International Journal of Approximate Reasoning 51 (2010) 499–514

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Improvements to message computation in lazy propagation

A.L. Madsen

HUGIN EXPERT A/S, Gasværksvej 5, DK-9000 Aalborg, Denmark

ARTICLE INFO

Article history:
Received 16 November 2008
Received in revised form 1 June 2009
Accepted 25 August 2009
Available online 28 January 2010

Keywords:
Bayesian network
Belief update
Lazy propagation

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): (1) we introduce myopic methods for sorting the operations involved in a variable elimination using arc-reversal and (2) 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 [27,7,19] as a formalism for modelling 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* [5,8], 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 100,000 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. Similarly, the performance of belief update is important when doing parameter estimation using the EM algorithm as the EM algorithm makes one propagation for each (incomplete) case for each iteration. 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 [27], Arc-Reversal (AR) [26,28,29], Symbolic Probabilistic Inference (SPI) [30], Recursive Decomposition (RD) [4], Variable Elimination (VE) [3,34], Bucket Elimination [11], the Fusion operator [32], Query DAGs [10], Recursive Conditioning (RC) [9] and Value Elimination (VU) [1] while the latter class contains, for instance, Lauritzen–Spiegelhalter [21], HUGIN [17], and Shenoy–Shafer [31].

LP [24] 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 [22] introduced LP algorithms where either AR, VE or SPI is used for message and marginal computation in a variable elimination

E-mail address: Anders.L.Madsen@hugin.com
URL: http://www.hugin.com

0888-613X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2010.01.009

based approach. The empirical results reported in [22] illustrates the relative performance of using AR, VE or SPI for message passing and marginal computation on a set of random and real-world networks. The experiments in [22] show that no algorithm dominates or is dominated by any of the other two algorithms.

The AR algorithm has a number of important advantages over VE and SPI when used for belief update in Bayesian networks (and solving influence diagrams [14]). The AR algorithm maintains a valid Bayesian network (of influence diagram) structure during the belief update process. This implies that more barren variables can be identified. This, in some cases, means improved time and space performance as described in [22]. Furthermore, AR is receiving an increased level of attention as a solution algorithm for Bayesian networks and influence diagrams with mixed variables, see e.g., [6,22,33,23].

In this paper, a number of improvement to the AR algorithm is suggested: (1) myopic methods for selecting the next edge to reverse using AR operations to perform a variable elimination and (2) a method to extend LP with the any-space property. The paper includes an empirical performance analysis of the suggested methods.

2. Preliminaries

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

$$P(\mathcal{X}) = \prod_{X \in \mathcal{X}} P(X|\mathrm{pa}(X)).$$

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 $P(X|\epsilon)$ for each $X \in \mathcal{X}$. Evidence $\epsilon = \left\{\epsilon_{X_{i_1}}, \ldots, \epsilon_{X_{i_n}}\right\}$ 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.

A probability *potential* [31] 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* $\mathrm{dom}(\phi) = \{X_1, \ldots, X_n\}$, we let $H(\phi)$ denote the *head* (i.e., the conditioned variables) and $T(\phi)$ denote the *tail* (i.e., the conditioning variables) of ϕ.

The *domain graph* $\mathcal{G}(\Phi) = (\mathcal{X}, \mathcal{E})$ of a set of probability potentials Φ over variables $\mathcal{X}$ is the graph spanned by $\mathcal{X}$ where for each $\phi \in \Phi$ an undirected edge is added between each pair of variables $X,\ Y \in H(\phi)$ and a directed edge is added from each $X \in T(\phi)$ to each $Y \in H(\phi)$. We let $\mathrm{dom}(\Phi)$ denote the set of domain variables of potentials in Φ.

Definition 2.1 (*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 $E = \mathrm{dom}(\Phi) \setminus T$ is referred to as the *elimination set*. The set of potentials Φ^* obtained by eliminating $\mathrm{dom}(\Phi) \setminus T$ from Φ s.t. $\prod_{\phi \in \Phi^*} \phi = \phi(T, \epsilon'|\epsilon'')$ where $\epsilon = \epsilon' \cup \epsilon''$ is a *solution* to query Q (where ϵ' is the subset of ϵ represented in the head of a potential in Φ and $\epsilon'' = \epsilon \setminus \epsilon'$). 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 $\Phi_X = \{\phi \in \Phi : X \in \mathrm{dom}(\phi)\}$.

In the process of eliminating E from Φ, it may be possible to eliminate some variables without performing any computations. These variables are referred to as *barren variables*:

Definition 2.2 (*Barren variable*). A variable X is a *barren* w.r.t. a set $T \subseteq \mathcal{X}$, evidence ϵ, and DAG $\mathcal{G}$, if $X \notin T,\ X \notin \mathcal{X}_\epsilon$ and X only has barren descendants in $\mathcal{G}$ (if any).

The notion of barren variables can be extended to graphs with both directed and undirected edges [22].

2.1. Solving queries

Query-based belief update algorithms solve a single query $Q = (T, \Phi, \epsilon)$ where the goal is to compute the posterior distribution of a set of variable T given evidence ϵ (usually $|T| = 1$). This means that query-based belief update algorithms can exploit irrelevance and independence properties in a preprocessing step. This amounts to removing barren variables as well as removing variables, which are separated from the target set T given ϵ.

In the following description of query-based algorithms 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 $\mathrm{pa}(X) = \{Y, J, K\}$ and let $\mathrm{pa}(Y) = \{I, J\}$. An AR operation on arc (Y, X) is performed as follows:

$$P(X|I,J,K) = \sum_Y P(Y|I,J)P(X|Y,J,K), \tag{1}$$

$$P(Y|X,I,J,K) = \frac{P(Y|I,J)P(X|Y,J,K)}{P(X|I,J,K)}. \tag{2}$$

The AR operation corresponds to arc-reversal in $\mathcal{G}(\Phi)$, see Fig. 1. The two operations (1) and (2) are only valid when X and Y are adjacent nodes in an ordering compatible with the arcs of a Bayesian network.

Using VE Y is eliminated from Φ by marginalisation of Y over the combination of potentials of $\Phi_Y = \{\phi \in \Phi : Y \in \mathrm{dom}(\phi)\}$ and setting Φ^* as:

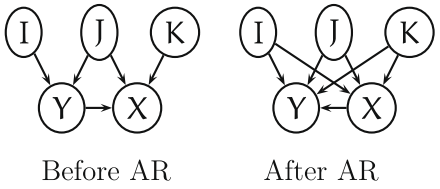

Fig. 1. The AR operation corresponds to arc-reversal in $\mathcal{G}(\Phi)$.

$$\phi_Y = \sum_Y \prod_{\phi \in \Phi_Y} \phi,$$
$$\Phi^* = \Phi \setminus \Phi_Y \cup \{\phi_Y\}.$$

The SPI algorithm views inference as a combinatorial optimisation problem, the optimal factoring problem, where potentials are combined pairwise eliminating variables not in the target when possible. If Φ is the set of potentials relevant for computing a marginal $P(X_1, \ldots, X_n)$, then SPI computes this marginal by combining potentials of Φ pairwise recursively and eliminating variables not in $P(X_1, \ldots, X_n)$ when possible. Binary join trees (BJT) [32], on the other hand, are based on variable elimination using a binary tree to guide the combination of potentials when more than two potentials need to be combined in the process of eliminating a variable. Notice the difference between BJT and SPI. In SPI two potentials not sharing any domain variables may be combined. This is not the case in BJT where only potentials related to the elimination of a variable are combined.

There is a rich literature on any-space algorithms for belief update in Bayesian networks, see e.g., [12,9,1]. RC is an any-space algorithm for exact query-based belief update based on recursive conditioning [9]. RC uses the idea of conditioning to recursively decompose the network until queries over single node networks are reached. RC is an instantiation of the family of algorithms referred to as VU [1]. A major difference between VU and RC is that VU supports a dynamic conditioning order, whereas the conditioning order is fixed in RC. RD [4], 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 [16].

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 sub-graph in an undirected graph[1] $\mathcal{G}^T$. The link between two neighbouring 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. $\mathrm{fa}(X) \subseteq C$ where $\mathrm{fa}(X) = \{X\} \cup \mathrm{pa}(X)$. 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. The message passed from clique A to clique B is computed by eliminating variables from a combination of potentials associated with A and messages received from neighbouring cliques except B. 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

A junction tree representation $\mathcal{T}$ of a Bayesian network $\mathcal{N}$ by construction supports the computation of any posterior marginal given any subset of evidence, i.e., $\mathcal{T}$ can facilitate the calculation of $P(X|\epsilon)$ for any $X \in \mathcal{X}$ and any evidence ϵ through the explicit representation of all possible dependence relations in $\mathcal{G}$ given any ϵ. This means that $\mathcal{T}$ often maintains too many dependence relations to take advantage of independence and irrelevance properties induced by the structure of $\mathcal{G}$ and a specific set of evidence ϵ. LP aims at taking advantage of independence and irrelevance properties in a Shenoy–Shafer message passing scheme [22,24].

In LP the set of CPDs Φ_C associated with C during initialisation are not combined to form the initial clique potential ϕ_C. Instead set Φ_C is maintained as a decomposition of ϕ_C. As part of the initialisation process CPDs are instantiated to reflect the evidence ϵ. The decomposition of clique (and separator) potentials enables LP to take advantage of independence and irrelevance properties induced by the evidence ϵ and the structure of $\mathcal{G}$.

[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.

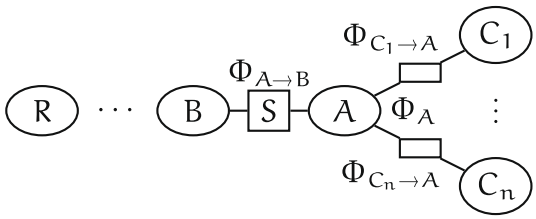

Fig. 2. $\Phi_{A\to B}$ is passed from A to B.

Message passing in LP proceeds according to the Shenoy–Shafer scheme: A clique A sends a message $\Phi_{A\to B}$ to its neighbour B when it has received messages from all neighbours (denoted $\mathrm{ne}(A)$) except B, see Fig. 2. A message $\Phi_{A\to B}$ is the solution to a query:

$$Q = \left(B, \Phi_A \cup \bigcup_{C\in \mathrm{ne}(A)\setminus B} \Phi_{C\to A}, \epsilon\right)$$

and it is computed as:

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

where M is the marginalisation 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 $\mathcal{G}\left(\Phi_A \cup \bigcup_{C\in \mathrm{ne}(A)\setminus B}\Phi_{C\to A}\right)$ are removed. Notice that $\Phi_{A\to B}$ and Φ_C are sets of potentials. The content of $\Phi_{A\to B}$ depends on marginalisation algorithm M.

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 variable elimination order in the computation of $P(X|\epsilon)$ 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 [22]. Evidence is entered in $\mathcal{T}$ by instantiating $\mathcal{X}_\epsilon$ according to ϵ.

Belief update proceeds in two steps: (1) an inward and outward message passing over the separators of $\mathcal{T}$ relative to a root of $\mathcal{T}$ (*message passing*) and (2) computation of $P(X|\epsilon)$ for each $X \in \mathcal{X}$ (*marginal computation*).

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 $|\mathrm{ch}(Y)| > 1$, then an order $\rho = ((Y, X_1), \ldots, (Y, X_{|\mathrm{ch}(Y)|}))$ has to be determined, see Fig. 3.

Different arc-reversal orders may produce different solutions Q. Consider the domain graph of Φ depicted in Fig. 4 over five variables and assume:

$$Q = (\{X_1, X_3, X_4, X_5\}, \Phi, \emptyset),$$

where

$$\Phi = \{P(X_1), P(X_2|X_1), P(X_3|X_2, X_5), P(X_4|X_2), P(X_5)\}.$$

Eliminating X_2 using AR involves reversing arcs (X_2, X_3) and (X_2, X_4). Figs. 5 and 6 show the calculations for the two possible orders $\rho_{\min} = ((X_2, X_4), (X_2, X_3))$ and $\rho_{\max} = ((X_2, X_3), (X_2, X_4))$, 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. Fig. 5 illustrates:

$$\Phi^{\mathrm{ARmax}\downarrow T} = \{P(X_1), P(X_3|X_1, X_5), P(X_4|X_1, X_3, X_5), P(X_5)\},$$

while the solution illustrated in Fig. 6 is:

$$\Phi^{\mathrm{ARmin}\downarrow T} = \{P(X_1), P(X_3|X_1, X_4, X_5), P(X_4|X_1), P(X_5)\}.$$

The difference between the solutions $\Phi^{\mathrm{ARmin}\downarrow T}$ and $\Phi^{\mathrm{ARmax}\downarrow T}$ is the set of tail variables of the potentials with X_3 and X_4 as head variables where $\mathrm{dom}(P(X_4|X_1, X_3, X_5)) = \mathrm{dom}(P(X_3|X_1, X_4, X_5))$ while $\|\mathrm{dom}(P(X_4|X_1))\| < \|\mathrm{dom}(P(X_3|X_1, X_5))\|$ when we assume $\|X_i\| = \|X_j\|$ for $i, j = 1, \ldots, 5$.

The (unique) solution to Q obtained using VE and the algorithm of Section 4 in [22] is:

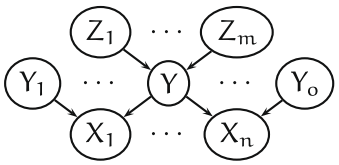

Fig. 3. If $|\mathrm{ch}(Y)| > 1$, then an arc-reversal order $\rho = ((Y, X_1), \ldots, (Y, X_{|\mathrm{ch}(Y)|}))$ has to be determined.

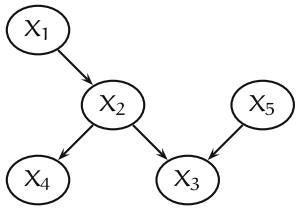

Fig. 4. Domain graph for Φ.

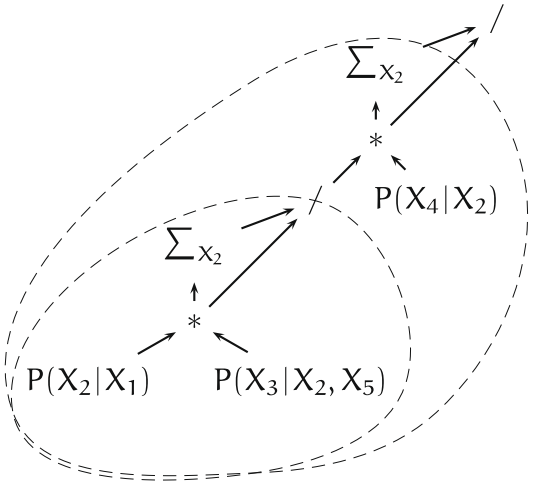

Fig. 5. Maximum fill-in-weight.

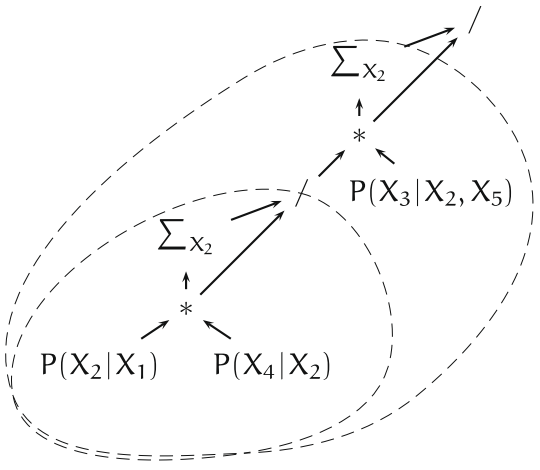

Fig. 6. Minimum fill-in-weight.

$$\Phi^{\mathrm{VE}\downarrow T} = \{P(X_1), P(X_3, X_4|X_1, X_5), P(X_5)\}.$$

The elimination of Y by a sequence of AR operations performed according to $\rho = ((Y, X_1), \ldots, (Y, X_{|\mathrm{ch}(Y)|}))$ will induce a set of new edges. The order ρ determines the set of induced edges and the set of induced edges has an impact on the performance of belief update. In the process of eliminating Y by a sequence of AR operations, the sequence $\hat{\rho}$ leading to the best time and space performance should be chosen. Since it is not possible by local computations only to identify the sequence $\hat{\rho}$ having the best time and space cost, the focus is on identifying the sequence ρ with minimum local cost.

Four different score functions[2] for computing the cost of a sequence ρ are considered: *fill-in*, *fill-in-weight*, *cpd-weight* and *number-of-parents* where the weight of an edge (Z_i, Z_j) is defined as $\|Z_i\|\|Z_j\|$. Each score function is defined w.r.t. to the result obtained after reversing arcs according to ρ. The score function *fill-in* is defined as the number of new edges, *fill-in-weight* is defined as the sum of the weights of the new edges, *number-of-parents* is defined as the size of $\mathrm{pa}(Y)$ and *cpd-weight* is defined as the total state space size of $P(Y|\mathrm{pa}(Y))$.

Under the *fill-in* score function, the cost of reversing (Y, X) is defined as:

$$s(Y, X) = |\{(Z_X, Y) : Z_X \in \mathrm{pa}(X) \setminus \mathrm{pa}(Y) \cup \{Y\}\} \cup \{(Z_Y, X) : Z_Y \in \mathrm{pa}(Y) \setminus \mathrm{pa}(X)\}|,$$

i.e., the cost is equal to the number of edges induced by new parents of X and Y. Under the *fill-in-weight* score function, the cost of reversing (Y, X) is defined as:

$$s(Y, X) = \sum_{Z_X \in \mathrm{pa}(X) \setminus \mathrm{pa}(Y) \cup \{Y\}} \|Z_X\| \cdot \|Y\| + \sum_{Z_Y \in \mathrm{pa}(Y) \setminus \mathrm{pa}(X)} \|Z_Y\| \cdot \|X\|,$$

i.e., the cost is equal to the sum of the weights of the edges induced by new parents of X and Y. Under the *number-of-parents* score function, the cost of reversing (Y, X) is defined as $s(Y, X) = |\mathrm{pa}(Y) \cup \mathrm{pa}(Y)|$, i.e., the cost is $|\mathrm{pa}(Y)|$ after reversing (Y, X). Finally, under the *cpd-weight* score function, the cost of reversing an edge (Y, X) is defined as $s(Y, X) = \|\mathrm{fa}(Y)\|$, i.e., the cost is equal to the total space size of $P(Y|\mathrm{pa}(Y))$ after reversing (Y, X).

The objective of considering different AR sequences is to minimise 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 $n = |\mathrm{ch}(Y)|$. Some of the sequences may be illegal due to the graph acyclicity constraint though. The large number of possible sequences means that the use of ordering heuristics is justified.

To investigate the impact of the sequence ρ on performance of belief update, we consider both minimising and maximising the cost functions. Notice that the score is only used to select the next edge to reverse in the process of eliminating Y. The score is not used to identify the variable elimination order. The result of eliminating Y from the set Φ_Y using VE is invariant under the order in which calculations are performed (i.e., the order in which potentials are combined using, e.g., BJT [32]). This is not the case for AR as the example above illustrates. The main purpose of this paper is to investigate the opportunities for exploiting this degree of freedom to improve performance of belief update using LP in combination with AR.

All rules use a myopic approach where the edge to reverse is selected based on its score. This means that they do not always find the optimal order (according to the cost function).

4.2. AR & VE combination

The advantage of AR over VE as the marginalisation operator is that AR maintains a valid DAG structure over the set of potentials during the belief update process. This means that $\Phi^{\mathrm{AR}\downarrow T}$ may represent a larger number of independence statements than $\Phi^{\mathrm{VE}\downarrow T}$, i.e., in the process of solving subsequent queries it may be possible to identify more independence relations and barren variables using AR for marginalisation compared to using VE. Maintaining a valid DAG structure is important for message passing and for identifying the potentials $\mathcal{R}_X$ relevant for computing a marginal $P(X|\epsilon)$. Maintaining a valid DAG structure in the solution of $Q = (\{X\}, \Phi, \epsilon)$ for each $X \in \mathcal{X}$ may, however, not be worth the additional computational cost as $\mathcal{R}_X \subseteq \Phi$ includes only the subset of Φ relevant for computing $P(X|\epsilon)$ and $\mathcal{R}_X^{\downarrow X}$ is not used in any subsequent calculations.

Inspired by the work of Butz and Hua [2], we consider the combination of AR and VE where AR is used for computing messages and VE is used for computing marginals. The following example adopted from [2] illustrates how AR may perform unnecessary calculations in the process of maintaining a valid BN structure at all times. Consider the elimination of X_1 from $\Phi = \{P(X_1), P(X_2|X_1), P(X_3|X_1, X_2)\}$ with $T = \{X_2, X_3\}$ and assume the arc (X_1, X_2) is reversed first. The solution to the query is:

$$\begin{aligned}\Phi^* = \Phi^{\mathrm{AR}\downarrow T} &= \left\{\sum_{X_1} P(X_1)P(X_2|X_1), \sum_{X_1} P(X_3|X_1, X_2)\frac{P(X_1)P(X_2|X_1)}{\sum_{X_1} P(X_1)P(X_2|X_1)}\right\} = \left\{P(X_2), \frac{1}{P(X_2)}\sum_{X_1} P(X_3|X_1, X_2)P(X_1)P(X_2|X_1)\right\} \\ &= \{P(X_2), P(X_3|X_2)\}.\end{aligned}$$

[2] We consider scores similar to the *fill-in*, *clique-size*, *fill-in-weight* and *clique-weight* scores often used for identifying triangulations using myopic node elimination. See [18] for empirical evidence on the performance of *fill-in*, *clique-size* and *clique-weight*.

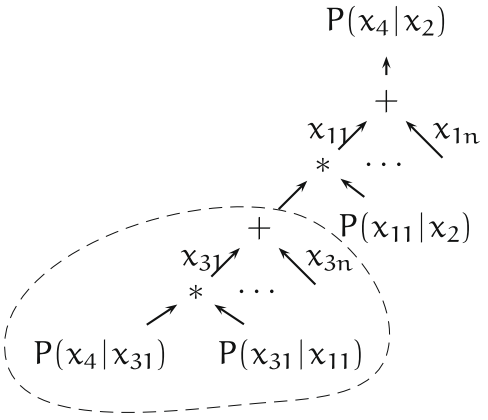

Fig. 7. VE calculation of $P(X_4|X_2)$.

It is clear that future calculations involving Φ^* will mean that the potential $P(X_2)$ is both multiplied in to and divided out of the calculations. This is not the case when VE is used as the marginalisation operation as the result of eliminating X_1 is:

$$\Phi^{\mathrm{VE}\downarrow T} = \left\{ \sum_{X_1} P(X_1)P(X_2|X_1)P(X_3|X_1,X_2) \right\} = \{P(X_2,X_3)\}.$$

Butz and Hua [2] describes an extension of Lazy AR propagation (LARP) referred to as Lazy-ARVE propagation where AR operations are used to identify the set $\Phi^* = \Phi^{\downarrow T}$ while VE is used to compute each $\phi \in \Phi^*$. Unfortunately, Butz and Hua [2] includes only a limited assessment of Lazy-ARVE using a simple network, includes no empirical assessment on larger networks nor does it discuss how to solve problems related to evidence. The advantage of applying VE for marginal calculation only is that no problems related to handling evidence arise.

The SPI algorithm may be applied for binary combination of potentials when applicable, see [22] for an experimental assessment of applying the principle of SPI (and BJT) in VE. The approach of Butz and Hua [2] is similar to the approach taken for Lazy SPI propagation [22] where the content of $\Phi_{A\to B}$ is defined by variable elimination, but computed using SPI.

4.3. 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 $\|\mathrm{dom}(\phi)\| > \delta$ where δ is a threshold value on the size of potentials. Instead of 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 marginalisation. Let ϕ_1 and ϕ_2 be two potentials. If $\mathrm{dom}(\phi_1) \setminus \mathrm{dom}(\phi_2) \neq \emptyset$ or $\mathrm{dom}(\phi_2) \setminus \mathrm{dom}(\phi_1) \neq \emptyset$, then $\|\mathrm{dom}(\phi_1 \cdot \phi_2)\| > \|\mathrm{dom}(\phi_i)\|$ 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 $\|\mathrm{dom}(\prod \phi)\| > \delta$ or $\|\mathrm{dom}(\phi^{\downarrow T})\| > \delta$, respectively. Notice that only marginalisation can enforce the construction of a potential, whereas both a marginalisation and a product may involve delayed potentials producing a recursive scheme. Fig. 7 illustrates the approach on calculation $P(X_4|X_2)$ by eliminating $\{X_1, X_3\}$ from $\Phi = \{P(X_1|X_2), P(X_3|X_1), P(X_4|X_3)\}$ using VE:

$$P(X_4|X_2) = \Phi^{\mathrm{VE}\downarrow\{X_2,X_4\}} = \sum_{X_1} P(X_1|X_2) \sum_{X_3} P(X_3|X_1)P(X_4|X_3). \tag{3}$$

Each entry $P(x_4|x_2)$ is computed by accessing and combining the values of its source potentials Φ recursively according to Eq. (3).

Eq. (3) 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 means that the calculation of an entry is delayed until the entry is accessed as part of the calculation of another potential.

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

The results of experiments suggest that VE is the best suited marginalisation operation to apply in the any-space scheme, see Table 2 for an example. Notice that neither RC nor VU is directly applicable as the marginalisation operation in LP.

Table 1
Statistics on test networks.

Network	$\|V\|$	$\|\mathcal{C}\|$	$\max_{C\in\mathcal{C}} s(C)$	$s(\mathcal{C})$
Barley	48	36	7,257,600	17,140,796
ship-ship	50	35	4,032,000	24,258,572
net_100_5	100	85	98,304	311,593
net_150_1	150	131	3,538,944	9,946,960
net_200_5	200	178	15,925,248	70,302,065

5. Performance evaluation

This section presents the results of a performance evaluation using a set of real-world and randomly generated BNs.[3] The experiments reported here include examples of performance in terms of time cost, space cost, number of multiplications and divisions performed, and the number of barren variables identified during belief update. Time and space costs are important from a practical point of view for obvious reasons, whereas the number of multiplications and divisions is related to both measures. The number of barren variables identified is the measure, the AR algorithm may improve to improve both time and space costs.

The set of real-world networks considered includes *Barley* [20] and *ship-ship* [13] while networks with $\|\mathcal{X}\| = 100, 125, 150, 200$ were generated randomly (10 networks of each size). For each network 10 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.[4] The junction trees were generated using *optimal triangulation* (total weight being the optimality criterion) [15].

5.1. Arc-reversal sort

To assess impact of ρ on performance, we compare the costs of belief update using the four different score functions described above when selecting the next arc to reverse. For each score we consider a heuristic for minimising the score and a heuristic for maximising the score. A performance comparison between minimising and maximising each score function will give insights into the importance of selecting a *good* arc-reversal order.

Fig. 4 illustrates the potential advantage of selecting sequence ρ carefully, whereas the network shown in Fig. 8 illustrates the advantage of considering AR as an alternative to VE for message and marginal computation. During belief update, variable $X_{1,1}$ should be eliminated in order to compute $P(X_{4,1})$. Using VE this would produce a potential $\phi(X_{2,1}, X_{2,2}|X_{1,1}, X_{1,3})$, whereas using AR the result would be either $\{\phi(X_{2,1}|X_{1,1}, X_{1,3}, X_{2,2}), \phi(X_{2,2}|X_{1,1}, X_{1,3})\}$ or $\{\phi(X_{2,1}|X_{1,1}, X_{1,3}), \phi(X_{2,2}|X_{1,1}, X_{1,3}, X_{2,1})\}$.

The difference between using VE and AR is that some information is lost when VE is used. The latter factorisation obtained from using AR would make it possible to detect $X_{2,2}$ as a barren variable in cases where both $X_{1,2}$ and $X_{2,2}$ are not barren. This makes AR run faster and use less space than VE on this network.

Figs. 9–11 compare the performance of VE and AR on this structure where we assume $\|X_{i,j}\| = 10$. The reader is referred to [22] for additional empirical evidence on the relative performance of AR and VE as the message and marginal computation algorithm in LP.

It is clear from the figures that AR performs better than VE with respect to all three performance measures on this example. The size of the largest potential created during inference is invariant under the two methods.

Figs. 12 and 13 show the cost of belief update in *net_100_5*. The time costs of heuristics minimising the score function are significantly lower than the time costs of heuristics maximising the score function, whereas the reductions in potential sizes are less significant and most significant for small subsets of evidence. Only in a few cases there is a reduction in the largest potential size when minimising the score compared to maximising the score.

The time cost improvements are not only produced by a reduction in the largest potential sizes, but also by a reduction in the number of arithmetic operations performed. Fig. 14 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.

Fig. 15 shows an example where minimising the score function not only significantly reduces time cost over maximising the score, but the variation of the cost is also significantly reduced.

Fig. 16 shows another example where minimising the score function has improved performance over maximising the score function. On this network, the cost of belief update using AR is comparable to the cost of belief update using VE. Fig. 17 shows that the different variants of AR are able to identify more barren variables than VE. The size of the largest potential created during inference is on this network the same for the methods considered.

[3] Due to space restrictions, a limited number of graphs are included for each experiment.
[4] A network *net_x_y* has $x = \|\mathcal{X}\|$ variables while y is an identifier.

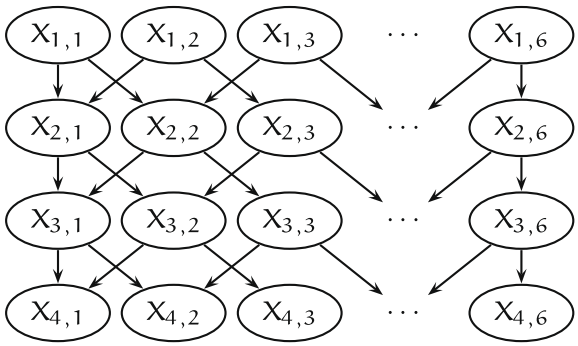

Fig. 8. An example where AR performs better than VE.

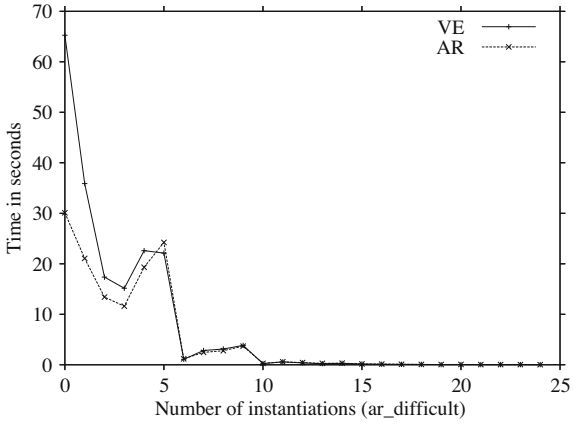

Fig. 9. Time cost for LARP.

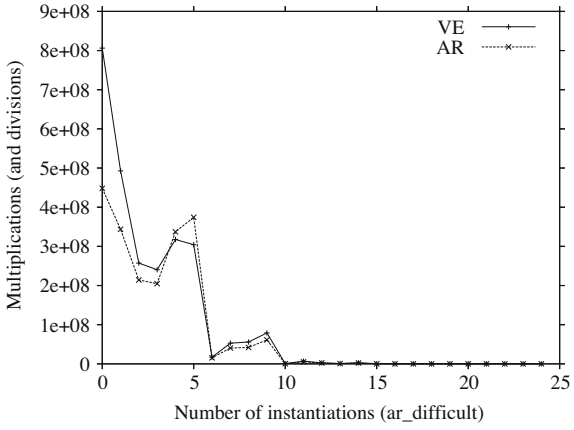

Fig. 10. Multiplications & divisions.

The experiments show the order ρ may have a significant impact on the performance of belief update using AR. The four different score functions have similar performance with no single measure dominating the remaining measures on all networks. There is a significant difference between minimising and maximising the score functions. This leads to the conclusion that the arc-reversal order is important for the performance of belief update.

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.

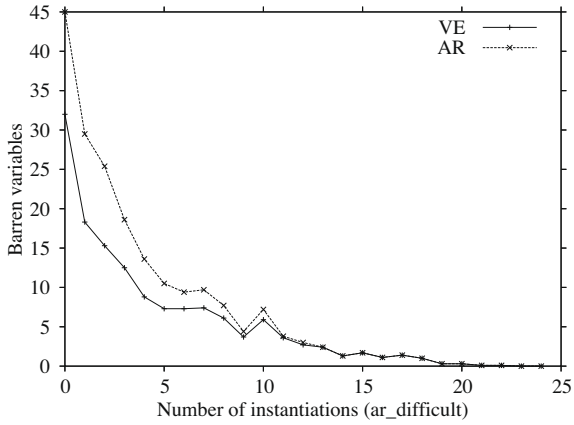

Fig. 11. Barren variables.

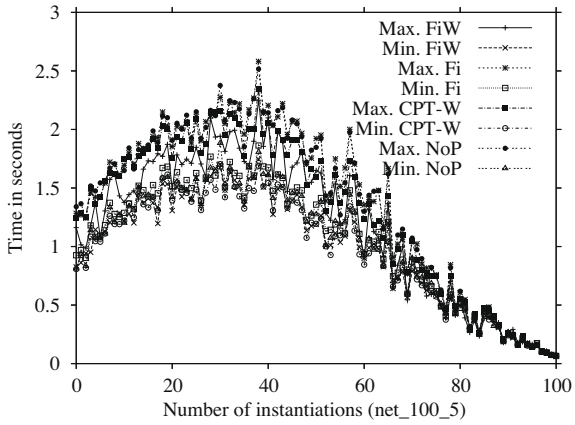

Fig. 12. Time cost of LARP with sorting.

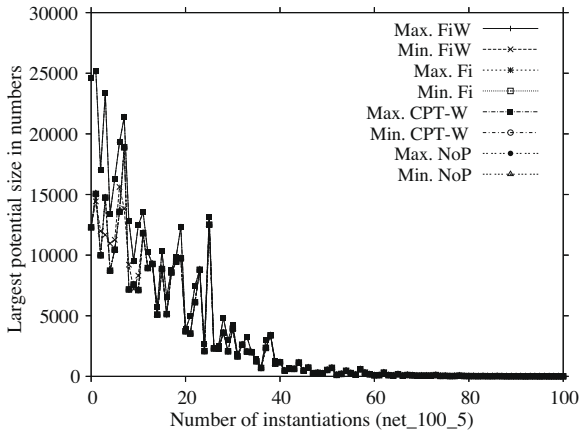

Fig. 13. Space cost of LARP with sorting.

The experiments reported here include examples where the time performance of LP with AR is comparable to the time performance of LP with VE or in one case better. This is not always the case though. We refer to [22] for a comparison between the use of VE, SPI and AR for belief update.

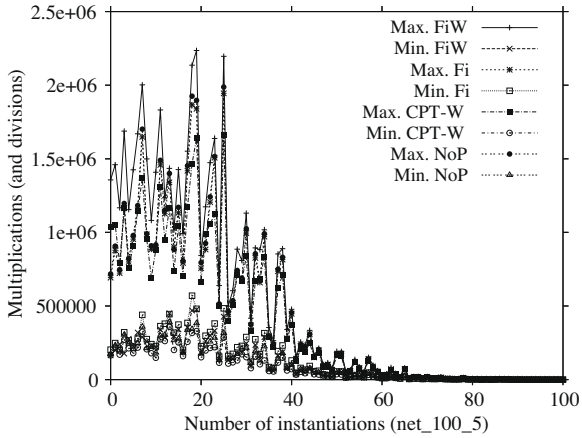

Fig. 14. Multiplications & divisions, sorting.

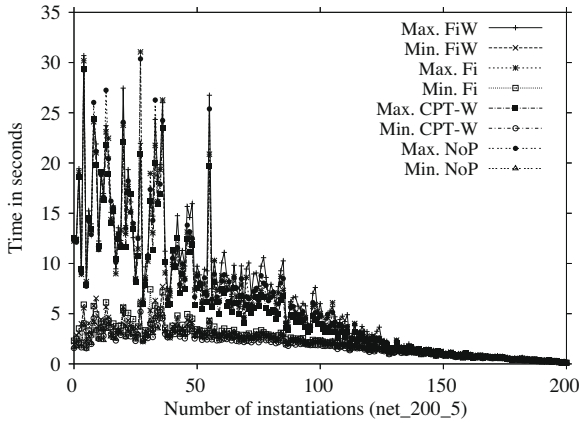

Fig. 15. Time cost of LARP with sorting.

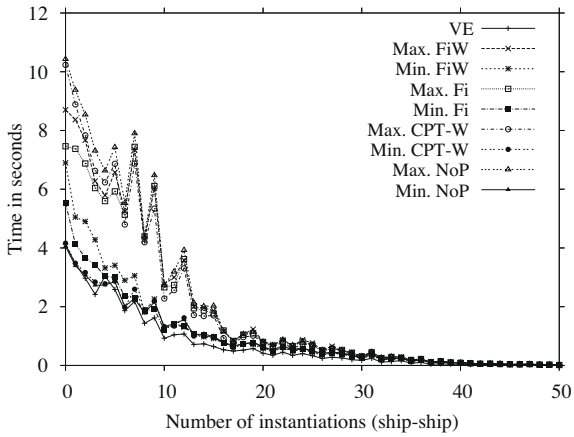

Fig. 16. Time cost of LARP with sorting.

5.2. AR & VE combination

To assess the cost of maintaining a DAG structure during belief update, we compare the belief update cost of LARP with the belief update cost of LARP for message computation and LP VE for marginal computation.

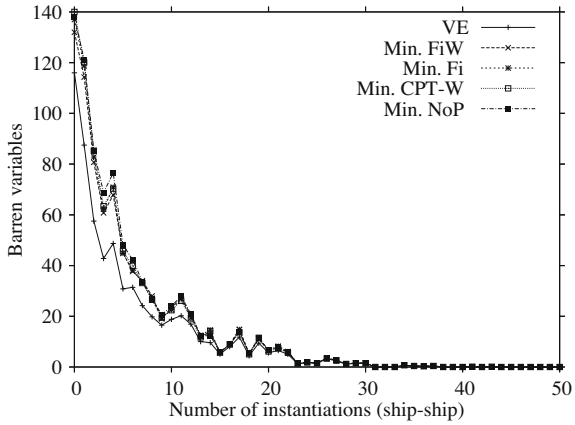

Fig. 17. Barren variables.

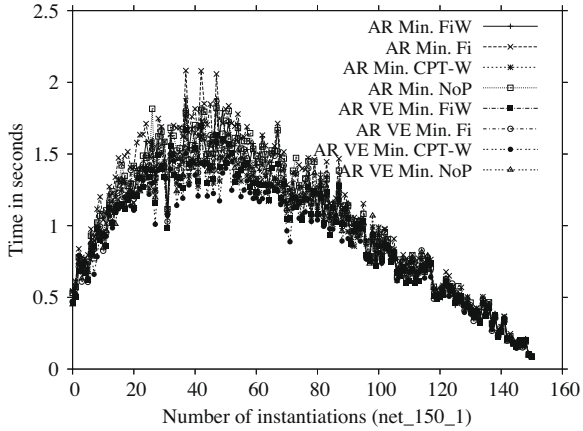

Fig. 18. Time cost of message passing and computing marginals (belief update).

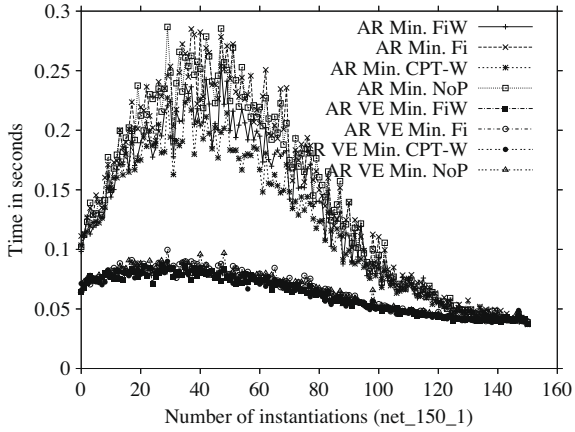

Fig. 19. Time cost of computing marginals (after message passing).

Fig. 18 shows the time cost of belief update, i.e., message passing and computation of marginals, in *net_150_1* using LARP and LP with AR and VE combined. The difference in performance is negligible.

Fig. 19 shows the time cost of computing marginals after message passing in *net_150_1* using LP with AR and LP with AR and VE combined. It is clear that there is a difference in performance of the algorithms in this case. The cost of computing

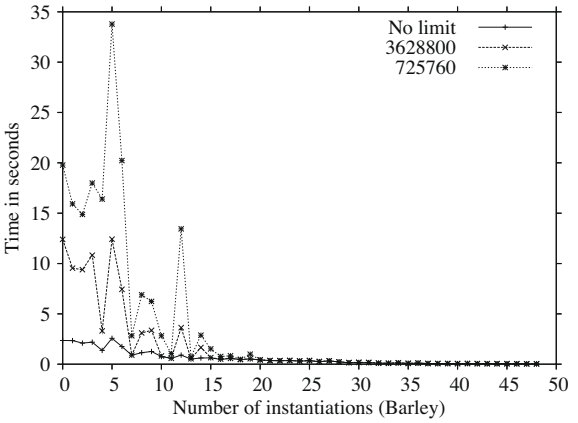

Fig. 20. Time cost for different δ values.

Table 2
Time cost of belief update given two different sets of evidence of size $|\mathcal{X}_\epsilon| = 14$.

Algorithm	δ			
	1,000,000	2,500,000	5,000,000	No limit
AR	333.65	41.84	41.50	3.89
AR & VE	304.68	39.13	39.91	3.78
VE	18.57	12.35	12.27	2.45
	5000	15,000	30,000	No limit
AR	1.79	1.25	1.20	0.51
AR & VE	1.70	1.06	1.00	0.47
VE	1.67	0.92	0.88	0.38

marginals is dominated by the cost of message passing for this network. The largest average potential is rarely reduced as the largest potentials are typically created during message passing and not during computation of marginals.

5.3. 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 $x < \|\mathrm{dom}(\phi)\| \leqslant y$ is created during belief update.

Fig. 20 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 marginalisation algorithm. The time cost increases as δ is reduced.

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.[5] The combined impact of these scenarios increases as δ decreases.

Table 2 shows the time cost for belief update using ARmin, VE and AR & 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.

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 penalise the algorithm under upper-limits constraints in the case of the evidence set represented in the upper part of Table 2. For this evidence set, the time efficiency of AR is reduced by a factor of approximately 85 while the time efficiency of VE is reduced only by a factor of approximately 8.

Fig. 21 shows the time cost of belief update in *Barley* for $\delta = 3,628,800$ using ARmin and ARmax. It is clear from the figure that the order in which AR operations are performed can have a significant impact on the performance of LARP with the any-space property.

[5] 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.

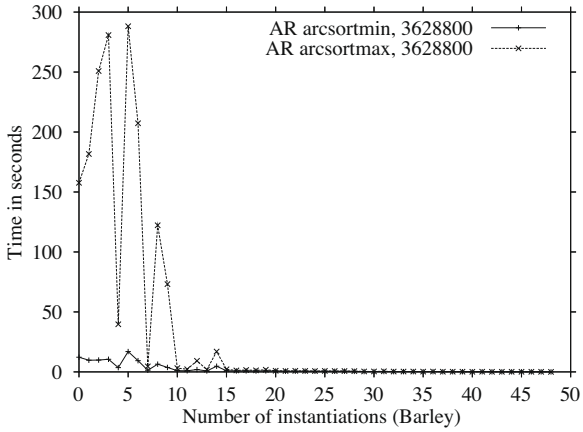

Fig. 21. Time cost for $\delta = 3,628,800$ using arc-reversal sorting.

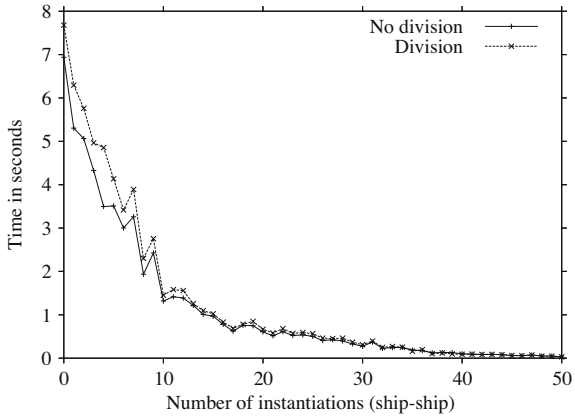

Fig. 22. Time cost with and without the last division.

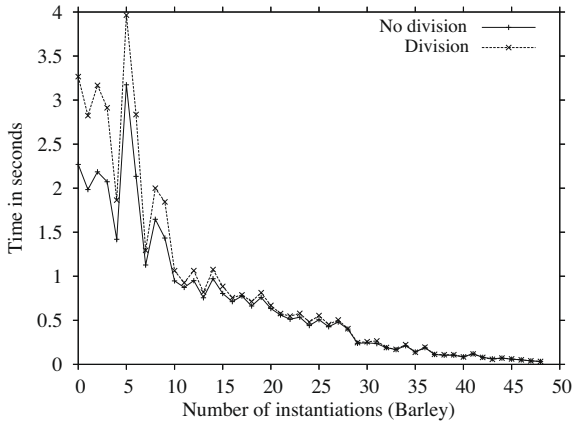

Fig. 23. Time costs with and without the last division.

The any-space scheme makes it possible to adjust δ to the memory that may be available at any given point in time. In the experiments we have assumed the upper limited to be constant. We have not considered the total amount of memory consumed by the belief update operation.

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.

5.4. Division operation

Using AR as marginalisation requires one invocation of Eqs. (1) and (2) for each arc reversed except for the last arc where the invocation of Eq. (2) can be skipped as the variable subsequently will be eliminated as barren [22].

Figs. 22 and 23 illustrate the cost of the division operation in the *ship-ship* and *Barley* network, respectively. 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. In many cases the impact of avoiding the last division is insignificant.

Ndilikilikesha [25] 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 means that barren variable elimination requires marginalisation operations and it therefore becomes a potentially expensive operation.

6. Conclusion

This paper has introduced two improvements to the message and marginal computation in LP: (1) four different score functions for myopically selecting the next edge to reverse when eliminating a variable using AR and (2) an any-space extension of LP. The four different score functions for selecting the next edge to reverse are: *fill-in*, *fill-in-weight*, *number-of-parents* and *cpd-weight*. To assess the importance of the arc-reversal order when eliminating a variable, we have performed a sequence of experiments using eight different heuristic rules for selecting the next edge to reverse. For each score function one heuristic for minimising the score and one heuristic for maximising the score have been considered. The results of the experiments show that the arc-reversal order is important for performance. Future work includes more experiments on the arc-reversal order (e.g., finding the local optimal order and considering arc-reversal orders across multiple variable eliminations).

The paper has introduced an any-space property of LP. A set of experiments has been performed to assess the impact of the any-space property on time cost. The experiments show that the time cost increases as the upper bound on the size of the largest potential is reduced. Future work includes a more in-depth analysis of the any-space potential of LP in message computation and a comparison with other any-space techniques. This includes the option of reconsidering the calculation of a potential at a later point in time. For instance, before accessing the elements of a delayed potential 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.

Acknowledgements

The author is grateful to the anonymous reviewers for their insightful comments and valuable suggestions for improving the quality of this manuscript.

References

[1] F. Bacchus, S. Dalmao, T. Pitassi, Value elimination: Bayesian inference via backtracking search, in: Proc. of UAI, 2003, pp. 20–28.
[2] C.J. Butz, S. Hua, An improved LAZY-AR approach to Bayesian network inference, in: Canadian Conference on AI, 2006, pp. 183–194.
[3] C. Cannings, E.A. Thompson, H.H. Skolnick, Probability functions on complex pedigrees, Advances in Applied Probability 10 (1978) 26–61.
[4] G.F. Cooper, Bayesian Belief-Network Inference Using Recursive Decomposition, Technical Report KSL 90-05, Knowledge Systems Laboratory, 1990.
[5] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artificial Intelligence 42 (2–3) (1990) 393–405.
[6] R.G. Cowell, Local propagation in conditional Gaussian Bayesian networks, Journal of Machine Learning Research 6 (2005) 1517–1550.
[7] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, 1999.
[8] P. Dagum, M. Luby, Approximating probabilistic inference in Bayesian belief networks is NP-hard, Artificial Intelligence 60 (1993) 141–153.
[9] A. Darwiche, Any-space probabilistic inference, in: Proc. of UAI, 2000, pp. 133–142.
[10] A. Darwiche, G. Provan, Query dags: a practical paradigm for implementing belief-network inference, in: JAIR, 1997, pp. 147–176.
[11] R. Dechter, Bucket elimination: a unifying framework for probabilistic inference, in: Proc. of UAI, 1996, pp. 211–219.
[12] R. Dechter, Topological parameters for time–space tradeoff, in: Proc. of UAI, 1996, pp. 220–227.
[13] P.F. Hansen, P.T. Pedersen, Risk Analysis of Conventional and Solo Watch Keeping, Research Report, Department of Naval Architecture and Offshore Engineering, Technical University of Denmark, 1998.
[14] R.A. Howard, J.E. Matheson, Influence diagrams, in: Readings in Decision Analysis, 1984, pp. 763–771 (Chapter 38).
[15] F. Jensen, HUGIN API Reference Manual, 2007. <http://www.hugin.com>.
[16] F.V. Jensen, F. Jensen, Optimal junction trees, in: Proc. of UAI, 1994, pp. 360–366.
[17] F.V. Jensen, S.L. Lauritzen, K.G. Olesen, Bayesian updating in causal probabilistic networks by local computations, Computational Statistics Quarterly 4 (1990) 269–282.
[18] U.B. Kjærulff, Aspects of Efficiency Improvement in Bayesian Networks, Ph.D. Thesis, Department of Computer Science, Aalborg University, Denmark, April 1993.
[19] U.B. Kjærulff, A.L. Madsen, Bayesian Networks and Influence Diagrams – A Guide to Construction and Analysis, Springer-Verlag, 2008.
[20] K. Kristensen, I.A. Rasmussen, The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides, Computers and Electronics in Agriculture 33 (2002) 192–217.
[21] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistical Society B 50 (2) (1988) 157–224.
[22] A.L. Madsen, Variations over the message computation algorithm of lazy propagation, IEEE Transactions on Systems, Man and Cybernetics, Part B 36 (3) (2006) 636–648.
[23] A.L. Madsen, Solving CLQG influence diagrams using arc-reversal operations in a strong junction tree, in: Fourth European Workshop on Probabilistic Graphical Models, 2008, pp. 201–208.

[24] A.L. Madsen, F.V. Jensen, Lazy propagation: a junction tree inference algorithm based on lazy evaluation, Artificial Intelligence 113 (1–2) (1999) 203–245.
[25] P. Ndilikilikesha, Potential influence diagrams, IJAR 11 (1) (1994) 251–285.
[26] S.M. Olmsted, On Representing and Solving Decision Problems, Ph.D. Thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA, 1983.
[27] J. Pearl, Probabilistic Reasoning in Intelligence Systems, Series in Representation and Reasoning, Morgan Kaufman Publishers, 1988.
[28] R.D. Shachter, Evaluating influence diagrams, Operations Research 34 (6) (1986) 871–882.
[29] R.D. Shachter, Probabilistic inference and influence diagrams, Operations Research 36 (4) (1988) 589–604.
[30] R.D. Shachter, B. D'Ambrosio, B. Del Favero, Symbolic probabilistic inference in belief networks, in: Proc. of Eighth National Conference on AI, 1990, pp. 126–131.
[31] G.R. Shafer, P.P. Shenoy, Probability propagation, Annals of Mathematics and Artificial Intelligence 2 (1990) 327–351.
[32] P.P. Shenoy, Binary join trees for computing marginals in the Shenoy–Shafer architecture, IJAR 17 (2–3) (1997) 239–263.
[33] P.P. Shenoy, Inference in hybrid Bayesian networks using mixtures of Gaussians, in: Proc. of UAI, 2006, pp. 428–436.
[34] N.L. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, Journal of Artificial Intelligence Research 5 (1996) 301–328.