New Methods for Marginalization in Lazy Propagation

Anders L. Madsen

Anders.L.Madsen@hugin.com

HUGIN EXPERT A/S

Outline

- Introduction
- Lazy Propagation
- Arc-reversal sort
- Any-space property
- Experiments
- Conclusion

Belief Update By Lazy Propagation

We consider belief update as the task of computing $P(X \mid \epsilon)$ for each $X \in \mathcal{X} \setminus \mathcal{X}_\epsilon$

LP is based on a Shenoy–Shafer message passing scheme in a junction tree representation $\mathcal{T}$ of a Bayesian network $\mathcal{N}$

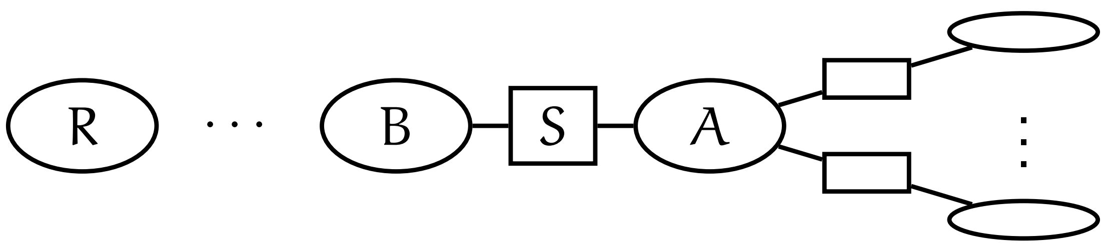
- Exploit hard evidence on discrete random variables $X = x$ to instantiate factors
- Postpone factor combination until mandatory
- Represent a product of factors $\prod \phi_i$ as a set $\{\phi_1, \ldots, \phi_n\}$
- Communicate sets $\{\phi_1, \ldots, \phi_n\}$ between cliques and separators
- Compute messages using *projection* and barren node eliminations

The motivation is to improve performance on complex networks

Lazy Propagation

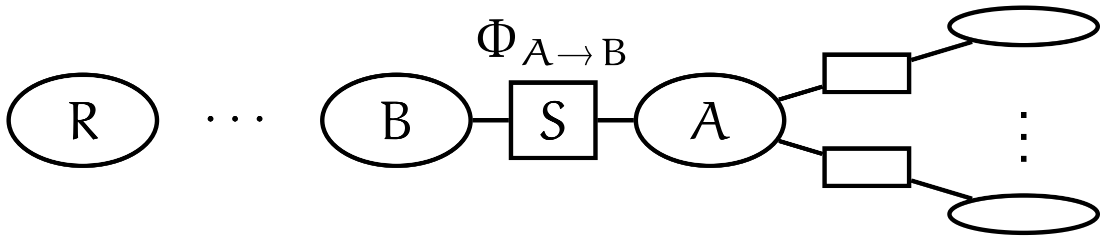
The main steps of Lazy Propagation are

1. Construct a junction tree $\mathcal{T}$ of $\mathcal{N}$
2. Assign $P(X \mid \mathrm{pa}(X))$ to clique C s.t. $\mathrm{fa}(X) \subseteq C$
3. Use hard evidence $X = x$ to instantiate each $\phi \in \mathcal{P}$ s.t. $X \in \mathrm{dom}(\phi)$ for all $X \in \mathcal{X}_\epsilon$
4. After initialization each clique C has been assigned a set Φ_C



$$\Phi_A = \{\phi \in \mathcal{P} \text{ s.t. } \mathrm{dom}(\phi) \subseteq A\}$$

Message Passing



4. Perform collect and distribute relative to root R
   - Messages computed as

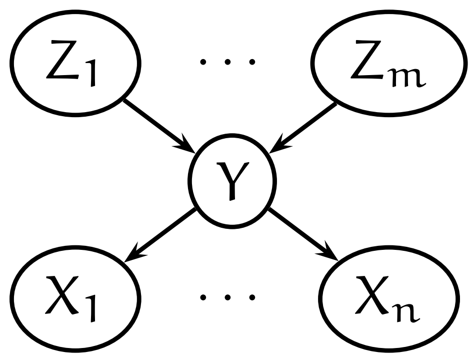
$$\Phi_{A\to B} = \Big(\Phi_A \cup \bigcup_{C\in \mathrm{ne}(A)\setminus\{B\}} \Phi_{C\to A}\Big)^{M\downarrow B}$$

     where M is a marginalization algorithm, e.g., AR or VE
   - A message is a set $\Phi_{A\to B} = \{\phi_1, \ldots, \phi_n\}$ where each ϕ_i is the result of a projection operation on set $\Phi_i \subseteq \Phi_A \cup \bigcup_{C\in \mathrm{ne}(A)\setminus\{B\}} \Phi_{C\to A}$ of source potentials
5. Marginal $P(X)$ is computed from any clique/separator containing X

Arc-Reversal Sort

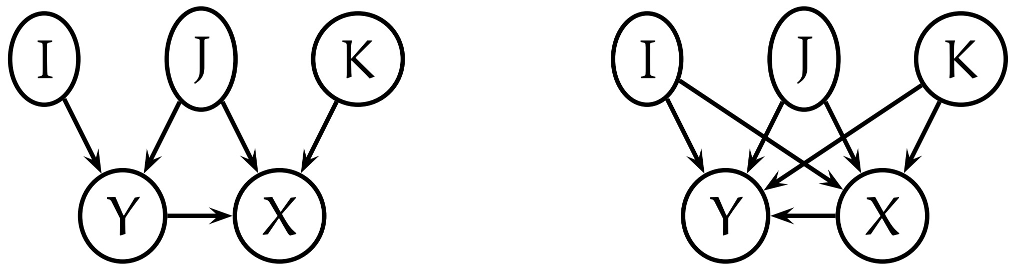
In $\Phi_{A \to B} = \{\phi_1, \ldots, \phi_n\}$ each ϕ_i is the result of series of variable eliminations on a set of source potentials Φ_i

The elimination of a Y by AR requires a sequence ρ of AR operations to make Y barren by reversing arcs



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

Arc-Reversal Sort



$$P(X \mid I, J, K) = \sum_{Y} P(X \mid Y, J, K) P(Y \mid I, J).$$

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

If $|\operatorname{ch}(Y)| > 1$, the set of edges introduced by eliminating Y may depend on the order ρ

Arc-Reversal Sort

To compare order, we define the cost of reversing edge (Y, X) as:

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

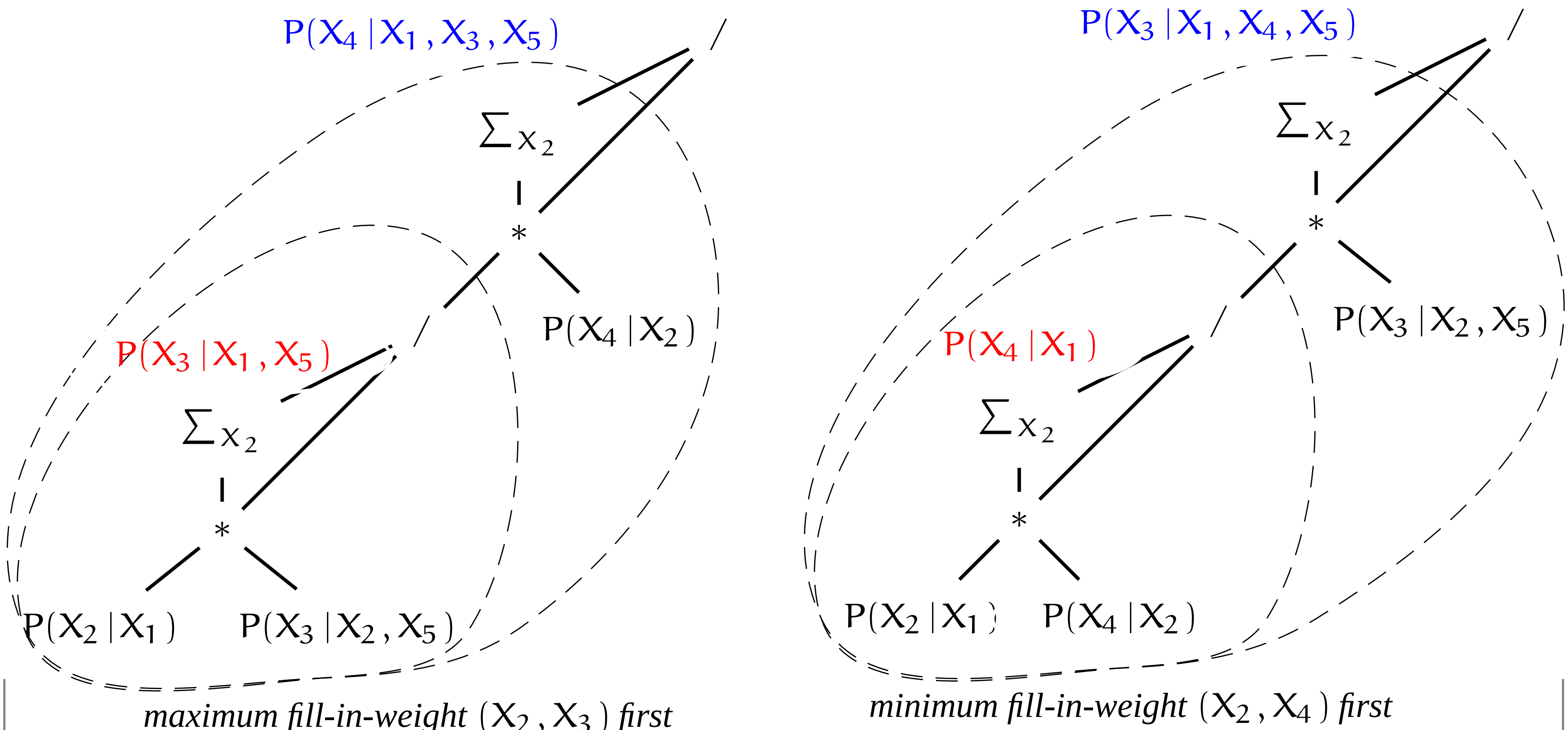
where each Z_X is parent of X, but not of Y and each Z_Y is parent of Y, but not of X. $s(Y, X)$ is the weight of new edges

Finding the optimal order $\hat{\rho}$ is hard

- We consider two different one step look-ahead rules:
  - *maximum fill-in-weight*: select edge with maximum score
  - *minimum fill-in-weight*: select edge with minimum score

Arc-Reversal Sort Example

Consider the example $\Phi = \{P(X_1), (P(X_2 \mid X_1), P(X_3 \mid X_2, X_5), P(X_4 \mid X_2), P(X_5)\}$. Eliminating X_2 using AR involves reversing arcs (X_2, X_3) and (X_2, X_4)



Any-Space

The elimination of X from a set Φ by VE may proceed as

1. Set $\Phi_X = \{\phi \in \Phi | X \in \text{dom}(\phi)\}$
2. Set $\phi_X = \sum_X \prod_{\phi \in \Phi_X} \phi$
3. Set $\Phi^* = \Phi \cup \{\phi_X\} \setminus \Phi_X$

Both ϕ_X and $\prod_{\phi \in \Phi_X} \phi$ may be too large to represent in main memory using a table representation

- An any-space property of LP can trade space for time
- Instead of computing a large potential, we represent and propagate an *expression* (product and sum) for its computation (*delay*)

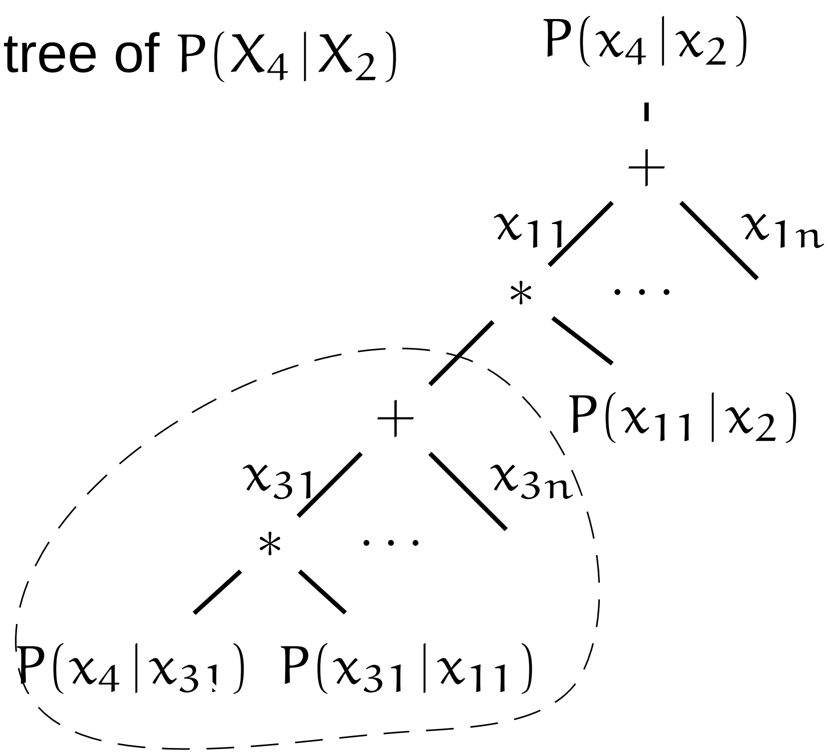
If $||\phi|| > \delta$ where δ is a threshold, we do not compute a table representation of ϕ

Any-Space

The formula for

$$P(X_4 \mid X_2) = \Phi^{VE\downarrow\{X_2, X_4\}} = \sum_{X_1} P(X_1 \mid X_2) \sum_{X_3} P(X_3 \mid X_1) P(X_4 \mid X_3),$$

VE calculation tree of $P(X_4 \mid X_2)$



A calculation is repeated each time an entry is accessed

Performance Analysis

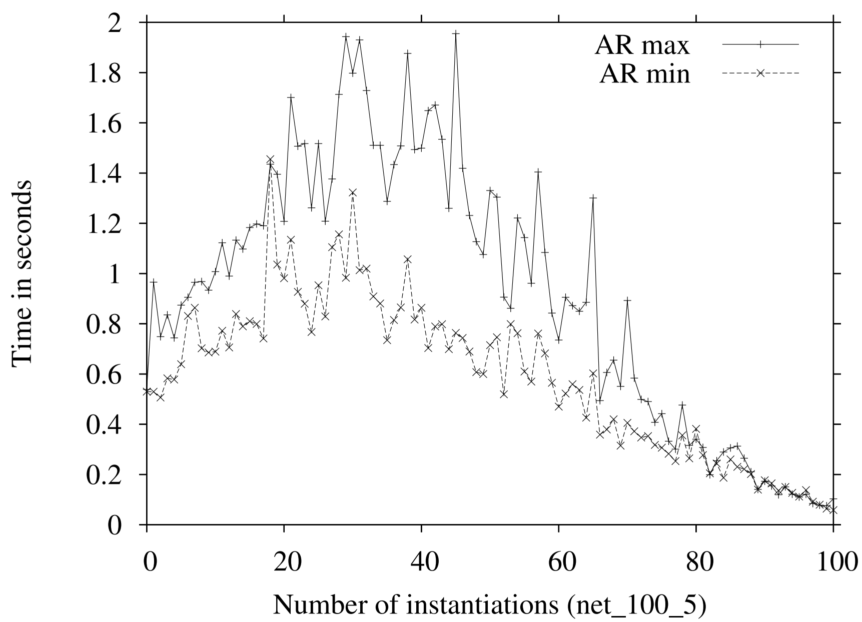
We measure average size of largest factor and average running time

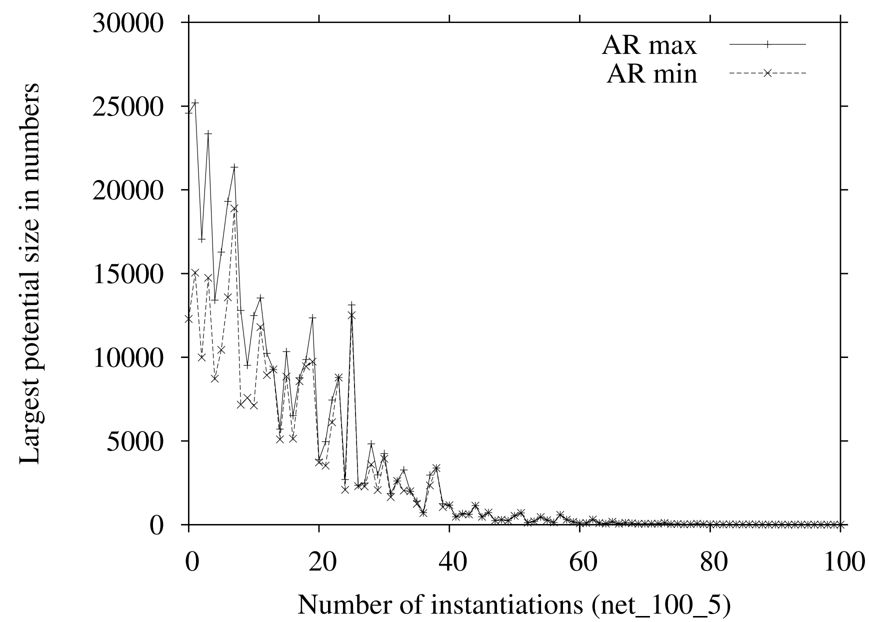
Network	$\|V\|$	$\|\mathcal{C}\|$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$
ship-ship	50	35	$4,032,000$	$24,258,572$
Barley	48	36	$7,257,600$	$17,140,796$
net_100_5	100	85	$98,304$	$311,593$
net_200_5	200	178	$15,925,248$	$70,302,065$

In the table, $s(A) = \prod_{X \in A} \|X\|$ is the state space size of clique $A \in \mathcal{C}$

Arc-Reversal Sort

Time and space cost of belief update in *net_100_5*

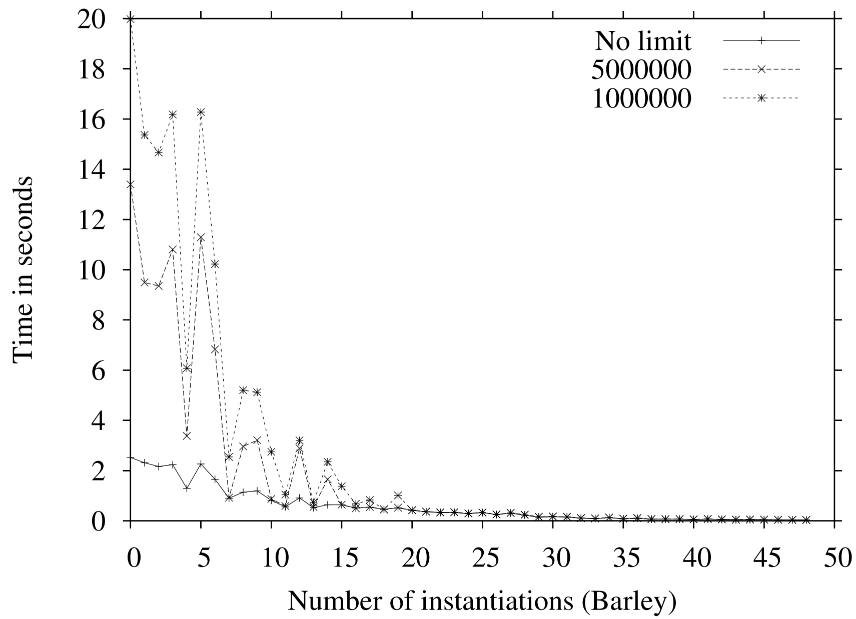




Network	$\|V\|$	$\|\mathcal{C}\|$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$
net_100_5	100	85	98,304	311,593

Any-Space

Time cost of belief update in *Barley* for three different δ values using VE as the marginalization algorithm



The most difficult evidence set contains four instantiations which are inserted into different leaf cliques

Network	$\lvert V \rvert$	$\lvert \mathcal{C} \rvert$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$	$\max_{S \in \mathcal{S}} s(S)$
Barley	48	36	$7,257,600$	$17,140,796$	$907,200$

Any-Space — VE or AR?

HUGIN EXPERT

Time cost for two specific evidence sets in *Barley*

ϵ_1	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
ϵ_2	$5 * 10^3$	$1.5 * 10^4$	$3 * 10^4$	No limit
AR	1.82	1.25	1.25	0.53
VE	2.28	0.96	0.92	0.46

The sizes of the evidence set are $|\mathcal{X}_{\epsilon_1}| = 14 = |\mathcal{X}_{\epsilon_2}| = 14$.

Network	$\lvert V \rvert$	$\lvert \mathcal{C} \rvert$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$	$\max_{S \in \mathcal{S}} s(S)$
Barley	48	36	7,257,600	17,140,796	907,200

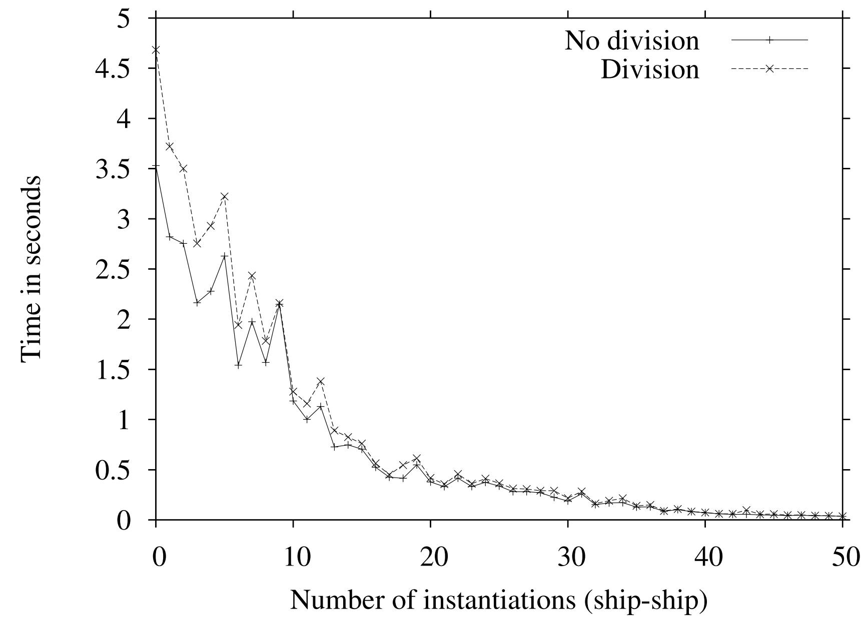
Division Operation

Time cost of belief update in *ship-ship*

An AR operation on arc (Y, X) is performed as follows:

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

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



Network	$\lvert V \rvert$	$\lvert \mathcal{C} \rvert$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$
ship-ship	50	35	4,032,000	24,258,572

Conclusion

The main contributions of the paper are

- introducing LP as a class of algorithms
- sorting arc-reversal operations
- introduced a (simple) any-space scheme for LP

The paper includes a preliminary experimental evaluation of the proposed extensions

Future work

- an in-depth analysis of the any-space potential of LP
- reconsidering the elimination order used in a *delayed* potential
- selection of marginalization operation