New Methods for Marginalization in Lazy Propagation

Anders L Madsen
HUGIN EXPERT A/S, Gasværksvej 5, DK-9000 Aalborg, Denmark
Anders.L.Madsen@hugin.com

Abstract

Even though existing algorithms for belief update in Bayesian networks (BNs) have exponential time and space complexity, belief update in many real-world BNs is feasible. However, in some cases the efficiency of belief update may be insufficient. In such cases minor improvements in efficiency may be important or even necessary to make a task tractable. This paper introduces two improvements to the message computation in Lazy Propagation (LP). We introduce one-step lookahead methods for sorting the operations involved in a variable elimination using Arc-Reversal (AR) and extend LP with the any-space property. The performance impacts of the methods are assessed empirically.

1 INTRODUCTION

There are two main reasons for the popularity of BNs (Pearl, 1988), (Cowell et al., 1999), and (Kjærulff and Madsen, 2008) as a formalism for modeling and reasoning with uncertainty: 1) a BN is an efficient and intuitive graphical representation of a joint probability distribution and 2) there exists tools implementing efficient algorithms for belief update.

As both exact and approximate belief update in general are *NP-hard* (Cooper, 1990b; Dagum and Luby, 1993), the use of exponential complexity algorithms is justified (unless P=NP). Even though existing algorithms for belief update have exponential time and space complexity, belief update on a large number of real-world BNs is feasible. However, in some cases the efficiency of belief update may be insufficient, but close to sufficient. In such cases minor improvements in efficiency may be important or even necessary to make a task tractable. Examples of such cases include analysis at the portfolio level in financial institutions where a belief update is performed for each customer. If the portfolio consists of 100000s of customers, then the time cost of belief update becomes an important issue and even a minor improvement in efficiency can have a large impact on the performance of the portfolio level analysis. In addition, the importance of belief update performance increases as the complexity of real-world BNs increases.

Most algorithms for exact belief update belongs to either the class of *query-based* or the class of *all-marginals* algorithms. The first class contains, for instance, Belief Propagation (Pearl, 1988), Arc-Reversal (AR) (Olmsted, 1983; Shachter, 1986), Symbolic Probabilistic Inference (SPI) (Shachter et al., 1990), Recursive Decomposition (RD) (Cooper, 1990a), Variable Elimination (VE) (Cannings et al., 1978; Zhang and Poole, 1996), Bucket Elimination (Dechter, 1996a), the Fusion operator (Shenoy, 1997), Query DAGs (Darwiche and Provan, 1997), Recursive Conditioning (RC) (Darwiche, 2000) and Value Elimination (VU) (Bacchus et al., 2003) while the latter class contains, for instance, Lauritzen-Spiegelhalter (Lauritzen and Spiegelhalter, 1988), HUGIN (Jensen et al., 1990), and Shenoy-Shafer (Shafer and Shenoy, 1990).

LP (Madsen and Jensen, 1999) combines query-based and all-marginals algorithms. Message passing is performed in a junction tree where clique and separator potentials are decomposed into sets of factors and messages are computed using a (revised) query-based algorithm in an attempt to exploit independence relations induced by evidence and barren variables. Recently, Madsen (2006) introduced LP algorithms where either AR, VE or SPI is used for message and marginal computation in a variable elimination based approach.

This paper introduces two improvements to message and marginal computation in LP: one-step look-ahead methods for sorting the AR operations involved in a variable elimination and a method to extend LP with the any-space property. We also report on the results of an empirical performance analysis.

2 PRELIMINARIES

A BN $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ over variables $\mathcal{X}$ consists of an acyclic, directed graph (DAG) $\mathcal{G} = (\mathcal{X}, E)$ and a set of conditional probability distributions (CPDs) $\mathcal{P}$. It induces a joint probability distribution over $\mathcal{X}$ s.t. $P(\mathcal{X}) = \prod_{X \in \mathcal{X}} P(X | \text{pa}(X))$.

We consider *belief update* as the task of changing beliefs due to changes in the world manifested through observations. It is the task of computing the posterior marginal $P(X \mid \epsilon)$ for each $X \in \mathcal{X}$. Evidence $\epsilon = \{\epsilon_1, \ldots, \epsilon_n\}$ consists of a set of variable instantiations. We let ϵ_X denote the instantiation of X, i.e., $\epsilon_X = \{X = x\}$ and $\epsilon_X \in \epsilon$, and let $\mathcal{X}_\epsilon$ denote the set of variables instantiated by evidence ϵ.

Definition 2.1 [Barren Variable]
A variable X is a *barren* w.r.t. a set $T \subseteq \mathcal{X}$, evidence ϵ, and DAG $\mathcal{G}$, if $X \not\in T$, $X \not\in \mathcal{X}_\epsilon$ and X only has barren descendants in $\mathcal{G}$ (if any).

A probability *potential* (Shafer and Shenoy, 1990) is a non-negative and not-all-zero function over a set of variables while a probability distribution is a potential that sums to one. For probability potential ϕ with *domain* $\text{dom}(\phi) = \{X_1, \ldots, X_n\}$, we let $\text{H}(\phi)$ denote the *head* (i.e., the conditioned variables) and $\text{T}(\phi)$ denote the *tail* (i.e., the conditioning variables) of ϕ.

The *domain graph* $\mathcal{G}(\Phi) = (\mathcal{X}, E)$ of a set of probability potentials Φ over variables $\mathcal{X}$ is the graph spanned by $\mathcal{X}$ where for each ϕ an undirected edge is added between each pair of variables $X, Y \in \text{H}(\phi)$ and a directed edge is added from each $X \in \text{T}(\phi)$ to each $Y \in \text{H}(\phi)$. We let $\text{dom}(\Phi)$ denote the set of domain variables of potentials in Φ. The notion of barren variables can be extended to graphs with both directed and undirected edges (Madsen, 2006).

Definition 2.2 [Query]
A *query* on a set of probability potentials Φ is a triple $Q = (T, \Phi, \epsilon)$ where $T \subseteq \mathcal{X}$ is the target.

The set $E = \text{dom}(\Phi) \setminus T$ is referred to as the *elimination set*. The set of potentials Φ^* obtained by eliminating $\text{dom}(\Phi) \setminus T$ from Φ s.t. $\prod_{\phi \in \Phi^*} \phi = P(T, \epsilon_1 \mid \epsilon_2)$ where $\epsilon = \epsilon_1 \cup \epsilon_2$ is a *solution* to query Q. Notice that a query may have multiple solutions as a solution is a decomposition of the joint potential over target T. We define $\Phi_X \subseteq \Phi$ as $\Phi_X = \{\phi \in \Phi : X \in \text{dom}(\phi)\}$.

2.1 SOLVING QUERIES

Query-based belief update algorithms solve a single query $Q = (T, \Phi, \epsilon)$. In the following we assume that Y is to be eliminated in the process of solving Q. AR performs a sequence ρ of arc-reversal operations to make Y barren prior to removing its potential from Φ. Let X be a variable with parent set $\text{pa}(X) = \{Y, X_1, \ldots, X_n\}$ and let $\text{pa}(Y) = \{X_1, \ldots, X_n\}$. An AR operation on arc (Y, X) is performed as follows:

$$P(X|X_1, \ldots, X_n) = \sum_Y P(X|Y, X_1, \ldots, X_n) P(Y|X_1, \ldots, X_n), \quad (1)$$

$$P(Y|X, X_1, \ldots, X_n) = \frac{P(X|Y, X_1, \ldots, X_n) P(Y|X_1, \ldots, X_n)}{P(X|X_1, \ldots, X_n)}. \quad (2)$$

The AR-operation corresponds to arc-reversal in $\mathcal{G}(\Phi)$. It is necessary to avoid making cycles in the process of reversing arcs to eliminate Y. If $\text{pa}(X) \setminus \{Y\} \neq \text{pa}(Y)$, then we perform straightforward domain extensions.

Using VE Y is eliminated from Φ by marginalization of Y over the combination of potentials of Φ_Y and setting $\Phi^* = \Phi \setminus \Phi_Y \cup \{\phi_Y\}$ where $\phi_Y = \sum_Y \prod_{\phi \in \Phi_Y} \phi$.

There is a rich literature on any-space algorithms, e.g., (Dechter, 1996b; Darwiche, 2000; Bacchus et al., 2003). We consider VU and RC. RC is an any-space algorithm for exact query-based belief update based on recursive conditioning (Darwiche, 2000). RC is an instantiation of the family of algorithms referred to as VU (Bacchus et al., 2003). A main difference is that VU supports a dynamic conditioning order whereas the order is fixed in RC. RD (Cooper, 1990a), on the other hand, is a divide-and-conquer method that recursively decomposes the network and maps the resulting decomposition into a corresponding equation.

2.2 ALL-MARGINALS

All-marginals-based belief update algorithms solve a single query $Q = (\{X\}, \Phi, \epsilon)$ for each $X \in \mathcal{X}$. The *all-marginals* problem is usually solved by local procedures operating on a secondary computational structure known as the *junction tree* (also known as a join tree and a Markov tree) representation of the BN (Jensen and Jensen, 1994). Let $\mathcal{T}$ denote a junction tree with cliques $\mathcal{C}$ and separators $\mathcal{S}$. The cliques $\mathcal{C}$ are the nodes of $\mathcal{T}$ whereas the separators $\mathcal{S}$ annotate the links of $\mathcal{T}$. Each clique $C \in \mathcal{C}$ represents a maximal complete subgraph in an undirected graph[1] $\mathcal{G}^T$. The link between two neighboring cliques A and B is annotated with the intersection $S = A \cap B$, where $S \in \mathcal{S}$.

Once $\mathcal{T}$ is constructed the CPD of each $X \in \mathcal{X}$ is associated with a clique C s.t. $\text{fa}(X) \subseteq C$ where $\text{fa}(X) = \{X\} \cup \text{pa}(X)$. We let Φ_C denote the set of CPDs associated with $C \in \mathcal{C}$. Belief update proceeds as a two phase process where information is passed as messages between cliques over separators in two steps. Two messages are passed over each $S \in \mathcal{S}$; one message in each direction. Once the message passing process has completed, the marginal of each $X \in \mathcal{X}$ is computed from any node in $\mathcal{T}$ including X.

Algorithms such as HUGIN, Shenoy-Shafer, Lauritzen-Spiegelhalter, and LP differ w.r.t. the representation of clique and separator potentials and the computation of messages.

3 LAZY PROPAGATION

Message passing proceeds according to the Shenoy-Shafer scheme: A clique A sends a message $\Phi_{A \to B}$ to its neighbor B when it has received messages from all neighbors (denoted $\text{ne}(A)$) except B, see Figure 1. A message $\Phi_{A \to B}$ is the solution to a query $Q = (B, \Phi_A \cup \bigcup_{C \in \text{ne}(A) \setminus B} \Phi_{C \to A}, \epsilon)$ and it is computed as:

$$\Phi_{A \to B} = \Big(\Phi_A \cup \bigcup_{C \in \text{ne}(A) \setminus \{B\}} \Phi_{C \to A}\Big)^{M \downarrow B},$$

where M is the marginalization algorithm, i.e., either AR, VE or SPI. Prior to applying M to solve Q, potentials for which all head variables are barren and potentials over variables which are all separated from B given ϵ in $\mathcal{G}(\Phi_A \cup \bigcup_{C \in \text{ne}(A) \setminus B} \Phi_{C \to A})$ are removed. Notice that $\Phi_{A \to B}$ and Φ_C are sets of potentials. The content of $\Phi_{A \to B}$ depends on M.

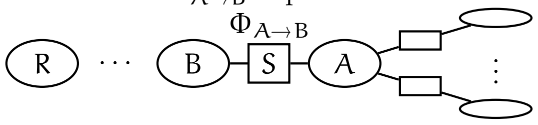

Figure 1: $\Phi_{A \to B}$ is passed from A to B.

The decomposition of potentials and the lazy elimination of variables enable an efficient exploitation of independence relations and barren variables during belief update. LP uses the structure of $\mathcal{T}$ to define a partial order on the elimination of variables in the computation of $P(X \mid \epsilon)$ for each $X \in \mathcal{X}$. While the domain of $\Phi_{A \to B}$ is defined by the elimination set $E = A \setminus B$, the computation of $\Phi_{A \to B}$ can be performed using a variety of algorithms, as described by Madsen (2006). Evidence is entered in $\mathcal{T}$ by instantiating $\mathcal{X}_\epsilon$ according to ϵ.

4 IMPROVING BELIEF UPDATE

4.1 ARC-REVERSAL SORT

Using AR a variable Y is eliminated by a sequence ρ of arc-reversal operations followed by a barren variable elimination. If $|\text{ch}(Y)| > 1$, then an arc-reversal order $\rho = ((Y, X_1), \ldots, (Y, X_{|\text{ch}(Y)|}))$ has to be determined.

Consider the query $Q = (T = \{X_1, X_3, X_4, X_5\}, \Phi, \emptyset)$ where $\Phi = \{P(X_1), (P(X_2 \mid X_1), P(X_3 \mid X_2, X_5), P(X_4 \mid X_2), P(X_5)\}$. Eliminating X_2 using AR involves reversing arcs (X_2, X_3) and (X_2, X_4). Figures 2 and 3 show the calculations for the two possible orders $\rho_{\min} = ((X_2, X_4), (X_2, X_5))$ and

[1] $\mathcal{G}^T$ is constructed from the moral graph $\mathcal{G}^m$ of $\mathcal{G}$ by adding undirected edges until the graph is triangulated. A graph is triangulated if every cycle of length greater than three has a chord.

$\rho_{\max} = ((X_2, X_5), (X_2, X_4))$, respectively. The inner circles represent the first arc-reversal operation while the outer circles represent the second arc-reversal operation. Even though the structures of the two graphs are identical, the solutions are different.

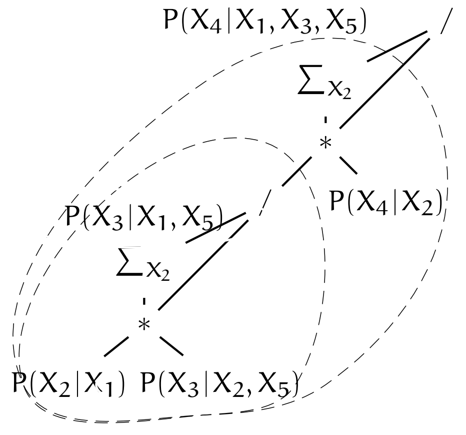

Figure 2: *maximum fill-in-weight.*

The solution illustrated in Figure 2 is $\Phi^{\text{AR}\max\downarrow T} = \{P(X_1), P(X_3 \mid X_1, X_5), P(X_4 \mid X_1, X_3, X_5), P(X_5)\}$ while the solution illustrated in Figure 3 is $\Phi^{\text{AR}\min\downarrow T} = \{P(X_1), P(X_3 \mid X_1, X_4, X_5), P(X_4 \mid X_1), P(X_5)\}$. Notice that the (unique) solution to Q obtained using VE and the algorithm of Section IV in (Madsen, 2006) is $\Phi^{\text{VE}\downarrow T} = \{P(X_1), P(X_3, X_4|X_1, X_5), P(X_5)\}$.

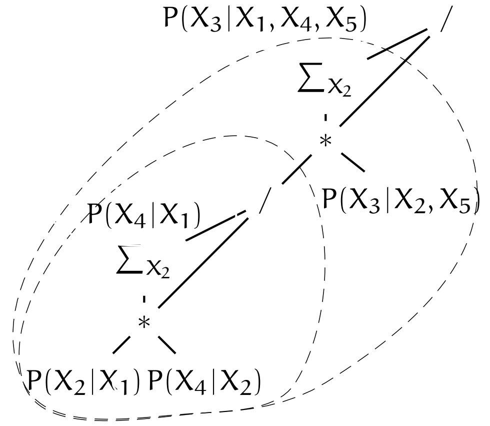

Figure 3: *minimum fill-in-weight.*

The elimination of Y by a sequence of AR operations $\rho = ((Y, X_1), \dots, (Y, X_{|\operatorname{ch}(Y)|}))$ will induce a set of new edges. The cost[2] of an AR sequence ρ is defined as the sum of the weights of the new edges induced by ρ.

The objective of considering different AR sequences is to minimize the total cost of new edges introduced by eliminating Y. It is infeasible to consider all possible sequences as the upper limit on the number of possible sequences is $n!$ where $n = |\operatorname{ch}(Y)|$. Some of the sequences may be illegal due to the graph acyclicity constraint though. The large number of possible sequences implies that the use of heuristics for determining the sequence to use is justified. We define the cost of reversing edge (Y, X) as:

$$\begin{aligned} s(Y, X) &= \sum_{Z_X \in \operatorname{pa}(X), Z_X \notin \operatorname{pa}(Y), Z_X \neq Y} \|Z_X\| \cdot \|Y\| \\ &+ \sum_{Z_Y \in \operatorname{pa}(Y), Z_Y \notin \operatorname{pa}(X)} \|Z_Y\| \cdot \|X\|. \end{aligned}$$

The cost is equal to the sum of the weights of the edges induced by new parents of X and Y.

We introduce two heuristic rules based on the score $s(Y, X)$: a *minimum fill-in-weight* rule for selecting the next edge to reverse when eliminating Y. We refer to AR in combination with *minimum fill-in-weight* as AR min. The rule where $s(Y, X)$ is maximized is referred to as *maximum fill-in-weight* and AR max denotes AR in combination with this rule.

Both *maximum fill-in-weight* and *minimum fill-in-weight* use a one step look-ahead. This implies that they do not always find the optimal order (according to the cost function). Finding an optimal order is similar to finding an optimal triangulation. It is well-known that this problem is $\mathcal{NP}$-complete, see e.g. (Yannakakis, 1981) or (Arnborg et al., 1987).

4.2 ANY-SPACE

Inspired by the work on RD and RC we extend LP with the any-space property. The basic idea is to avoid computing a representation over all values of ϕ, if $\|\operatorname{dom}(\phi)\| > \delta$ where δ is a threshold value on the size of potentials. Instead of

[2]Alternative scores may be considered. In this work, we use a score similar to the *fill-in-weight* score often used for identifying triangulations using one-step lookahead node elimination, as this rule has shown a high performance when applied to triangulation, see (Kjærulff, 1993) for more details.

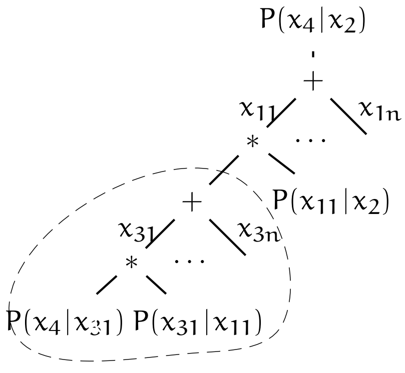

Figure 4: VE calculation of $P(X_4|X_2)$.

maintaining a large (table) representation of ϕ, values are recomputed as needed in subsequent operations. During belief update potential sizes may increase due to multiplications and decrease due to marginalizations. Let ϕ_1 and ϕ_2 be two potentials. If $\text{dom}(\phi_1) \setminus \text{dom}(\phi_2) \neq \emptyset$ or $\text{dom}(\phi_2) \setminus \text{dom}(\phi_1) \neq \emptyset$, then $\|\text{dom}(\phi_1 \cdot \phi_2)\| > \|\text{dom}(\phi_i)\|$ for $i = 1, 2$. This simple insight drives the proposed scheme. The calculation of a product $\prod \phi$ or a marginal $\phi^{\downarrow T}$ is *delayed* if $\|\text{dom}(\prod \phi)\| > \delta$ or $\|\text{dom}(\phi^{\downarrow T})\| > \delta$, respectively. Only marginalization can enforce the construction of a potential whereas both a marginalization and a product may involve delayed potentials producing a recursive scheme. Figure 4 illustrates the approach on:

$$\begin{aligned} P(X_4|X_2) &= \Phi^{\text{VE}\downarrow\{X_2,X_4\}} \\ &= \sum_{X_1} P(X_1|X_2) \sum_{X_3} P(X_3|X_1) P(X_4|X_3), \end{aligned}$$

where $\Phi = \{P(X_1 \mid X_2), P(X_3 \mid X_1), P(X_4 \mid X_3)\}$. Each entry of $P(X_4|X_2)$ is computed by accessing and combining the values of its source potentials Φ recursively. The equation becomes a formula for accessing the values of $P(X_4|X_2)$ by recursive computation. Each time an entry is accessed, it is computed. No entries are computed when the formula is constructed. This implies that the calculation of an entry is delayed until the entry is accessed as part of the calculation of another potential.

Even though $\|\text{dom}(\phi)\| > \|\text{dom}(\phi^{\downarrow T})\|$, it may be that $\|\text{dom}(\phi^{\downarrow T})\| > \delta$. In this case, the marginalization is postponed. Notice that a marginalization is always performed over a combination of at least two potentials. If $\|\text{dom}(\phi^{\downarrow T})\| \leq \delta$, then $\phi^{\downarrow T}$ is computed.

The results of experiments suggest that VE is the most suited marginalization operation to apply in the any-space scheme. Notice that neither RC nor VU is directly applicable as the marginalization operation in LP.

5 PERFORMANCE EVALUATION

This section presents the results of a preliminary performance evaluation[3]. The evaluation is performed using a set of real-world and randomly generated BNs. The set of real-world networks considered includes *Barley* and *ship-ship* while networks with $\|\mathcal{X}\| = 100, 125, 150, 200$ were generated randomly (ten networks of each size). For each network ten different $\mathcal{X}_\epsilon$ were generated randomly for each $\|\mathcal{X}_\epsilon\| = 0, \ldots, \|\mathcal{X}\|$. Table 1 (where $s(C) = \prod_{X \in C} \|X\|$ and $s(\mathcal{C}) = \sum_{C \in \mathcal{C}} s(C)$) contains statistics on some test networks (in the name *net_x_y* $x = \|\mathcal{X}\|$ and y is an identifier). The junction trees have been generated using *optimal triangulation* (total weight being the optimality criterion) (Jensen, 2007).

Table 1: Statistics on test networks.

Network	$\|V\|$	$\|\mathcal{C}\|$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$
ship-ship	50	35	4,032,000	24,258,572
Barley	48	36	7,257,600	17,140,796
net_100_5	100	85	98,304	311,593
net_200_5	200	178	15,925,248	70,302,065

This section also presents the results of an evaluation of the cost of the last and most expensive division operation involved in the elimination of a variable by AR. Ndilikilikesha (1994) introduces operations on the DAG structure where the need for division is eliminated. This is achieved by associating a potential instead of a CPD with each variable. This implies that barren variable elimination requires marginalization operations and it therefore becomes a potentially expensive operation.

[3]Due to space restrictions, a limited number of graphs are included for each experiment.

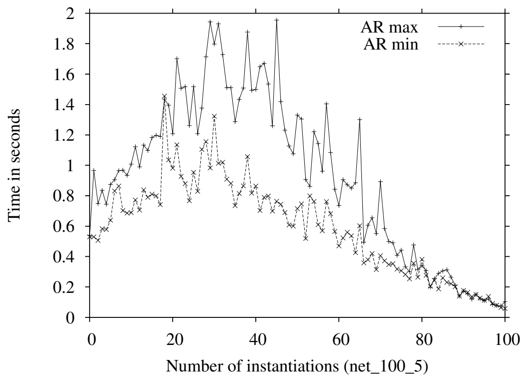

Figure 5: Time cost of LARP with sorting.

5.1 ARC-REVERSAL SORT

To assess the performance impact of ρ, we compare the costs of belief update using AR min and AR max. The *minimum fill-in-weight* rule selects as the next arc to reverse an arc with lowest cost while the *maximum fill-in-weight* rule selects an arc with highest cost. A performance comparison between AR min and AR max will give insights into the importance of selecting a *good* arc-reversal order.

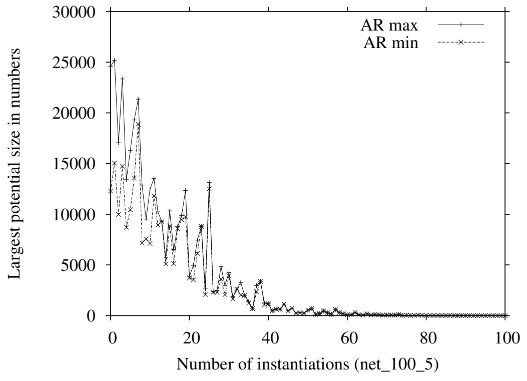

Figure 6: Space cost of LARP with sorting.

Figures 5 and 6 show the cost of belief update in *net_100_5*. The time cost of AR min is significantly lower than the time cost of AR max whereas the reduction in potential size is less significant and it is most significant for small subsets of evidence. Only in a few cases there is a reduction in the largest potential size when using AR min compared to using AR max.

The time cost improvement is not only produced by a reduction in the largest potential

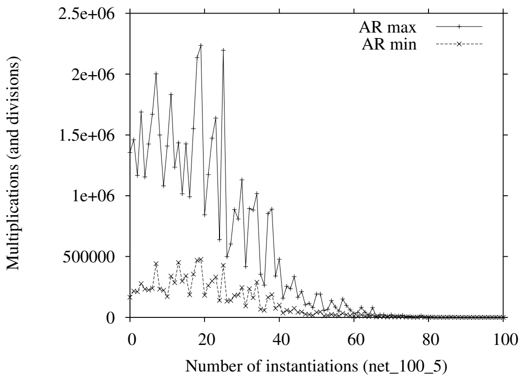

Figure 7: Multiplications & divisions, sorting.

size, but also by a reduction in the number of arithmetic operations performed. Figure 7 shows the cost of belief update in *net_100_5* in terms of the number of multiplications and divisions performed. There is a reduction in time cost and number of operations even though there is no reduction in the (average) size of the largest potential.

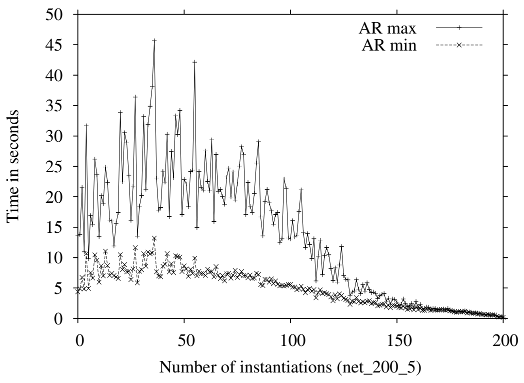

Figure 8: Time cost of LARP with sorting.

Figure 8 shows an example where the use of *minimum fill-in-weight* not only gives a significant reduction in time cost over the use of *maximum fill-in-weight*, but the variation of the cost is also significantly reduced.

We expected the implementation overhead introduced by the sorting algorithm to dominate the time efficiency improvement (e.g., testing for potential cycles in the graph), but this was clearly not the case. The arc-reversal order can have a significant impact on the time cost of belief update. In conclusion, it may be important to identify an efficient arc-reversal order.

5.2 ANY-SPACE

The any-space property is achieved by not constructing any potential ϕ with $\|\mathrm{dom}(\phi)\| > \delta$. To illustrate the any-space property, we performed a sequence of experiments with different δ values. Notice that reducing δ from y to x only has an impact on performance when at least one potential ϕ with $x < \|\mathrm{dom}(\phi)\| \leq y$ is created during belief update.

Figure 9 shows the time cost of belief update in *Barley* for three different δ values ($\max_{S \in \mathcal{S}} s(S) = 907{,}200$) using VE as the marginalization algorithm. The time cost increases as δ is reduced.

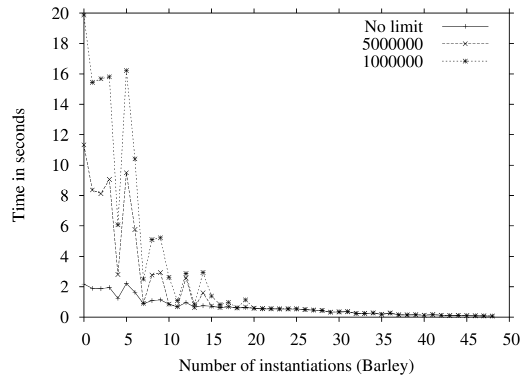

Figure 9: Time cost for different δ values.

The experiments show that the average largest potential size has a major variation. The peaks in the graphs are caused by a few *difficult* evidence scenarios[4]. The combined impact of these scenarios increases as δ decreases.

Table 2 shows the time cost for belief update using AR min and VE in *Barley* given two specific evidence scenarios as a function of δ. The time cost has a large variation across evidence scenarios and the time cost increases as δ decreases. Notice that the time costs for two different values of δ are (almost) equal. The reason is that the largest domain size created during belief update is the same in both cases.

[4]In this case the most expensive set of evidence to propagate consists of four instantiations of leaf variables with multiple parents which are inserted into four different leaf cliques. This evidence introduces additional dependence relations.

Table 2: Time cost of belief update given two different sets of evidence of equal size.

	10^6	$2.5 * 10^6$	$5 * 10^6$	No limit
AR	333.65	41.84	41.50	3.89
VE	18.57	12.35	12.27	2.45
	$5 * 10^3$	$1.5 * 10^4$	$3 * 10^4$	No limit
AR	1.82	1.25	1.25	0.53
VE	2.28	0.96	0.92	0.46

The results of the experiments indicate that the VE algorithm is better suited than AR for implementing upper-limit constraints. The AR algorithm performs additional calculations in order to maintain as many (conditional) independence statements as possible. This seems to penalize the algorithm under upper-limits constraints when compared to VE.

The table indexing for potentials with sizes larger than δ is naïve compared to the table indexing for potentials with sizes below δ. The former table indexing is expected to add an additional overhead to the time costs.

5.3 DIVISION OPERATION

Using AR as the marginalization operation requires one invocation of Equations 1 and 2 for each arc reversed except for the last arc where the invocation of Equation 2 can be skipped as the variable subsequently will be eliminated as barren (Madsen, 2006).

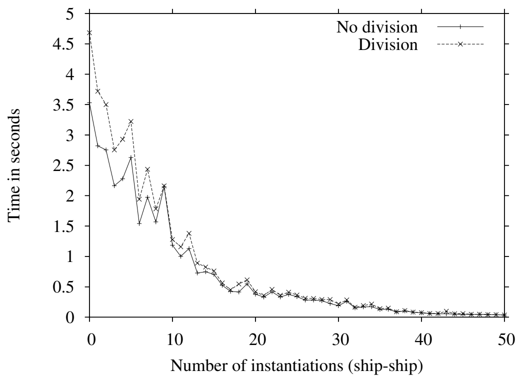

Figure 10: Time cost w/w.o. last division.

Figure 10 illustrates the cost of the division operation in the *ship-ship* network. It is clear from the figure that the cost of the division operation is most significant for the case of small sized evidence sets. The impact of the division operation is reduced as $||\mathcal{X}_\epsilon||$ increases.

6 CONCLUSION

This paper introduces LP as a class of algorithms for computing all-marginals. The elements of the class differ with respect to the algorithm or algorithms used for message computation. We have proposed two methods for message computation in LP and considered the importance of certain properties of the algorithm. One method is based on sorting arc-reversal operations according to a complexity score while the second method is a simple scheme for extending LP with the any-space property. The paper includes an experimental evaluation of the proposed extensions.

Future work includes a more in-depth analysis of the any-space potential of LP in message computation. This includes the option of reconsidering the calculation of a factor at a later point in time. For instance, before accessing the elements of a delayed factor ϕ recursively, it may be possible to identify a more efficient elimination and combination order from the source potentials of ϕ. Future work also includes an analysis of methods for selecting between different algorithms for solving a query. This would produce a propagation scheme where different algorithms may be used to solve different queries during belief update.

References

S. Arnborg, D. G. Corneil, and A. Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284.

F. Bacchus, S. Dalmao, and T. Pitassi. 2003. Value Elimination: Bayesian Inference via Backtracking Search. In *Proc. of UAI*, pages 20–28.

C. Cannings, E. A. Thompson, and H. H. Skolnick. 1978. Probability functions on complex pedigrees. *Advances in Applied Probability*, 10:26–61.

G. F. Cooper. 1990a. Bayesian Belief-Network Inference Using Recursive Decomposition. Technical Report KSL 90-05, Knowledge Systems Laboratory.

G. F. Cooper. 1990b. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems.* Springer-Verlag.

P. Dagum and M. Luby. 1993. Approximating probabilistic inference in Bayesian belief netwoks is NP-hard. *Artificial Intelligence*, 60:141–153.

A. Darwiche and G. Provan. 1997. Query dags: A pratical paradigm for implementing belief-network inference. In *JAIR*, pages 147–176.

A. Darwiche. 2000. Any-Space Probabilistic Inference. In *Proc. of UAI*, pages 133–142.

R. Dechter. 1996a. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. of UAI*, pages 211–219.

R. Dechter. 1996b. Topological Parameters for time-space tradeoff. In *Proc. of UAI*, pages 220–227.

F. V. Jensen and F. Jensen. 1994. Optimal junction Trees. In *Proc. of UAI*, pages 360–366.

F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian Updating in Causal Probabilistic Networks by Local Computations. *Computational Statistics Quarterly*, 4:269–282.

F. Jensen. 2007. *HUGIN API Reference Manual.* Available from http://www.hugin.com.

U. B. Kjærulff and A. L. Madsen. 2008. *Bayesian Networks and Influence Diagrams - A Guide to Construction and Analysis.* Springer-Verlag.

U. B. Kjærulff. 1993. *Aspects of efficiency improvement in Bayesian networks.* Ph.D. thesis, Department of Computer Science, Aalborg University, Denmark, April.

S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, B*, 50(2):157–224.

A. L. Madsen and F. V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245.

A. L. Madsen. 2006. Variatoins Over the Message Computation Algorithm of Lazy Propagation. *IEEE Transactions on Systems, Man. and Cybernetics Part B*, 36(3):636–648.

P. Ndilikilikesha. 1994. Potential influence diagrams. *IJAR*, 11(1):251–285.

S. M. Olmsted. 1983. *On representing and solving decision problems.* Phd thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligence Systems.* Series in Representation and Reasoning. Morgan Kaufmann Publishers.

R. D. Shachter, B. D'Ambrosio, and B. Del Favero. 1990. Symbolic probabilistic inference in belief networks. In *Proc. of 8th National Conference on AI*, pages 126–131.

R. D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34(6):871–882.

G. R. Shafer and P. P. Shenoy. 1990. Probability Propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–351.

P. P. Shenoy. 1997. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *IJAR*, 17(2-3):239–263.

M. Yannakakis. 1981. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79.

N. L. Zhang and D. Poole. 1996. Exploiting Causal Independence in Bayesian Network Inference. *Journal of Artificial Intelligence Research*, 5:301–328.