Performance Evaluation of Algorithms for Soft Evidential Update in Bayesian Networks: First Results

Scott Langevin and Marco Valtorta*

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

Abstract. In this paper we analyze the performance of three algorithms for soft evidential update, in which a probability distribution represented by a Bayesian network is modified to a new distribution constrained by given marginals, and closest to the original distribution according to cross entropy. The first algorithm is a new and improved version of the big clique algorithm [1] that utilizes lazy propagation [2]. The second and third algorithm [3] are wrapper methods that convert soft evidence to virtual evidence, in which the evidence for a variable consists of a likelihood ratio. Virtual evidential update is supported in existing Bayesian inference engines, such as Hugin. To evaluate the three algorithms, we implemented BRUSE (Bayesian Reasoning Using Soft Evidence), a new Bayesian inference engine, and instrumented it. The resulting statistics are presented and discussed.

Keywords: Bayesian networks, Iterative Proportional Fitting Procedure, Soft Evidence, Virtual Evidence, Cross Entropy.

1 Introduction and Motivation

The issue of how to deal with uncertain evidence in Bayesian networks appears in Pearl's foundational text [4, sections 2.2.2, 2.3.3] and has recently been the subject of methological inquiry and algorithm development (e.g., [1,5,6,7,8,9,3,10]). A result of these studies has been to clarify the distinction between soft and virtual evidence. Briefly, representing uncertain probabilistic evidence as virtual evidence is appropriate when we model the reliability of an information source, while the soft evidence representation is appropriate when we want to incorporate the distribution of a variable of interest into a probabilistic model. Update based on virtual evidence (sometimes called likelihood evidence) is supported

* This work was funded in part by the Intelligence Advanced Research Projects Activity (IARPA) Collaboration and Analyst System Effectiveness (CASE) Program, contract FA8750-06-C-0194 issued by the Air Force Research Laboratory (AFRL). The views and conclusions are those of the authors, not of the US Government or its agencies. We thank three anonymous reviewers for their comments.

in several existing Bayesian inference engines, such as Hugin[1]. This paper is concerned with soft evidence only.

For the purpose of this paper, we define evidence in a Bayesian network as a collection of findings on variables of the Bayesian network. A hard finding specifies which value (state) the variable is in. 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. See [1] for more general definitions of evidence. Some authors describe the problem of update in the presence of soft evidence as a model revision or parameter tuning problem. In the case of soft evidential update, all evidence (hard and soft) is presented simultaneously; in the case of model revision, the soft evidence is better considered as a constraint on the probability distribution encoded by the model, which is modified before evidence is applied. Despite the clear difference in the problems that are solved, similar algorithms can be used to solve both problems, as can be seen by contrasting [9], which takes the model revision approach, with [3], which takes the evidential update approach.

Belief update in the presence of hard evidence is carried out by conditioning. As observed by many authors, conditioning cannot be used to update beliefs in the presence of soft evidence. The general soft evidential update method of [1] will be used in this paper; this general method admits several detailed algorithmic variants, which have different efficiency characteristics with respect to network topologies and evidence presentations. The input to the method consists of a Bayesian network and a set of soft and hard findings. The method computes implicitly a joint probability that has two properties: (1) the evidence is respected, i.e. the findings are marginals for the joint probability distribution; (2) the joint probability is as close as possible to the initial distribution represented in the input Bayesian network, where distance is measured by cross-entropy (I-divergence). The joint probability is computed implicitly in that only its single-variable marginals are output. The focus of this paper is the experimental comparison of three such variants: the big clique algorithm of [1,6], and the two wrapper-based methods of [3]. The authors of [3] prove that the three variants compute the same distribution. The three variants are described in the following section. We aim (in future work) to provide further insight into the appropriateness of the three variants for different network topologies and evidence presentations.

2 Algorithms

In this section we first describe lazy propagation, an efficient probabilistic update algorithm that is used as the core update mechanism for the three algorithms analyzed in this paper. We then describe the big clique algorithm and the two variations of wrapper algorithms that can utilize any inference engine without modification. In this paper we only consider wrappers for lazy propagation. Finally, we list various other methods that have been proposed for soft evidential update.

[1] Virtual evidence is, confusingly, called soft evidence in [11].

2.1 Lazy Propagation

Lazy propagation [2] is an efficient junction tree algorithm that utilizes d-separation properties of the original Bayesian network by maintaining a multiplicative decomposition of clique and separator potentials. It merges the ideas of belief update algorithms that compute all marginals and direct query-based algorithms that compute a marginal for a query and better exploit evidence-induced d-separation.

Lazy propagation can be used with any computational tree structure that maintains the d-separation properties of the original Bayesian network. Our implementation uses the junction tree structure as does Hugin propagation, rather than the original Lauritzen-Spiegelhalter or the Shafer-Shenoy structures. In particular, two mailboxes per separator are used, one for messages sent during the collect evidence phase and the other for the distribute evidence phase [12]. Cliques and separators in the junction tree maintain sets of potentials and combination of potentials is delayed as long as possible to take advantage of d-separation and unity potential properties of the Bayesian network.

Lazy propagation consists of two phases: collecting evidence to the designated root clique, and distributing evidence from the designated root clique to the rest of the junction tree. Evidence is collected and distributed by message passing, where each message is a collection of potentials.

Following is a description of the lazy propagation algorithm:

1. Build a junction tree for the Bayesian network (see [13] and [2] for details).
2. Apply hard evidence (see procedure on page 287 for details).
3. Invoke *Collect Evidence* on designated root of junction tree.
4. Invoke *Distribute Evidence* on designated root of junction tree.
5. Invoke *Calculate Posterior Marginals.*

Collect Evidence. Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be adjacent cliques in the junction tree and let $\mathcal{S}$ be the separator between $\mathcal{C}_i$ and $\mathcal{C}_j$. Let $\Phi_{\mathcal{C}_i}$ and $\Phi_{\mathcal{C}_j}$ be the set of potentials associated with $\mathcal{C}_i$ and $\mathcal{C}_j$. Let $\Phi^{\uparrow}$ and $\Phi^{\downarrow}$ be the set of potentials stored in the collect and distribute mailboxes of $\mathcal{S}$ respectively. If *Collect Evidence* is invoked on $\mathcal{C}_j$ from $\mathcal{C}_i$, then:

1. $\mathcal{C}_j$ invokes *Collect Evidence* on all adjacent cliques except $\mathcal{C}_i$.
2. The message $\Phi^{\uparrow}$ from $\mathcal{C}_j$ to $\mathcal{C}_i$ is calculated (using the algorithm on page 287) and stored in the collect mailbox of $\mathcal{S}$.
3. Update $\Phi_{\mathcal{C}_i} = \Phi_{\mathcal{C}_i} \cup \Phi^{\uparrow}$.

Distribute Evidence. Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be adjacent cliques in the junction tree and let $\mathcal{S}$ be the separator between $\mathcal{C}_i$ and $\mathcal{C}_j$. Let $\Phi_{\mathcal{C}_i}$ and $\Phi_{\mathcal{C}_j}$ be the set of potentials associated with $\mathcal{C}_i$ and $\mathcal{C}_j$. Let $\Phi^{\uparrow}$ and $\Phi^{\downarrow}$ be the set of potentials stored in the collect and distribute mailboxes of $\mathcal{S}$ respectively. If *Distribute Evidence* is invoked on $\mathcal{C}_j$ from $\mathcal{C}_i$, then:

1. The message $\Phi^{\downarrow}$ from $\mathcal{C}_i$ to $\mathcal{C}_j$ is calculated (using the algorithm on page 287) and stored in the distribute mailbox of $\mathcal{S}$.
2. Update $\Phi_{\mathcal{C}_j} = \Phi_{\mathcal{C}_j} \cup \Phi^{\downarrow} \backslash \Phi^{\uparrow}$.
3. $\mathcal{C}_j$ invokes *Distribute Evidence* on all adjacent cliques except $\mathcal{C}_i$.

Calculate Message. Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be adjacent cliques in the junction tree and let $\mathcal{S}$ be the separator between $\mathcal{C}_i$ and $\mathcal{C}_j$. Let $\Phi_{\mathcal{C}_i}$ be the set of potentials associated with $\mathcal{C}_i$. Let $dom(A)$ be the set of variables associated with A (where A is either a potential or a separator). The message passed from $\mathcal{C}_i$ to $\mathcal{C}_j$ is calculated as follows:

```
1. Set $\mathcal{R}_\mathcal{S}$ = Invoke *Find Relevant Potentials* on $\Phi_{\mathcal{C}_i}$ for $dom(\mathcal{S})$.
2. For each variable $\mathcal{X}$ in $\{\mathcal{X} \in dom(\phi) | \phi \in \mathcal{R}_\mathcal{S}, \mathcal{X} \notin dom(\mathcal{S})\}$
   (a) Marginalize $\mathcal{X}$ out of $\mathcal{R}_\mathcal{S}$:
       i. Set $\Phi_\mathcal{X} = \{\phi \in \mathcal{R}_\mathcal{S} | \mathcal{X} \in dom(\phi)\}$.
       ii. Let $\phi^*_\mathcal{X} = \sum_\mathcal{X} \prod_{\phi \in \Phi_\mathcal{X}} \phi$.
       iii. Update $\mathcal{R}_\mathcal{S} = \{\phi^*_\mathcal{X}\} \cup \mathcal{R}_\mathcal{S} \backslash \Phi_\mathcal{X}$
3. Return $\mathcal{R}_\mathcal{S}$.
```

A detailed presentation on alternative ways to perform the second step can be found in [14].

Find Relevant Potentials. Let Φ be a set of potentials and let $\mathcal{S}$ be a set of variables. The relevant potentials of Φ for calculating the joint probablity of $\mathcal{S}$ are calculated as follows:

```
1. Let $\mathcal{R}_\mathcal{S} = \{\exists \mathcal{X} \in dom(\phi) | \mathcal{X}$ is d-connected to $\mathcal{Y} \in \mathcal{S}\}$.
2. Use the unity-potential axiom to remove from $\mathcal{R}_\mathcal{S}$ all potentials containing
   only barren head variables (defined in, e.g., [13]) to obtain $\mathcal{R}'_\mathcal{S}$.
3. Return $\mathcal{R}'_\mathcal{S}$.
```

Apply Hard Evidence. In the lazy propagation algorithm, hard evidence is incorporated by applying hard evidence on a variable $\mathcal{X}$ with all cliques $\mathcal{C}_i$ where $\mathcal{X} \in dom(C_i)$. This is done to fully exploit d-separation properties of the Bayesian network induced by the evidence. Hard evidence on a variable $\mathcal{X} = x$ is incorporated by the reduction of the domain of all potentials ϕ_i where $\mathcal{X} \in dom(\phi_i)$ to only include configurations of the potential where $\mathcal{X} = x$. All configurations where $\mathcal{X} \neq x$ are simply removed. This process is called an instantiation of ϕ_i.

Calculate Posterior Marginals. In the lazy propagation algorithm, marginals of all variables in the Bayesian network can be calculated by first applying any hard evidence entered, then performing the collect evidence and distribute evidence phases, known collectively as a full propagation of evidence. Let $\mathcal{P}(\mathcal{X}|\epsilon)$ be the posterior marginal of $\mathcal{X}$. Calculation of marginals is then performed on each variable $\mathcal{X}$ by the following:

```
1. For each variable $\mathcal{X}$
   (a) Let $\Phi_\mathcal{X} = \{argmin_{\Phi_{\mathcal{C}_i}} dom(\mathcal{C}_i) | \mathcal{X} \in dom(\mathcal{C}_i)\}$.
   (b) Set $\mathcal{R}_\mathcal{X}$ = Invoke *Find Relevant Potentials* on $\Phi_\mathcal{X}$ for $dom(\{\mathcal{X}\})$.
   (c) For each variable $\mathcal{Y}$ in $\{\mathcal{Y} \in dom(\phi) | \phi \in \mathcal{R}_\mathcal{X}, \mathcal{Y} \neq \mathcal{X}\}$
```

```
        i. Marginalize $\mathcal{Y}$ out of $\mathcal{R}_{\mathcal{X}}$:
           A. Set $\Phi_{\mathcal{Y}} = \{\phi \in \mathcal{R}_{\mathcal{X}} | \mathcal{X} \in dom(\phi)\}$.
           B. Let $\phi^*_{\mathcal{Y}} = \sum_{\mathcal{Y}} \prod_{\phi \in \Phi_{\mathcal{Y}}} \phi$.
           C. Update $\mathcal{R}_{\mathcal{X}} = \{\phi^*_{\mathcal{Y}}\} \cup \mathcal{R}_{\mathcal{X}} \backslash \Phi_{\mathcal{Y}}$
    (d) Calculate
```

$$\mathcal{P}(\mathcal{X}|\epsilon) = \frac{\prod_{\phi \in \mathcal{R}_{\mathcal{X}}} \phi}{\sum_{\mathcal{X}} \prod_{\phi \in \mathcal{R}_{\mathcal{X}}} \phi}$$

The above algorithm can be modified to also calculate posterior marginals using the separators as well as the cliques.

2.2 Lazy Big Clique Algorithm

The big clique algorithm [1] incorporates soft evidence by combining two methods: junction tree propagation and Iterative Proportional Fitting Procedure (IPFP; [15,16,1]). The original big clique algorithm modified the Hugin propagation algorithm and therefore did not exploit d-separation properties of the underlying Bayesian network. A new version of the big clique algorithm was developed (the lazy big clique algorithm), that is more efficient by taking advantage of d-separation using the lazy propagation algorithm described in the previous section.

The lazy big clique algorithm modifies the lazy propagation algorithm as follows:

1. Construct a junction tree that includes all variables that have soft evidence in one clique - the big clique $\mathcal{C}_1$.
2. Apply hard evidence and invoke the lazy propagation routine *Collect Evidence* on $\mathcal{C}_1$.
3. Combine all potentials associated with $\mathcal{C}_1$ to produce the joint probability distribution $\mathcal{P}(\mathcal{C}_1)$.
4. Absorb all soft evidence in $\mathcal{C}_1$ (with the algorithm described on page 289).
5. Invoke the *Big Clique Distribute Evidence* routine. A special method is needed to distribute evidence from the big clique since during absorption of soft evidence the decomposition of potentials in $\mathcal{C}_1$ is lost, and therefore a division by the evidence received from a neighboring clique is necessary when calculating messages to avoid passing back redundant information.

Big Clique Distribute Evidence

1. For each clique $\mathcal{C}_i$ adjacent to $\mathcal{C}_1$, combine potentials of message in collect mailbox of separator S between $\mathcal{C}_i$ and $\mathcal{C}_1$, call this result Φ^*_i - the evidence $\mathcal{C}_1$ received from $\mathcal{C}_i$.
2. Calculate message passed from $\mathcal{C}_1$ to $\mathcal{C}_i$ as follows:

$$\Phi^{\downarrow}_i = \frac{\Phi_{C_1}}{\Phi^*_i}$$

3. For each variable $\mathcal{X}$ in $\{\mathcal{X} \in dom(\Phi_i^{\downarrow}) | \mathcal{X} \notin \mathcal{S}\}$
   (a) Marginalize out $\mathcal{X}$.
4. Let $\Phi_i^{\downarrow *}$ be the potential obtained.
5. Store $\Phi_i^{\downarrow *}$ in the distribute mailbox of $\mathcal{S}$.
6. Update $\Phi_{\mathcal{C}_i} = \Phi_{\mathcal{C}_i} \cup \Phi_i^{\downarrow *}$.
7. $\mathcal{C}_i$ invokes the lazy propagation routine *Distribute Evidence* on all adjacent cliques except $\mathcal{C}_1$.

Absorption of Soft Evidence. We define absorption in the special big clique $\mathcal{C}_1$ as the process by which the joint probability $\mathcal{P}(\mathcal{C}_1)$ is updated to satisfy the constraints imposed by soft evidence on variables $\mathcal{S} \subseteq \mathcal{C}_1$, where $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_k\}$. Let $\mathcal{Q}(\mathcal{C}_1)$ be the joint probability after absorption.

Then $\forall i \sum_{\mathcal{C}_1 \backslash \mathcal{S}_i} \mathcal{Q}(\mathcal{C}_1) = \mathcal{P}(\mathcal{S}_i)$, where $\mathcal{P}(\mathcal{S}_i)$ is the soft evidence on $\mathcal{S}_i$, $i = 1, ..., k$. Absorption of soft evidence in clique $\mathcal{C}_1$ is done using IPFP and consists of cycles of k steps, one per finding. Each step corresponds to one soft finding. The procedure is as follows:

$$\mathcal{Q}_0(\mathcal{C}_1) = \mathcal{P}(\mathcal{C}_1)$$
$$\mathcal{Q}_i(\mathcal{C}_1) = \frac{\mathcal{Q}_{i-1}(\mathcal{C}_1) \cdot \mathcal{P}(\mathcal{S}_j)}{\mathcal{Q}_{i-1}(\mathcal{S}_j)}$$

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

2.3 Wrapper Method 1: Iterate over Network

Both wrapper methods [3] utilize any existing Bayesian inferencing engine that supports virtual evidence by converting soft evidence findings into virtual evidence that are applied to the Bayesian network using standard inference. Convergence is achieved using an iterative method. For wrapper method 1, at each iteration one soft evidence finding is converted to virtual evidence and applied. The process is performed repeated until convergence as follows:

Let $\mathcal{P}(\mathcal{X})$ be the joint probability of the Bayesian network $\mathcal{N}$ obtained using standard BN inference. Let $\mathcal{S}$ be the variables with soft evidence, where $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_k\}$, and $\mathcal{P}(\mathcal{S}_i)$ is the soft evidence on $\mathcal{S}_i, i = 1, .., k$. This algorithm applies soft evidence by iterating over the whole network as follows:

1. $\mathcal{Q}_0 = \mathcal{P}(\mathcal{X});\ k = 1;$
2. Repeat the following until convergence:
   (a) $i = 1 + (k - 1) \bmod m;\ j = 1 + \lfloor (k - 1)/m \rfloor;$
   (b) (Convert the soft evidence to virtual evidence) Construct virtual evidence $\mathcal{V}_{i,j}$ with likelihood ratio:
   $$\mathcal{L}(\mathcal{S}_i) = \frac{\mathcal{P}(\mathcal{S}_i)}{\mathcal{Q}_{k-1}(\mathcal{S}_i)}$$
   (c) Obtain $\mathcal{Q}_k(\mathcal{X})$ by updating $\mathcal{Q}_{k-1}(\mathcal{X})$ with $\mathcal{V}_{i,j}$ using standard BN inference.
   (d) $k = k + 1$

2.4 Wrapper Method 2: Iterate over Soft Evidence

Wrapper method 2 is similar to the big clique algorithm in that both methods calculate the joint probability of the soft evidence variables and use IPFP to absorb soft evidence. The big clique performs IPFP on all variables in the big clique, while the wrapper 2 method only performs IPFP on the soft evidence variables. The wrapper 2 method converts the soft evidence to virtual evidence that is applied to the Bayesian network using standard inference. As a result, the wrapper 2 method requires two full propagations: one to calculate the joint probability of the soft evidence variables, and another to calculate the posterior marginals. The process is as follows:

Let $\mathcal{P}(\mathcal{X})$ be the joint probability of the Bayesian network $\mathcal{N}$ obtained using standard BN inference. Let $\mathcal{S}$ be the variables with soft evidence, where $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_k\}$ and $\mathcal{P}(\mathcal{S}_i)$ is the soft evidence on $\mathcal{S}_i, i = 1, .., k$. Let $\mathcal{P}(\mathcal{S})$ be the joint probability of $\mathcal{S}$. This algorithm applies soft evidence as follows:

1. Use any BN inference method on $\mathcal{N}$ to obtain $\mathcal{P}(\mathcal{S})$.
2. Absorb all soft evidence in $\mathcal{P}(\mathcal{S})$ (with the algorithm described below) to obtain $\mathcal{Q}(\mathcal{S})$.
3. (Convert the soft evidence to virtual evidence) Construct virtual evidence $\mathcal{V}$ with likelihood ratio:
$$\mathcal{L}(\mathcal{S}) = \frac{\mathcal{Q}(\mathcal{S})}{\mathcal{P}(\mathcal{S})}$$
4. Update the beliefs in $\mathcal{N}$ with $\mathcal{V}$ using standard BN inference.

Bayesian network engines of the "all-marginal" variety (junction tree based) do not compute joint probabilities, but rather calculate single-variable marginals for all variables. Junction tree algorithms can be modified to calculate joint probabilities for a set of variables by adding pairwise edges between all variables of interest to the moral graph before performing triangulation. This ensures the resulting junction tree will contain a clique that contains all variables of interest. After propagation, the joint probability of the variables can be constructed by combining all potentials associated with this clique. Our implementation of the wrapper 2 method uses this technique to calculate the joint probability of the soft evidence variables. See [17] and [13, Section 5.2] for a discussion of other methods to calculate joint probabilities in "all-marginal" algorithms.

Absorption of Soft Evidence. We define absorption of soft evidence as the process by which the joint probability $\mathcal{P}(\mathcal{S})$ is updated to satisfy the constraints imposed by soft evidence on variables $\mathcal{S}$, where $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_k\}$. Let $\mathcal{Q}(\mathcal{S})$ be the joint probability after absorption. Then $\forall i \sum_{\mathcal{S}\setminus\mathcal{S}_i} \mathcal{Q}(\mathcal{S}) = \mathcal{P}(\mathcal{S}_i)$, where $\mathcal{P}(\mathcal{S}_i)$ is the soft evidence on $\mathcal{S}_i$, $i = 1, ..., k$. Absorption of soft evidence is done 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 procedure is as follows:

$$\mathcal{Q}_0(\mathcal{S}) = \mathcal{P}(\mathcal{S})$$

$$\mathcal{Q}_i(\mathcal{S}) = \frac{\mathcal{Q}_{i-1}(\mathcal{S}) \cdot \mathcal{P}(\mathcal{S}_j)}{\mathcal{Q}_{i-1}(\mathcal{S}_j)}$$

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

2.5 Other Methods

Here, we only list several other methods for soft evidence update: the space-saving implementation of IPFP [18,19,20]; the soft updating algorithm of [10]; and the approximate update algorithms by Peng and Ding [9].

3 Experimental Setup

To evaluate the lazy big clique (referred to as big clique from here on) and wrapper algorithms, a new Bayesian reasoning engine was constructed that utilizes lazy propagation, the Bayesian Reasoning Using Soft Evidence (BRUSE) engine. BRUSE was developed using the Java framework and implements the three discussed algorithms for soft evidential update. In order to evaluate algorithm performance, an instrumentation framework was implemented into BRUSE to gather statistics during inferencing. Statistics collected are: number of table multiplication operations performed, number of table addition operations performed, number of table division operations performed, IPFP iterations required for convergence, domain size of the IPFP table, and time to perform inference. Our testing was done on a Dell Optiplex Intel Core 2 Duo, 2.4 GHz machine with 2GB of RAM. Each test configuration was performed ten times and average statistics were calculated. The tests were performed using four Bayesian networks of varying sizes and complexity. Two of the networks were downloaded from a web-based repository [21]: stud farm (12 nodes) [13] and alarm (37 nodes) [22]. The other two networks, test71 (80 nodes) and test61 (200 nodes), were randomly generated to simulate complex networks. Table 1 shows statistics for the four networks. These statistics show the relative complexity of the networks and corresponding junction trees when one soft evidence finding is chosen.

Each network was tested with ten different test configurations consisting of one to ten soft evidence findings. Hard evidence was not used in our tests. Each test, randomly selects soft evidence variables accordingly to satisfy the test configuration chosen. The same set of soft evidence findings are applied to each of the three algorithms to compare their relative performance.

Table 1. Statistics for test networks

Network	*Number of Nodes*	*Number of Cliques*	*Max Clique Size*	*Triangulation Weight*
studfarm	12	9	16	116
alarm	37	27	144	1065
test71	80	65	2916	13793
test61	200	175	262144	347180

4 Results

We present here some of the experimental results obtained so far, without interpretation. Since we use the min-size heuristic, which is widely recognized as excellent [23], the cost to generate the junction tree is negligible. Accordingly, for all networks, we collect statistics only after the construction of the junction tree. Also, we found that, in our implementation, inference time corresponds closely to the number of elementary table operations performed, where we define number of elementary table operations as the sum of table multiplications, additions and divisions. As an example, compare Figure 1 with Figure 2. Therefore, the number of table operations provides a good measure of relative performance.

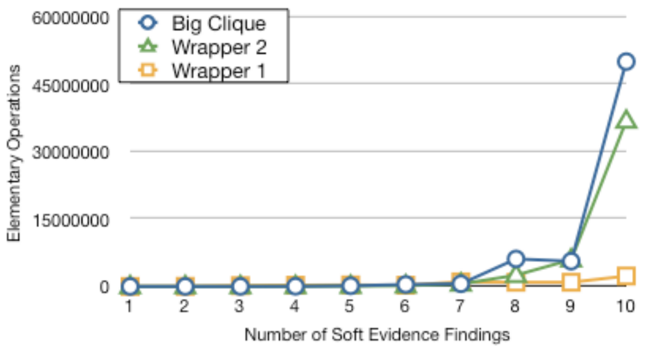

Fig. 1. Average number of elementary table operations for the alarm network

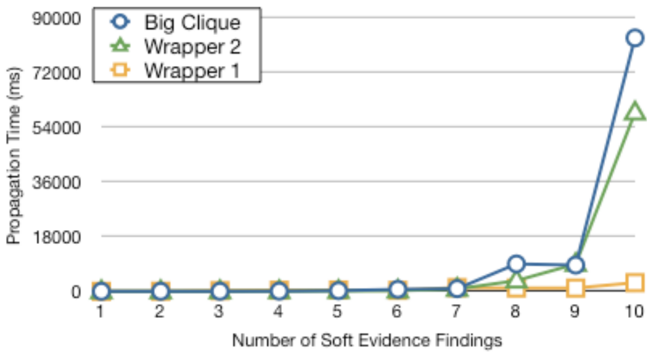

Fig. 2. Average propagation time for the alarm network

For all networks, it appears that wrapper 1 is slower than the other two methods when the number of soft evidence findings is small (less than 7 for the networks we consider). (We apologize to the reader for the fact that several of the graphs do not provide sufficient resolution to show this.) We conjecture that

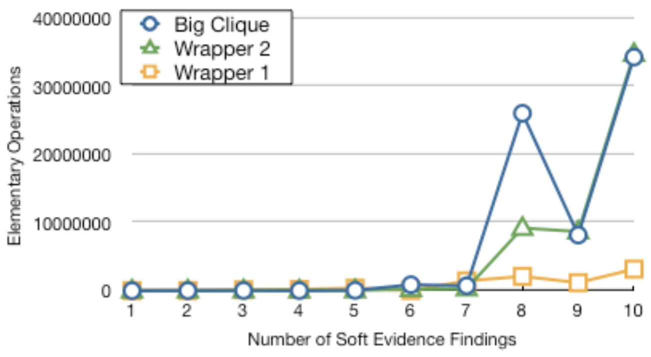

Fig. 3. Number of elementary table operations for test case 3 of the alarm network

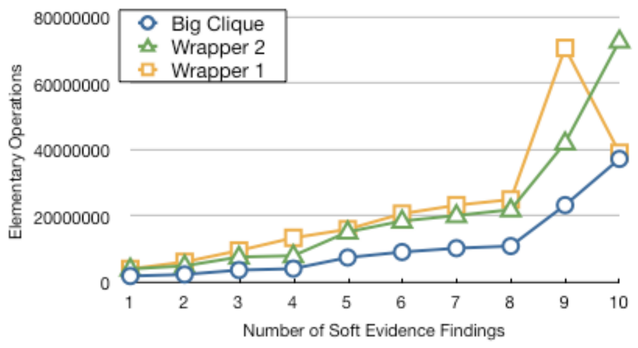

Fig. 4. Average number of elementary table operations for the test61 network

the reason is that the cost of propagation through the whole network dominates the cost of IPFP over a rather small joint probability. As a consequence, we also conjecture that this would not be the case for networks with large state spaces, for which the joint probability tables are large, even when they contain only a few nodes.

For all networks, wrapper 2 and big clique have similar run times. This is to be expected, because both methods need to compute the joint probability of the soft evidence variables, which requires, in a junction tree algorithm, the computation of the joint probability of variables in a clique that contains all the soft evidence variables. The big clique algorithm also performs IPFP on all variables in that clique, while the wrapper 2 method only performs IPFP on the soft evidence variables. On the other hand, the wrapper 2 method uses virtual evidence, which requires two full propagations, to compute posterior marginals, while the big clique method only needs one full propagation. When the cost of propagation is higher than the cost of IPFP on the big clique, the big clique algorithm will perform better, and vice versa.

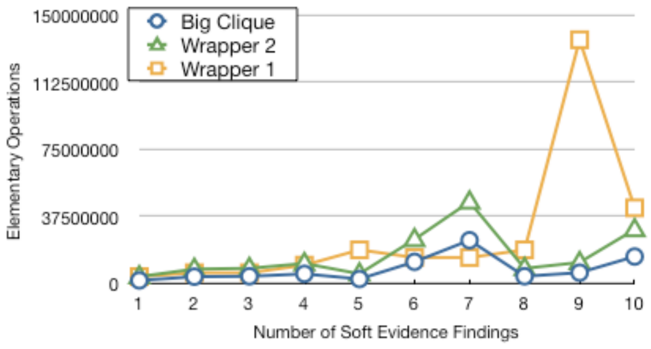

Fig. 5. Number of elementary table operations for test 6 of the test61 network

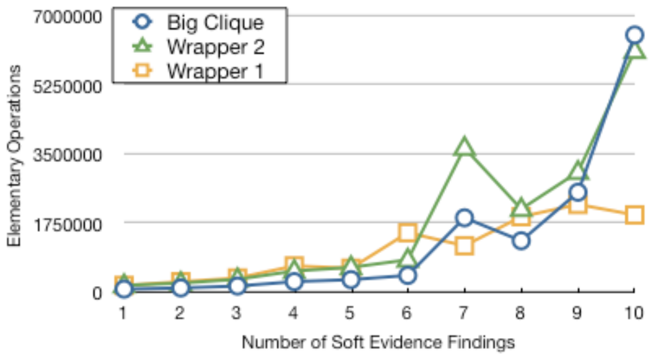

Fig. 6. Average number of elementary table operations for the test71 network

For the stud farm network (Figure 7), the cost of propagation in the very small junction tree is dominated by the cost of IPFP in the big clique and wrapper 2 algorithms. Use of IPFP requires the computation of the joint probability of the soft evidence variable(s), by first computing the joint probability of the variables in the cliques containing the soft evidence variable(s) and then marginalizing down to the soft evidence variable(s). On the other hand, the wrapper 1 method computes posterior marginal probabilities by updating with respect to each individual soft evidence variable in turn. This computation is very fast on the small junction tree of the stud farm network. Accordingly, the wrapper 1 method is the fastest for this network.

For the alarm network (Figure 1), the results are similar to those for stud farm. The relatively poor performance of big clique for eight soft evidence findings is explained by a particularly difficult evidence scenario, whose performance is reported in Figure 3. The resulting big clique for these soft evidence findings is very large resulting in an expensive IPFP computation. A similar situation occurs for ten soft evidence findings.

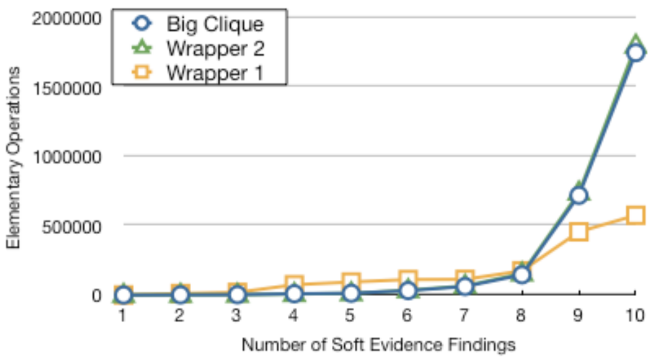

Fig. 7. Average number of elementary table operations for the stud farm network

For the test61 network (Figure 4), the state spaces of one of the cliques in the junction tree is very large, reflecting the fact that this is indeed a random network and not a typical, human-constructed, low-treewidth network [24]. The performance of the wrapper 1 algorithm is accordingly poor, because the cost of additional propagations required by this method overcomes the savings resulting from not performing IPFP on a joint distribution. Similarly, wrapper 2 is slower than the big clique algorithm, because it performs twice the number of propagations. The spike in the number of table operations for the wrapper 1 method with nine soft evidence findings is due mainly to one difficult evidence scenario, whose performance is reported in Figure 5, for which the number of IPFP iterations before convergence is very high.

For the test71 network (Figure 6), the number of operations for the wrapper 2 method is approximately double the number for big clique. This indicates that the contribution of IPFP is negligible, while the propagation cost for probability update after IPFP dominates the number of operations. Since wrapper 2 needs to perform two such propagations, as opposed to one for the big clique algorithm, the experimental result is explained. The junction tree constructed for the wrapper one method, which does not need to include all soft evidence variables in one clique, is much simpler than the one built for the other two methods, and this explains the comparatively better performance of wrapper one for seven evidence findings.

5 Conclusion

This paper only presents initial results. As discussed in the previous section, our initial tests indicate that the three algorithms for soft evidential update we have implemented have definite relative strengths and weaknesses. However, future work, such as improving the instrumentation of the implementation to collect better convergence data, designing experiments to test specific features of networks and evidence configurations that may include hard findings, and testing on a wider range of large networks, remains to be done in order to conclude under which conditions each algorithm is preferable.

Additional future work includes the analysis of several other proposed algorithms: the space-saving implementation of IPFP [18] and [19]; the soft updating algorithm of [10]; the approximate update algorithms by Peng and Ding [9]; and possibly more.

It will also be necessary to evaluate memory usage, which leads to a consideration of any-space algorithms such as recursive conditioning [13] instead of junction tree algorithms, and to evaluate the effect of performance tuning, such as the use of different query methods to calculate lazy messages and of different methods (e.g., variable passing [13] and query-based methods [17]) to calculate joint probabilities in BRUSE.

References

1. Valtorta, M., Kim, Y.G., Vomlel, J.: Soft evidential update for probabilistic multiagent systems. International Journal of Approximate Reasoning 29(1), 71–106 (2002)
2. Madsen, A.L., Jensen, F.V.: Lazy propagation: A junction tree inference algorithm based on lazy evaluation. Artificial Intelligence 113, 203–245 (1999)
3. Pan, R., Peng, Y., Ding, Z.: Belief update in Bayesian networks using uncertain evidence. In: ICTAI, pp. 441–444. IEEE Computer Society Press, Los Alamitos (2006)
4. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, San Mateo (1988)
5. Chan, H., Darwiche, A.: On the revision of probabilistic beliefs using uncertain evidence. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, pp. 99–105 (2003)
6. Kim, Y.-G., Valtorta, M., Vomlel, J.: A prototypical system for soft evidential update. Applied Intelligence 21(1) (2004)
7. Vomlel, J.: Probabilistic reasoning with uncertain evidence. Neural Network World, International Journal on Neural and Mass-Parallel Computing and Information Systems 14(5), 453–456 (2004)
8. Chan, H., Darwiche, A.: On the revision of probabilistic beliefs using uncertain evidence. Artificial Intelligence 163, 67–90 (2005)
9. Peng, Y., Ding, Z.: Modifying Bayesian networks by probability constraints. In: Proceedings of the Twenty-first Annual Conference on Uncertainty in Artificial Intelligence (UAI 2005), Edinburgh, Scotland, July 2005, pp. 459–466 (2005)
10. Di Tomaso, E., Baldwin, J.: An approach to hybrid probabilistic models. International Journal of Approximate Reasoning 47, 202–218 (2008)
11. Madsen, A.L., Jensen, F., Kjaerulff, U.B., Lang, M.: The Hugin tool for probabilistic graphical models. Internationl Journal on Artificial Intelligence Tools 14, 507–543 (2005)
12. Lepar, V., Shenoy, P.P.: A comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer architectures for computing marginals of probability distributions. In: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1998), Madison, WI, July 1998, pp. 328–337 (1998)
13. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer, New York (2007)

14. Madsen, A.: Variations over the message computation algorithm of lazy propagation. IEEE Transactions on Systems, Man, and Cybernetics Part B 36, 636–648 (2006)
15. Csiszár, I.: *I*-divergence geometry of probability distributions and minimization problems. Ann. Prob. 3(1), 146–158 (1975)
16. Vomlel, J.: Methods of Probabilistic Knowledge Integration. PhD thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University (December 1999)
17. Bloemeke, M., Valtorta, M.: A hybrid algorithm to compute marginal and joint beliefs in Bayesian networks and its complexity. In: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1998), Madison, WI, July 1998, pp. 208–214 (1998)
18. Jiroušek, R.: Solution of the marginal problem and decomposable distributions. Kybernetika 27(5), 403–412 (1991)
19. Hajek, P., Havranek, T., Jirousek, R.: Uncertain Information Processing in Expert Systems. NRC Press, Boca Raton (1992)
20. Jiroušek, R., Přeučil, S.: On the effective implementation of the iterative proportional fitting procedure. Computational Statistics and Data Analysis 19, 177–189 (1995)
21. Elidan, G.: Bayesian network repository (2001) (accessed May 22, 2008), `http://www.cs.huji.ac.il/labs/compbio/Repository/`
22. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, London, pp. 247–256 (1989)
23. Kjaerulff, U.: Triangulation of graphs—algorithms giving small total state space, Technical Report R90-09. Technical report, Department of Computer Science, University of Aalborg (March 1990)
24. Boedlander, H.L.: Discovering treewidth. In: Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O. (eds.) SOFSEM 2005. LNCS, vol. 3381, pp. 1–16. Springer, Heidelberg (2005)