Thirdly, we may allow distributions on arbitrary events (equivalently, arbitrary logic formulae). These three extensions can also be handled by the introduction of observation variables as shown in [1].

2.2. An Example

We extend the cow pregnancy network of [5] to a three-agent system. One of the agents represents a farmer who needs to evaluate the probability that one of his or her cows is pregnant. The other agents represent a Urine Test (UT) expert and a Scanning Test (ST) expert, respectively.

The farmer subscribes to variable UT, which is published by the UT expert, and to variable ST, which is published by the ST expert, as indicated in Fig. 1. While we require that the distribution of a variable remain the same across agents, nothing prevents the farmer from having a model of the experts and therefore somehow discounting their advice, as indicated in Fig. 2, where a

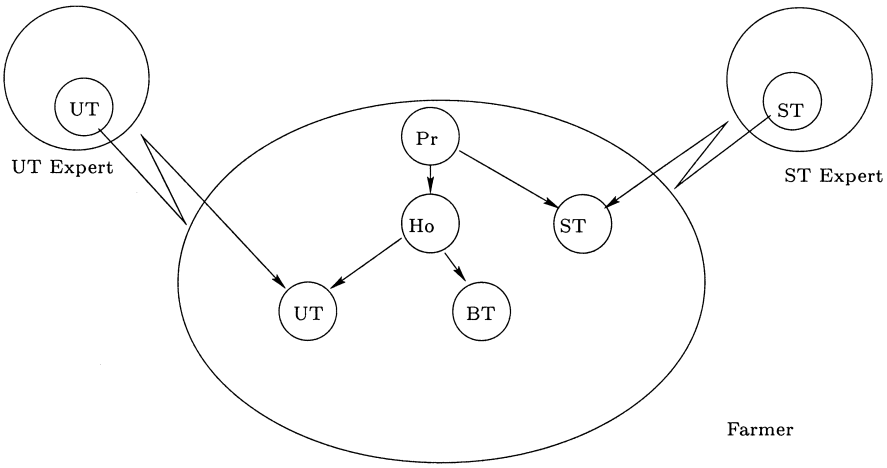


Figure 1. The agent graph of a three-agent system.

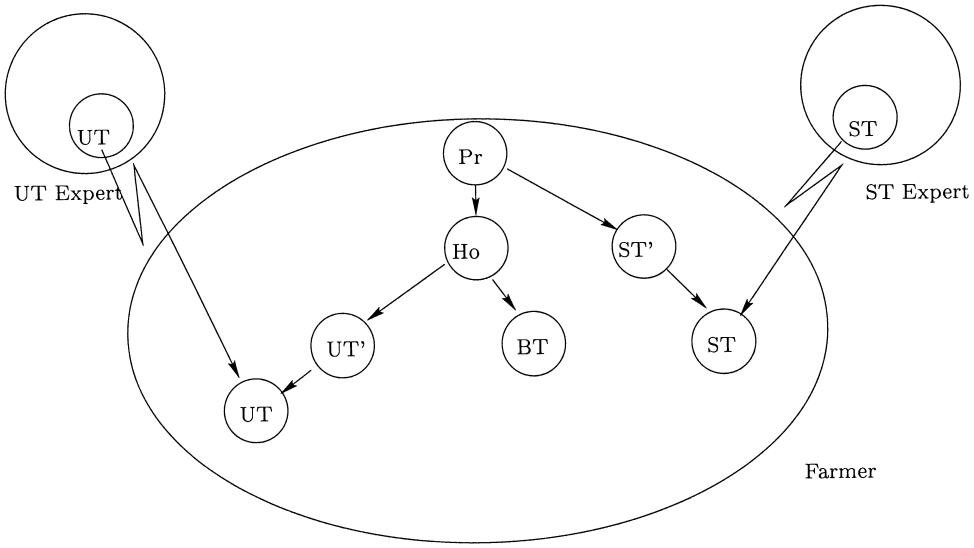


Figure 2. Agent graph of Fig. 1 with simple models of the experts.

simple model of the reliability and sensitivity of the two experts is encoded in the link between the primed variables (UT' and ST'), which represent the farmer's view of the results of the Urine and Scanning tests, respectively, and the unprimed variables (UT and ST), which are the test results as communicated by the experts. The farmer possesses a more complicated model of the two experts in the situation described in Fig. 3, where it is assumed that the farmer knows that the expert's advice is affected by some environmental factor E.

3. Soft Evidence Problem

3.1. Why Do the Classical Propagation Methods Fail?

We show, by a simple example, that the junction tree algorithm does not treat soft evidence properly.

The skeleton of the argument is as follows. In the junction tree algorithm, messages are passed across separators from clique to clique. Exactly two messages are passed between two cliques (say, C_i and C_j), one

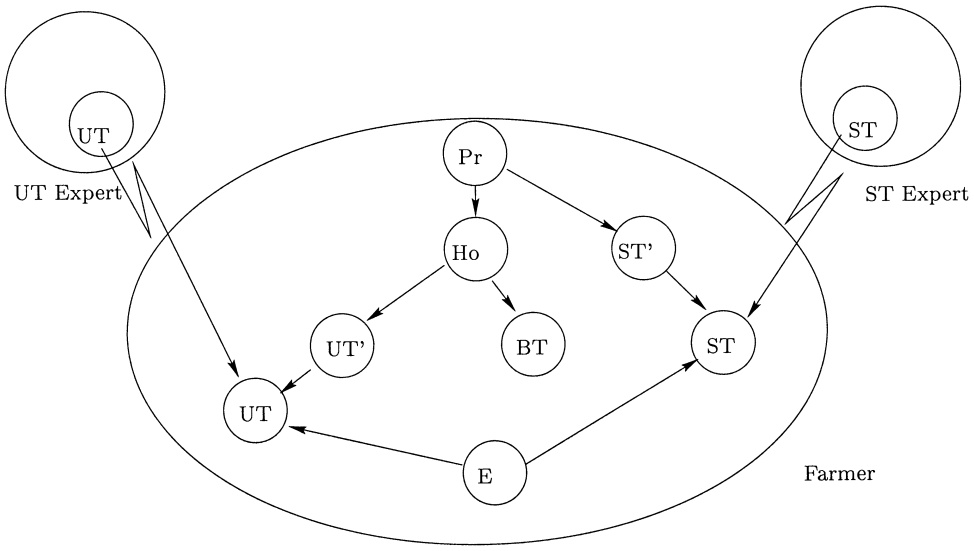


Figure 3. Agent graph of Fig. 1 with complex models of the experts.

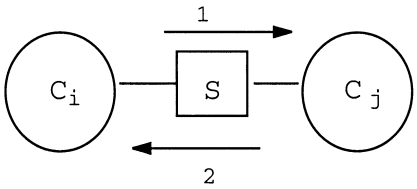


Figure 4. The soft finding $P(V_i)$ is not treated as evidence.

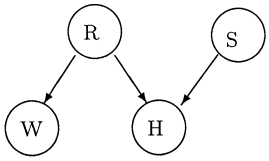


Figure 5. The wet grass Bayesian network structure.

in each direction, as shown in Fig. 4. The first message is passed during the *DistributeEvidence* phase of the method (say, from C_i to C_j), the second during the *CollectEvidence* phase (say, from C_j to C_i). Suppose that clique C_i contains a node (say, V_i) for which we have a soft finding (say, $P(V_i)$), When C_i sends its message to C_j, $P(C_j)$ in modified (by calibration). After the probability of C_j has been fully updated (say, to $Q(C_j)$), C_j sends its message to C_i, and the probability of C_i is modified by calibration. In general, letting $Q(C_i)$ be the modified probability, we have that $\sum_{C_i \setminus \{V_i\}} Q(C_i) \neq P(V_i)$, which implies that the soft finding $P(V_i)$ is not treated as evidence.

We now present the promised example that establishes our claim that the junction tree algorithm does not handle soft evidence properly. Consider the wet grass example, shown in Fig. 5 [5, p. 23], and suppose that our hard evidence is that Holmes' lawn is certainly dry (i.e, $H = n$), while we have soft evidence that Watson's lawn, in the form $P(W) = (0.7, 0.3)$. If this soft evidence is absorbed in clique *WR* and the hard evidence is absorbed in clique *HRS*, propagation will consist of two messages through the separator R, as shown in Fig. 6. After the messages are passed, P is updated to Q, and $Q(W) = (.4439, .5561) \neq (.7, .3) = P(W)$.

We have shown that even if soft findings are absorbed correctly into cliques, the propagation method itself would not respect the evidence characteristics of the findings.

The reason is that it is not possible to enter new evidence that contradicts the evidence already entered, and therefore zero entries are treated in a special way by the junction tree algorithm: zeroes in probability tables remain zeroes after each message (cf. [5, Lemma 4.1).

Jeffrey's rule, also known as the rule of probability kinematics, provides a way to update a probability distribution from soft (uncertain, non-categorical, non-propositional) evidence.

Jeffrey's rule can be written as:

$$Q(A) = \sum_i P(A \mid B_i) \cdot Q(B_i)$$

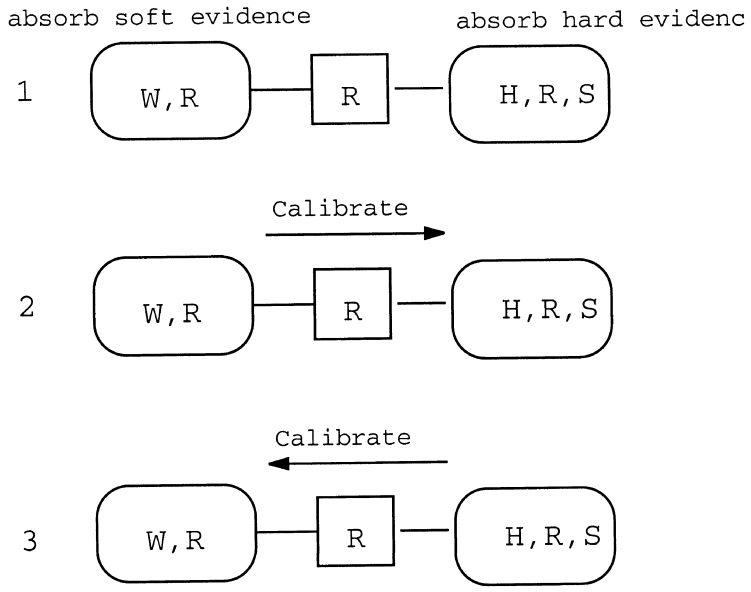


Figure 6. The junction tree algorithm and the wet grass network.

where $Q(B)$ is soft evidence, and $P(A \mid B)$ is the conditional probability of A given B *before evidence*. Jeffrey's rule applies in situations in which $P(A \mid B)$ is invariant w.r.t. $P(B)$ but leads to errors when this condition does not hold. The reader is referred to [9, 10] and [1] for a discussion of the rule and its deficiencies.

An alternative method of updating in the presence of uncertain evidence is the *virtual evidence* method [9–12]. The virtual evidence method takes the position that likelihood ratios are stabler than probability distributions [9, Section 2.3.3]. Pearl [9, 10] observes that the virtual evidence method can be viewed as formally equivalent to the likelihood ratio version of Jeffrey's rule. Suppose that we obtain evidence on a variable B, with values b_i. Letting $Q(b_i)$ be the evidence and $P(b_i)$ be the prior probability of B, the ratio $Q(B)/P(B)$ may be used in the virtual evidence method, resulting in an updated distribution whose marginal over B is the soft evidence $Q(B)$. As Pearl [10, p. 71] explains, "beliefs updated by Jeffrey's rule cannot be distinguished from those updated by Bayes conditionalization on *some* [our emphasis] virtual evidence." Please see [1, Section 3.2] for further discussion of the difference between the virtual evidence method and soft evidential update and the end of Section 5.2 for an example contrasting the two. From a methodological viewpoint, it is important to note that, unlike mechanical applications of maximum-entropy methods, the soft evidential update method preserves the independence structure captured in a Bayesian network and therefore avoids paradoxical results noted by several authors (e.g., [13]). We refer to [1] and [14, Theorem 3.3] for a precise discussion of this point.

4. Big Clique Algorithm

Here, we suggest a soft evidence absorption algorithm. This algorithm combines two methods: junction tree propagation and Iterative Proportional Fitting Procedure (IPFP).

4.1. Algorithm

The *big clique algorithm* modifies the junction tree algorithm as follows:

1. Build a junction tree that includes all variables for which soft evidence is given in one clique, the *big clique* C_1. (These variables may appear in other cliques as well.)
2. Update $P(V)$ to a distribution $P^*(V)$ by executing the junction tree algorithm using only hard evidence. $P^*(V)$ is a distributed representation of $P(V \mid \text{hard evidence})$, in the sense of the remark following Theorem 4.2 in [5]: the product of all clique tables divided by the product of all separator tables is equal to $P(V \mid \text{hard evidence})$.

3. Absorb all soft evidence in C_1 (with the algorithm described in Section 4.4).
4. Call the routine DistributeEvidence from C_1. This routine and the correctness of this step are presented in Section 4.3.

4.2. Construction of the Junction Tree

In order to insure that all variables for which soft evidence is given (*soft evidence nodes*) belong to the big clique, the following steps are executed:

1. moralize the Bayesian network graph;
2. add edges between each pair of soft evidence nodes;
3. triangulate.

The second step is unique to the big clique algorithm. Moralization and triangulation may be performed in the usual way. In particular, any of the many heuristic or approximate algorithms for triangulation may be used (see, e.g., [15, 16]). The implementation described in this paper uses a very simple heuristic, because performance is not the main concern of this research.

4.3. Propagation of Soft Evidence

First, recall that Step 2 in the modified junction tree algorithm leads to a distributed representation of the posterior probability of all variables given all hard findings. Second, observe that the product of the table for the special clique that has absorbed all soft evidence (as done in Step 3) multiplied by the tables for the other cliques and divided by the tables of the separators is a representation of the posterior probability of all variables given the soft evidence and the hard evidence. (Hard evidence had already been absorbed in Step 2).

We now need to restore consistency between the special clique and the other cliques. To do so, we propagate from the clique that has absorbed soft evidence using the Hugin *DistributeEvidence* algorithm, which is described in [5]. This algorithm has three important properties: (1) it updates the probability tables of the other cliques while it maintains the invariant that the product of the clique tables divided by the separator tables is equal to the joint probability table for all variables in the Bayesian network; (2) it insures local consistency and (Theorem 4.5 in [5]) global consistency; (3) it does not disturb hard findings, because it does not remove zeros. (It may introduce new zeros in special cases.)

Finally, observe that the table for the clique that contains all variables for which we have soft findings in unchanged by a *DistributeEvidence* call that starts at that clique. Therefore, the result of propagation is to obtain a globally consistent distributed representation of the posterior in which all findings, hard and soft, hold.

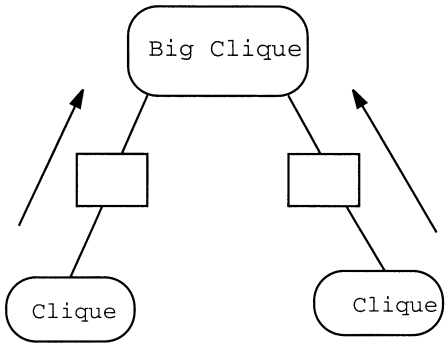


Figure 7. The big clique calls *CollectEvidence*.

We remark that the big clique algorithm could be simplified by removing the *DistributeEvidence* part from the second step. In other words, it is sufficient to carry out one *CollectEvidence* operation to the special clique (using only hard evidence) and one *DistributeEvidence* from the special clique (after absorbing soft evidence in the special clique). See Fig. 9 for an example: there is only one *DistributeEvidence* operation, in Step 4, after absorption of soft evidence in Step 3. From this point on, we redefine the big clique algorithm to be this simplified version. Figures 7 and 8 illustrate the *CollectEvidence* and *DistributeEvidence* operations. Figure 9 illustrates the operation of the whole big clique algorithm on the lawn example. In this special case, the junction tree is the same as that constructed by the junction tree algorithm (ef. Fig. 6), but the order of operations is different. In particular, note how the absorption of soft evidence is delayed.

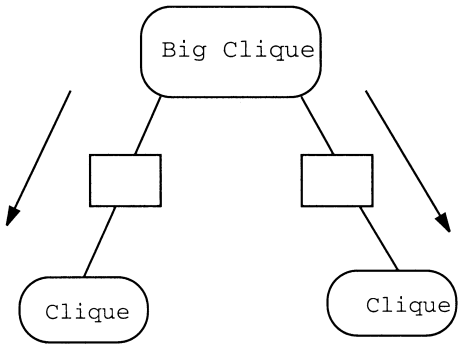


Figure 8. The big clique calls *DistributeEvidence*.

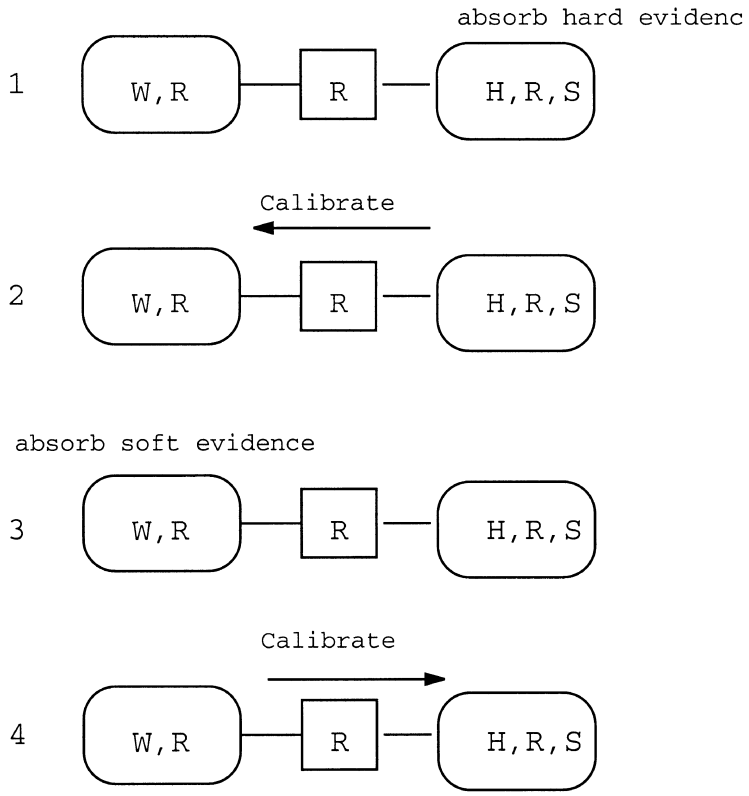


Figure 9. Operation of the big clique algorithm on the lawn example.

4.4. *Absorption of Soft Evidence*

We define absorption in the special big clique C_1 as the process by which the joint probability $P(C_1)$, is updated to conform to soft evidence on variables $A \subseteq C_1$, where $A = \{A_1, A_2, \ldots, A_k\}$.

Let $Q(C_1)$ be the joint probability after absorption. Then $\forall\, i \sum_{C_1 \setminus A_i} Q(C_1) = P(A_i)$, where $P(A_i)$ is the soft evidence on A_i, $i = 1, \ldots, k$. Absorption of soft evidence in clique C_1 is carried out by using the Iterative Proportional Fitting Procedure (IPFP) and consists of cycles of k steps, one per finding. Each step corresponds to one soft finding. The appropriate formulae are:

$$Q_{(0)}(C_1) = P(C_1)$$
$$Q_{(i)}(C_1) = \frac{Q_{i-1}(C_1) \cdot P(A_j)}{Q_{i-1}(A_j)}$$

where $j = ((i-1) \bmod k) + 1$.

For a simple example, suppose we have the clique $\{A, B\}$ with joint probability as given in Table 1 (all variables are binary). Suppose that soft evidence on variable B is available in the form of $P(B) = (.7, .3)$. We compute the updated joint probability $Q(B)$ in two steps.

Table 1. Table for $P(A, B)$.

	A	
B	y	n
y	.56	.03
n	.14	.27

Table 2. Table for $P(A, B, e)$.

	A	
B	y	n
y	.392	.021
n	.042	.081

The result of the multiplication by $P(B)$ is in Table 2. The result of the division by $Q_{(0)}(B) = (.59, .41)$ is in Table 3. Note that $\sum_{\{A,B\}\setminus\{B\}} Q(A, B) = P(B) = (0.7, 0.3)$, as claimed.

Table 3. Table for $P(A, B, e)$ after normalization.

	A	
B	y	n
y	.664	.036
n	.102	.198

One step of IPFP is sufficient when there is only one soft finding. In general, however, several cycles may be necessary for IPFP to converge. See [15, 17, 18] for the proof of convergence in the general discrete case and for bounds on the number of cycles in special cases.

More generally, if all pairs of distinct observation variables are independent in the original distribution then they are independent in the updated distribution as well, and the soft evidential update method requires only one cycle to converge, i.e. only n steps are sufficient (see [1] for details).

In AEBN systems, soft observations are messages from publishing agents and may be dependent when the AEBN graph is not a tree. The correct modeling of dependence in the receiving agent requires knowledge of the AEBN agent graph. A full treatment of this aspect is beyond the scope of this paper, but we mention Bloemeke's work on this topic [2].

4.5. Implementation: BC-Hugin

In order to support the big clique algorithm, we implemented in Java BC-Hugin, which is an extension of the Hugin system (see Fig. 10 for the BC-Hugin introductory screen). As mentioned earlier, the big clique algorithm requires new methods, such as creating a big clique and soft evidence absorption, that are not supported by the junction tree propagation that is already implemented by the Hugin API. Also the Hugin API does not allow us to control basic methods that are necessary for the new methods.

Therefore, we implemented BC-Hugin from scratch. The creation of a junction tree and the propagation method was implemented by following Jensen's algorithms [5]. BC-Hugin reads *net* files or *hkb* files that were created by Hugin. The Hugin API (version 5.1) is used to read and load these files.[4] An example of a file open menu window is presented in Fig. 11. BC-Hugin propagates the evidence (hard or soft) that is entered, thereby computing the marginal posterior probability of every variable.

4.5.1. Example: Flood. This example is adapted from Jensen [5]. Figure 12 shows the given flood model, and we set the *rain* node as the only soft evidence node. Figure 13 shows the junction tree created by BC-Hugin. Clique 1 on the top of the tree is the big clique that includes all soft evidence nodes. The value window (in Fig. 14) is designed to provide visual results for each propagation. A hard finding is represented by a radio button while a text-box represents a soft finding.

5. Evaluation

Soft evidential updating can be formalized as a constrained optimization task. The goal is to find a probability distribution such that

1. It satisfies all the constrains introduced by different types of soft evidence.
2. It optimizes a chosen criterion among all distributions satisfying these constraints.

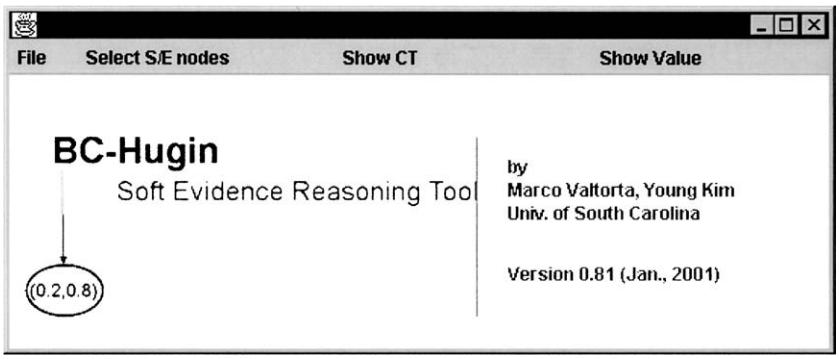


Figure 10. Opening window for BC-Hugin.

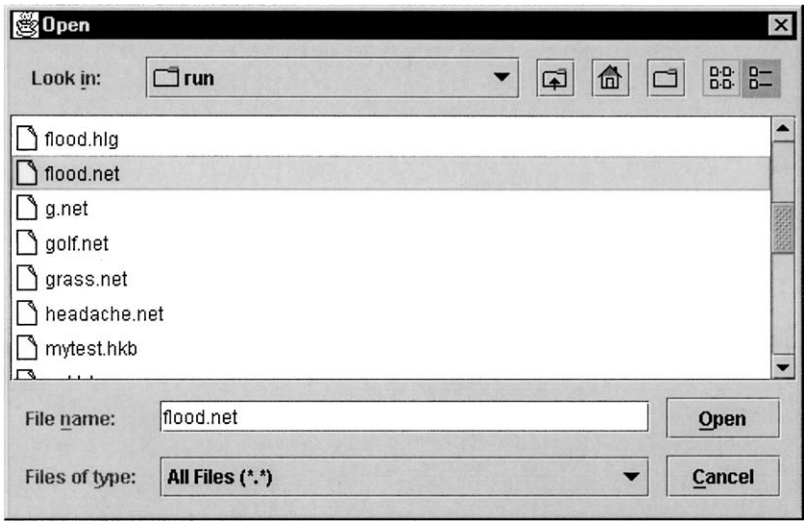


Figure 11. File open window for BC-Hugin.

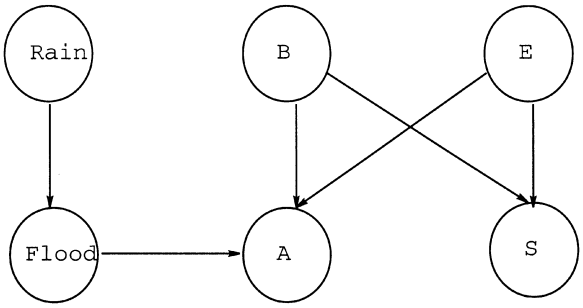


Figure 12. Flood example.

It can be proved that the big clique algorithm computes a distribution that both satisfies all the constraints introduced by soft evidence and is closest to the original distribution. See [1] for details and proofs.

The evaluation of BC-Hugin consists of two steps:

1. Testing the hard evidential absorption of BC-Hugin: we use only hard evidence and compare results with Hugin.
2. Testing the soft evidence absorption: we create results with hard and soft evidence.

5.1. *Testing Hard Evidence Absorption*

In this test, we test the hard evidential absorption of BC-Hugin by comparing results with Hugin. We use the junction tree propagation algorithm for hard evidential updating. Thus, BC-Hugin with only hard evidence must produce the same results as Hugin. For the test, we used the input files listed in Table 4, and those files were provided by the Hugin system. For each test file, we created a full combination of findings for each node, propagated and compared results. For example, if there are five nodes in a test file and each node has two states, the total number of combination of findings will be 3^5 by considering the absence of a finding as a state. We found that even though BC-Hugin was much slower than Hugin for large networks (such as the simple poker example), BC-Hugin produced correct values, sometimes with minor numeric differences. The biggest numeric differences for each test file are listed in Table 4. The magnitude of the relative error ($\frac{(\alpha - a)}{\alpha}$, where α is the correct value as computed by Hugin and a is the "approximation" computed by BC-Hugin) is typically small.

5.2. *Testing the Dependency of Soft Evidence*

In hard evidence absorption, each item of hard evidence is independent of the others, and the joint probability of hard evidence is represented by the multiplication of the probability of each item of evidence. On the other hand, as explained in Section 4.4, there exists a dependency relationship between items of soft evidence when the corresponding variables are dependent in the model of the agent that receives the evidence.

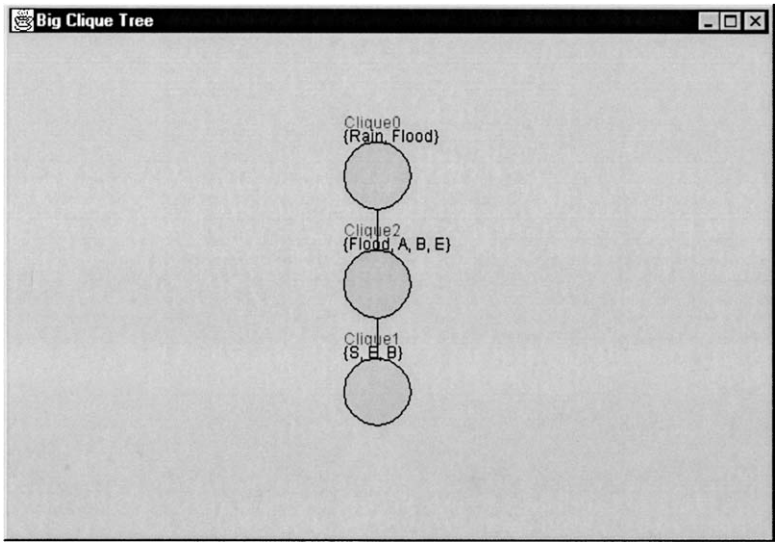


Figure 13. BC-Hugin clique tree window for flood example. The node *Rain* is the soft evidence.

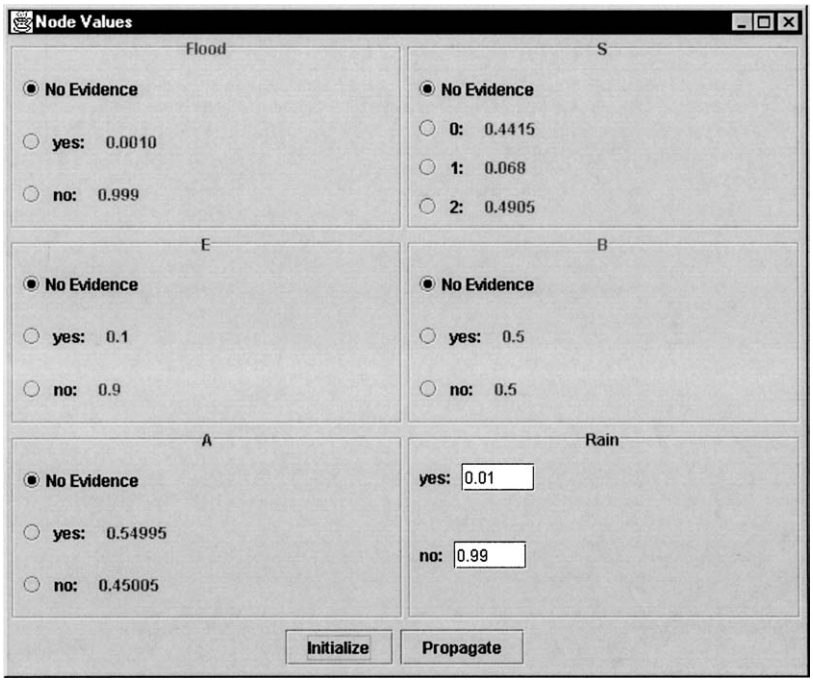


Figure 14. BC-Hugin node window for flood example. The node *Rain* is the soft evidence.

Two examples based on the stud farm model [5, Section 3.2.1] follow. See Fig. 15 for the Bayesian network structure. In both examples, hard evidence is entered for node "John", and soft evidence is entered for nodes "Ann" and "Eric". The clique tree built by BC-Hugin for this situation is shown in Fig. 16.[5] The initial marginal probabilities before entering evidence are displayed by BC-Hugin in the window shown in Fig. 17.

Suppose that we enter the hard evidence that "John" is sick. The marginal posterior probabilities that "Ann" is a carrier and that "Eric" is a carrier increase to 0.6236 and 0.3862, respectively. When there is hard

Table 4. Test file list.

Test name	File name	Variables/ Number of instantiations	Largest error
Family out	family out.net	5/243	4.9471855E-6
Wet grass	wet grass.net	4/81	4.708767E-6
Visiting asia	asia.net	8/6561	5.185604E-6
Stud farm (Fig. 15)	stfrm.net	12/708588	5.364418E-6
Flood	flood.net	6/972	5.1259995E-6
Mrs Gibbon	mrs gibbon.net	5/243	4.827976E-6
Simple Poker (Exercise 3.5(i) in [9])	ex-3.5-i.net	5/18000	
Simple Poker (Exercise 3.5(ii) in [9])	ex-3.5-ii.net	7/486000	5.066395E-6

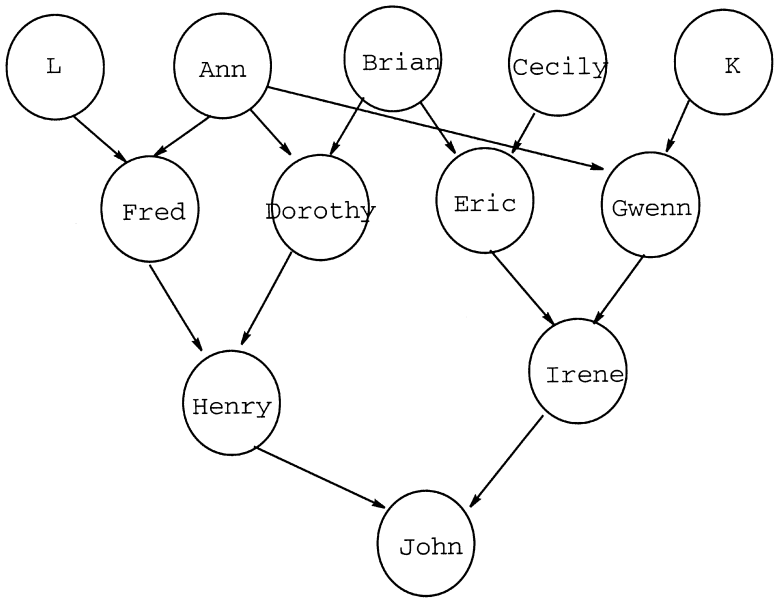


Figure 15. Stud farm model.

evidence that "John" is sick, its ancestors "Ann" and "Eric" become d-connected and therefore[6] dependent, because only "John" is instantiated in the chain ⟨"Ann", "Dorothy", "Henry", "John", "Irene", "Eric"⟩. Therefore, when we enter (0.5, 0.5) as soft evidence into "Ann" and "Eric" respectively, and do a propagation, the resulting joint probability for $P(Ann, Eric)$ shows dependency of soft evidence, as displayed in Table 5. $P(Ann) \times P(Eric)$ is 0.25 for each combination of the values of Ann and Eric, while the joint probabilities, as computed using BC-Hugin, reflect a strong preference for single faults. For example, $P(Ann = pure, Eric = carrier)$ is much greater than $P(Ann = carrier, Eric = carrier)$. It is easy to show (by algebraic manipulation) that $P(Ann = pure, Eric = pure)$ must be equal to $P(Ann = carrier, Eric = carrier)$ and $P(Ann = pure, Eric = carrier)$ must be equal to $P(Ann = carrier, Eric = pure)$.

Table 5. $P(Ann, Eric)$ and $(P(Ann) \times P(Eric))$ for soft evidence (0.5, 0.5).

	$Ann = pure$	$Ann = carrier$
$Eric = pure$	0.0001 (0.25)	0.4999 (0.25)
$Eric = carrier$	0.4999 (0.25)	0.0001 (0.25)

Table 6 shows similar results for the situation in which the soft evidence (0.8, 0.2) is entered for both

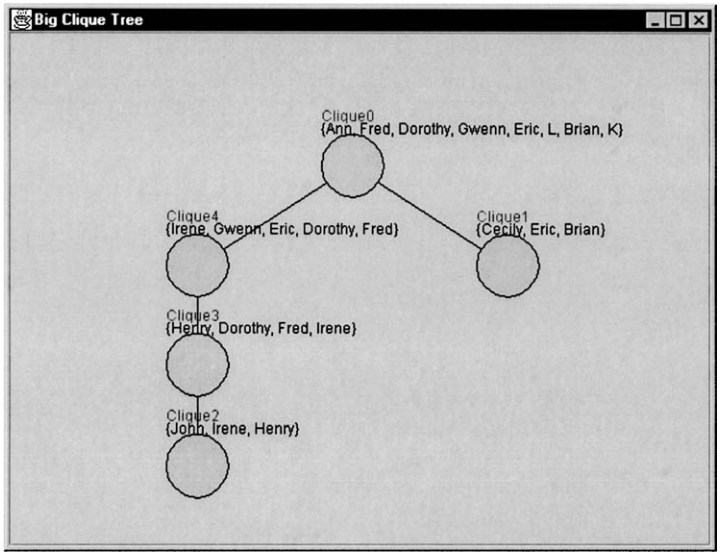


Figure 16. BC-Hugin clique tree window for stud farm example. There is soft evidence for *Ann* and *Eric*.

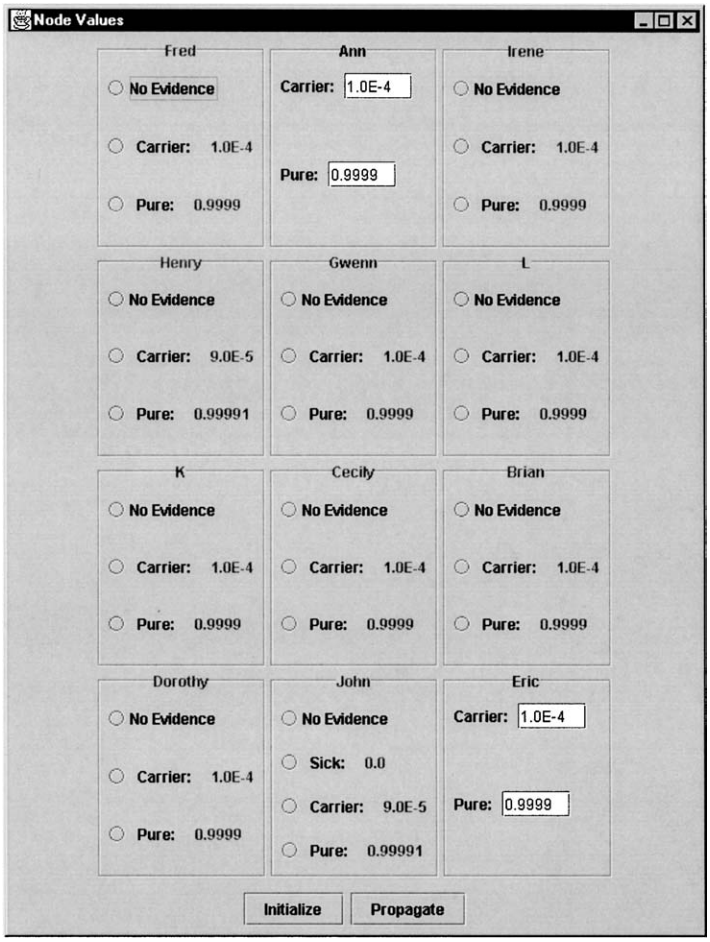


Figure 17. BC-Hugin probability input window for stud farm example, before entering evidence.

Table 6. $P(Ann, Eric)$ and $(P(Ann) \times P(Eric))$ for soft evidence (0.8, 0.2).

	$Ann = pure$	$Ann = carrier$
$Eric = pure$	0.6000 (0.64)	0.2000 (0.16)
$Eric = carrier$	0.2000 (0.16)	0.0000 (0.04)

Table 7. $P(Ann, Eric)$ computed using the soft evidence method and $(P(Ann, Eric)$ computed using the virtual evidence method), with evidence (0.5, 0.5).

	$Ann = pure$	$Ann = carrier$
$Eric = pure$	0.0001 (0.0152)	0.4999 (0.6048)
$Eric = carrier$	0.4999 (0.3710)	0.0001 (0.0054)

"Ann" and "Eric". In this case, $P(Ann = carrier, Eric = carrier)$ is much smaller than the value (0.04) obtained by assuming independence. Analogously to the previous case, it is easy to show that $P(Ann = pure, Eric = carrier)$ must be equal to $P(Ann = carrier, Eric = pure)$.

Table 7 compares the results obtained by the virtual evidence and soft evidence methods when the evidence for Ann and Eric is $(0.5, 0.5)$. The virtual evidence interpretation is that we do not know anything new, whereas the soft evidence interpretation is that the probability that Ann is a carrier is $\frac{1}{2}$, and the probability that Eric is a carrier is $\frac{1}{2}$, which is something very different and, in this case at least, clearly appropriate.

5.3. *Testing Soft Evidence Absorption 1: Comparing Hard with Soft Evidence*

The main purpose of this test is to observe the behavior of BC-Hugin when soft evidence is entered. Since hard evidence is a special case of soft evidence, when we enter 0 and 1 as soft evidence, the result from BC-Hugin's soft evidential absorption must be same as the result from Hugin with the findings entered as hard evidence. Note that the evidence was entered in BC-Hugin as soft evidence, even though it could have been entered as hard evidence, as it should be apparent from the BC-Hugin probability input windows. Therefore, this test is different from the one of Section 5.1. We executed this test with the files in Table 4 and obtained correct results. Table 8 shows one example from stud farm. We chose to display the updated probability of three nodes ("Henry," "Ann," and "John") among the 12 nodes in the model. For each case, the hard evidence (step a) and soft evidence (step b) for "Irene" are entered and compared, and BC-Hugin produced correct results.

5.4. *Testing Soft Evidence Absorption 2: Observation with Soft Evidence*

In this test, we demonstrate the operation of BC-Hugin on two examples with soft evidence as well as hard evidence. The first example is the *Wet grass* example of Section 3.1. Since this example is reasonably small, we manually computed each value of nodes with soft evidence in order to verify the correctness of the result. Please note that the value of the soft evidence node ("Holmes") did not change after propagation, which fulfills the requirement of soft evidence absorption.

We entered soft evidence for the "Holmes" node and hard evidence for another node to capture the behavior of soft evidence. In Table 9, when we assigned higher probability for the *yes* state in "Holmes," then "Sprinkler," "Watson," and "Rain" have higher probability for the *yes* state too. This is the correct behavior because the probability for Holmes' grass being wet is influenced by "Sprinkler" and "Rain." When there is hard evidence added to this soft evidence (say there is no rain, case 5), "Watson" has the same value as in case 2, because the node "Watson" is blocked by the hard evidence "Rain". Also "Sprinkler" now is the only influence for

Table 8. Stud farm example for soft/hard evidence update.

Case no.	Evidence (hard or soft)	Ann(A) P(carrier, pure)	Henry(H) P(carrier, pure)	John(J) P(sick, carrier, pure)
1a	Hard: $I = (0, 1)$	(0.0001, 0.9999)	(0.0001, 0.9999)	(0, 0, 1)
1b	Soft: $I = (0, 1)$	(0.0001, 0.9999)	(0.0001, 0.9999)	(0, 0.0001, 0.9999)
2a	Hard: $I = (1, 0)$	(0.25, 0.75)	(0.17, 0.83)	(0.042, 0.5, 0.458)
2b	Soft: $I = (1, 0)$	(0.25, 0.75)	(0.17, 0.83)	(0.042, 0.5, 0.458)

Table 9. Wet grass example with soft evidence node "Holmes".

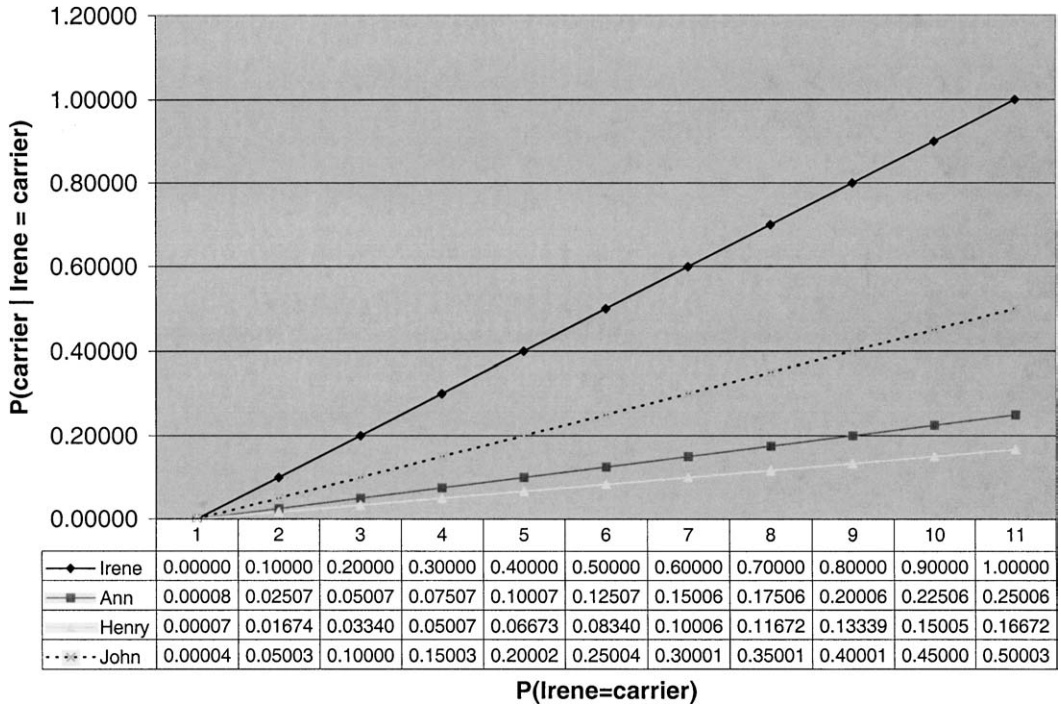
Case no.	Evidence (hard or soft)	Holmes(H) P(yes, no)	Watson(W) P(yes, no)	Sprinkler(S) P(yes, no)	Rain(R) P(yes, no)
1	No evidence	(0.27, 0.73)	(0.36, 0.64)	(0.1, 0.9)	(0.2, 0.8)
2	Hard: $R = (0, 1.0)$	(0.09, 0.91)	(0.2, 0.8)	(0.1, 0.9)	(0, 1.0)
3	Soft: $H = (0.5, 0.5)$	(0.5, 0.5)	(0.49, 0.51)	(0.17, 0.83)	(0.37, 0.63)
4	Soft: $H = (0.8, 0.2)$	(0.8, 0.2)	(0.67, 0.33)	(0.27, 0.73)	(0.59, 0.41)
5	Soft: $H = (0.8, 0.2)$, Hard: $R = (0, 1.0)$	(0.8, 0.2)	(0.2, 0.8)	(0.8022, 0.1978)	(0, 1.0)

"Holmes", the probability distribution is close to that of "Holmes", the soft evidence, because $Q(S = y) = P(S = y \mid H = y) \cdot Q(H = y) + P(S = y \mid H = n) \cdot Q(H = n) = 1 \cdot 0.8 + 0.11 \cdot 0.2 = (0.8022)$. (Since $R = y$, $P(S = y \mid H = y) = P(S \mid H = y, R = y)$ and $P(S = y \mid H = n) = P(S \mid H = n, R = y)$.)

Figure 18 shows the behavior of three nodes in the stud farm model when the soft evidence node ("Irene") has incremental changes. The values for the *carrier* state increase linearly following with the linear increase of the value in "Irene". For the single soft evidence case, this linear increment behavior is correct because soft evidential update follows Jeffrey's rule. In general, this is only true after introducing observation variables, as explained in Sections 2 and 3.1 and, in more detail, in [1]. In this particular example, however, the required conditional probabilities are invariant upon presentation of soft evidence. Recall that Jeffrey's rule is,

$$Q(A) = \sum_i P(A \mid B_i) \cdot Q(B_i)$$

where $Q(B)$ is soft evidence, and $P(A \mid B)$ is the conditional probability of A given B.



	1	2	3	4	5	6	7	8	9	10	11
Irene	0.00000	0.10000	0.20000	0.30000	0.40000	0.50000	0.60000	0.70000	0.80000	0.90000	1.00000
Ann	0.00008	0.02507	0.05007	0.07507	0.10007	0.12507	0.15006	0.17506	0.20006	0.22506	0.25006
Henry	0.00007	0.01674	0.03340	0.05007	0.06673	0.08340	0.10006	0.11672	0.13339	0.15005	0.16672
John	0.00004	0.05003	0.10000	0.15003	0.20002	0.25004	0.30001	0.35001	0.40001	0.45000	0.50003

Figure 18. The linear behavior of nodes in stud farm model.

Since the conditional probability $P(A \mid B)$ is fixed, this formula can be represented by a multivariate polynomial such that the soft evidence is the determining variable. For example, let α_1, α_2 be the conditional probability for $P(John = pure \mid Irene = pure)$ and $P(John = pure \mid Irene = carrier)$ respectively, and Q_1, Q_2 be $Q(Irene = pure)$ and $Q(Irene = carrier)$ respectively. Then $Q(John = pure) = \alpha_1 \cdot Q_1 + \alpha_2 \cdot Q_2 = \alpha_1 \cdot Q_1 + \alpha_2 \cdot (1 - Q_1) = (\alpha_1 - \alpha_2) \cdot Q_1 + \alpha_2$. Since α_1 and α_2 are constant, this function is linear. We note that this is a slight generalization of the polynomial network representation discussed in [19] and based on the following theorem [20] (as stated in [21]): "Let BN be a Bayesian network. Let a be a state of the variable A, let **e** be a set of observations, and let t be a simple parameter in BN. Then, $P(a, \mathbf{e})$ as well as $P(\mathbf{e})$ are linear functions in t."

6. Conclusion

We have described BC-Hugin, a program for soft evidential update that implements the big clique algorithm. We have given several examples of use of BC-Hugin. We have tested BC-Hugin in various ways.

6.1. Complexity Issues

In [1], we describe an alternative algorithm for soft evidential update, based on the space-saving implementation of IPFP [22], which does not require the construction of a big clique but requires iteration over all the cliques. To be more precise, it is necessary to iterate only on cliques that form a path in the junction tree that includes all soft evidence variables.

We illustrate this point and the existence of a trade-off between the two algorithms by examples on a simple class of Bayesian networks. Consider a Bayesian network structure that consists of a single path of n nodes, as in Fig. 19. The junction tree of minimum state-space size for that graph is shown in Fig. 20.

Assume that soft evidence is entered on two nodes that are not adjacent. The big clique built by the big clique algorithm in this case will contain three nodes:

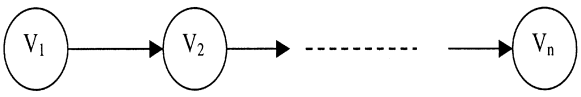


Figure 19. A path of n nodes.

two of them are the soft evidence nodes, and the third one is a node between the soft evidence nodes in the chain. Clearly, the state space for the junction tree built by the big clique algorithm is larger than the state space for the junction tree of Fig. 20, if each variable has more than two values.

With respect to Fig. 19, assume that soft evidence is observed for nodes V_t and V_u, with $u > t + 1$. The algorithm based on the state-space saving implementation of IPFP requires cycling over each clique between V_t, V_{t+1} and V_{u-1}, V_u, i.e., $\{V_t, V_{t+1}\}, \{V_{t+1}, V_{t+2}\}, \ldots, \{V_{u-1}, V_u\}$. The union of these cliques is a superset of the big clique. Still, it is possible for the state-space saving implementation of IPFP to be faster, because each cycle of IPFP requires propagation in a junction tree all of whose cliques contain only two nodes, while the junction tree for the big clique algorithm contains at least one cliques with three nodes.

The precise trade-off depends on the state-space size of the individual variables and on how far apart in the path the variables V_t and V_u are: by making the distance between t and u larger, we force the space saving implementation of IPFP to iterate over a larger number of nodes, while the big clique algorithm will always iterate only over three nodes. We conjecture that this is the dominant factor in the complexity of these procedures; further analysis and experimentation are needed.

6.2. Future Work

The implementation could be tuned for performance. In particular, as described in Section 4.2, the size of the junction tree constructed by BC-Hugin could be greatly reduced by using standard techniques. We need to work on application of soft evidential update to decision nets (influence diagrams). We need to compare more thoroughly the big clique algorithm to the iterative modification of the junction tree algorithm described in [1] and based on the state-space saving implementation of IPFP [22]. We need to study further the possibility or impossibility of algorithms that require neither a big clique nor iteration over all nodes. The rumor problems for agent-encapsulated Bayesian networks needs to be solved. The infrastructure necessary for large-scale applications of soft evidential update needs to be built. See [23] for the design and prototypical implementation of a system for time-critical decision making with communicating agents that uses soft evidential update.

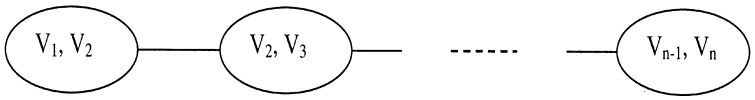


Figure 20. The junction tree of minimum total state-space size for the path network.

Acknowledgments

Y.-G.K. and M.V.'s work has been supported in part by the U.S. Department of Defense through the DARPA Autonomous Negotiating Teams (ANTS) project (AO Number H359 and Contract Number F30602-99-2-0513). M.V. and Y.-G.K.'s work has also been supported in part by the U.S. Office of Naval Research through project DAG (Grant Number N00014-97-1-0806). M.V.'s work was supported in part by the Advanced Research and Development Activity (ARDA), an entity of the U.S. Government. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government. Clif Presser developed the Java bridge to the Hugin 5.1 API that was used to read Hugin files in BC-Hugin. M.V. thanks in a special way Finn V. Jensen and his Decision Support Systems Group in the department of Computer Science at the University of Aalborg for providing a quiet and productive research environment during a sabbatical visit between October 1999 and June 2000. J.V. is grateful to his Ph.D. adviser Radim Jiroušek from the Laboratory of Intelligent Systems at the University of Economics in Prague for comments on several issues related to the paper. M.V. and J.V. thank Finn V. Jensen for comments on matters related to this paper.

Notes

1. We thank Christian Riekher, managing director of Hugin Ltd. for allowing us to use this name for our research prototype. Actual use of Hugin in BC-Hugin is minimal, as explained in Section 4.5.
2. Some introductory material is adapted from [1].
3. BC-Hugin identifies zero probability situations at run time in a way similar to Hugin.
4. Since this version of the API is written in C++, a Java bridge was developed to allow use of these functions. Newer versions of the Hugin API are available in Java.
5. This tree is not optimal. As discussed in Section 4.2, no attempt was made to optimize junction tree construction.
6. The absence of d-connectedness (*d-separation*) implies independence. D-connectedeness implies dependence, except for pathological probability tables [24, Section 4]. The tables in the stud farm example are not pathological.

References

1. M. Valtorta, Y.-G. Kim, and J. Vomlel, "Soft evidential update for multiagent systems," *International Journal of Approximate Reasoning*, vol. 29, no. 1, pp. 71–106, 2002.
2. M. Bloemeke, *Agent Encapsulated Bayesian Networks.* Ph.D. thesis, Department of Computer Science, University of South Carolina, 1998.
3. S.L. Lauritzen and D. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *Journal of the Royal Statistical Society, Series B*, vol. 50, pp. 157–224, 1988.
4. G.R. Shafer and P.P. Shenoy, "Probability propagation," *Annals of Mathematics and Artificial Intelligence*, vol. 2, pp. 327–352, 1990.
5. F.V. Jensen, *An Introduction to Bayesian Networks*. Springer-Verlag: New York, NY, 1995.
6. S.L. Lauritzen and F.V. Jensen, "Local computation with valuations from a commutative semigroup," in *Annals of Mathematics and Artificial Intelligence*, vol. 21, pp. 51–69. J.C. Baltzer A.G., 1997.
7. Y. Xiang, "A probabilistic framework for cooperative multiagent distributed interpretation and optimization of communication," *Artificial Intelligence*, vol. 86, pp. 295–342, 1996.
8. Y. Xiang and V. Lesser, "Justifying multiply sectioned Bayesian networks," in *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, Boston, MA, July 2000, pp. 349–356.
9. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufman: San Mateo, CA, 1988.
10. J. Pearl, "Jeffrey's rule, passage of experience, and *neo*-Bayesianism," in *Knowledge Representation and Defeasible Reasoning*, edited by H.E. Kyburg, Jr., Kluwer Academic Publishers, 1990, pp. 245–265.
11. H.E. Kyburg, Jr., "Bayesian and non-Bayesian evidential updating," *Artificial Intelligence*, vol. 31, pp. 271–293, 1987.
12. R.E. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley and Sons: New York, NY, 1990.
13. A.J. Grove and J.Y. Halpern, "Probability update: Conditioning vs. cross-entropy," in *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, Providence, RI, Aug. 1997, pp. 208–214.
14. J. Vomlel, *Methods of Probabilistic Knowledge Integration*. Ph.D. thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Dec. 1999.
15. U. Kjaerulff, "Triangulation of graphs—algorithms giving small total state space," Technical Report R90-09. Technical report, Department of Computer Science, University of Aalborg, March 1990.

16. A. Becker and D. Geiger, "A sufficiently fast algorithm for finding close to optimal clique trees," *Artificial Intelligence*, vol. 125, pp. 3–17, 2001.
17. I. Csiszár, "I-divergence geometry of probability distributions and minimization problems," *Ann. Prob.*, vol. 3, no. 1, pp. 146–158, 1975.
18. S. Haberman, *The Analysis of Frequency Data*, The University of Chicago Press: Chicago and London, 1974.
19. A. Darwiche, "A differential approach to inference in Bayesian networks," in *Proceedings of the Sixteenth Annual Conference on Uncertainty in Arificial Intelligence (UAI-00)*, Stanford, CA, July 2000, pp. 123–132.
20. E. Castillo and J.M. Gutiérrez, "A new method for efficient symbolic propagation in discrete Bayesian networks," *Networks*, vol. 28, pp. 31–43, 1996.
21. F.V. Jensen, "Gradient descent training of Bayesian networks," in *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU-99)*, London, 1999.
22. R. Jiroušek, "Solution of the marginal problem and decomposable distributions," *Kybernetika*, vol. 27, no. 5, pp. 403–412, 1991.
23. Y.-G. Kim, *Time Critical Decision Making with Communicating Influence Diagrams*. Ph.D. thesis, Department of Computer Science and Engineering, University of South Carolina, 2001.
24. J. Pearl and T.S. Verma, "A theory of inferred causation," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, Morgan Kaufman: San Mateo, CA, April 1991, pp. 441–452.

Young-Gyun Kim is an assistant professor of Mathematics and Computer Science at South Carolina University.

He obtained B.S. and M.A. degrees in Nuclear Engineering from Han-Yang University in Korea, and M.A. and Ph.D. degrees in Computer Science from the University of South Carolina in 1994 and 2001 respectively.

He was involved in research funded by Office of Naval Research (ONR) and DARPA. Also he was conducted multiple commercial projects for web-based information retrieval and tracking, data mining, and distributed systems.

His research interests are Bayesian Network, decision making, time-critical decision making, reliability of models, and their use in intelligent agent and multi-agent systems.

His current research focuses on the design and implementation of intelligent agents to support users in various areas including medicine, transportation and education. His research is currently funded by the Minority Science and Engineering Improvement Program (MSEIP) of the US Department of Education, the Distance Learning Center at South Carolina State University and the Transportation Center at South Carolina State University.

Marco Valtorta is an associate professor of computer science and engineering at the University of South Carolina. He obtained a Laurea in Electrical Engineering from the Politecnico di Milano in 1980, an M.A. in Computer Science from Duke University in 1984, and a Ph.D. in Computer Science from Duke University in 1987. Between 1985 and 1988, he was a project officer for ESPRIT at the Commission of the European Communities in Brussels, where he supervised projects in the Advanced Information Processing area. As a faculty member at the University of South Carolina since 1988, he has conducted research funded by SPAWAR, ARDA, DARPA, the Office of Naval Research (ONR), the U.S. Department of Agriculture (DOA), CISE (an Italian laboratory controlled by ENEL, the state electricity company), and the South Carolina Law Enforcement Division. He is the author of over 30 refereed publications, an associate editor of the International Journal of Approximate Reasoning and a member of the editorial board of Applied Intelligence. His research interests are in the areas of normative reasoning under uncertainty (especially Bayesian networks, influence diagrams, and their use in stand-alone and multiagent systems), heuristics for problem solving, and computational complexity in artificial intelligence.

Jiří Vomlel is a research fellow at the Institute of Information Theory and Automation at the Academy of Science of the Czech Republic and in the Laboratory for Intelligent Systems at the University of Economics in Prague, Czech Republic. He obtained M.Sc. and Ph.D. degrees in Artificial Intelligence from the Czech Technical University in Prague in 1992 and 2000, respectively. Between 1999 and 2002, he was a research assistant at the Department of Computer Science at Aalborg University in Denmark. His research interests are in the area of uncertainty in artificial intelligence, especially computerized adaptive testing, decision-theoretic troubleshooting, knowledge integration, and computationally efficient methods for reasoning with uncertainty.