A Discussion on LAZY propagation

Y. (Yao) Chen and Y. (Yuping) Liu
June 20, 2017

Abstract

In this paper, we present LAZY propagation for probabilistic inference in Bayesian networks. We first briefly illustrate how LAZY propagation works and how it exploits the independence relations to reduce time and space costs. And then we empirically compare the performance of LAZY propagation architecture with that of HUGIN and Shafer-Shenoy architectures. We found with the increase of the number of instantiated variables, the time and space costs of Lazy propagation decreases. Furthermore, we touched upon the topic of different message computation algorithms, such as VE, AR and SPI. Overall, there is no big difference but relatively, AR produced better results.

1 Introduction

A Bayesian network is an efficient and intuitive graphical model that allows people to represent and reason about an uncertain domain. When a Bayesian network is defined over a complex domain, it may become unfeasible to calculate the posterior marginal distribution for non-evidence variables given some evidence. In fact, exact belief update is an *NP-hard* problem.

Generally, algorithms for probabilistic inference in Bayesian networks are classified into two classes: the class of query-based (direct computation) algorithms and the class of all-marginals (indirect computation) algorithms. The first class of algorithms performs inference based on the structure of the graph of the Bayesian network. This class contains Belief Propagation [1], Arc-Reversal (AR) [2], Symbolic Probabilistic Inference (SPI) [3], Variable Elimination (VE) [4], the Fusion operator [5] , and Value Elimination (VU) [6]. The second class performs probabilistic inference through message passing in a secondary computational structure such as a junction tree and a join tree built by moralisation and triangulation. This class includes Lauritzen-Spiegelhalter [7], HUGIN [8], and Shafer-Shenoy [9].

LAZY propagation combines query-based and all-marginals algorithm. Message passing is performed based on the scheme of Shafer-Shenoy propagation in a junction tree while messages are computed by query-based algorithms using a variable elimination approach [10].

2 LAZY Propagation

LAZY Propagation (Madsen and Jensen 1999) is an inference algorithm, which combines direct and indirect computation for all posterior marginal.

The basic computational structure of the LAZY propagation inference architecture is a junction tree constructed from the Bayesian network under consideration (Madsen & Jensen, 1999). The algorithm is based on message passing. It mainly aims to maintain a multiplicative decomposition of clique and separator potentials and to postpone combination of potentials. In this way, barren variables and independence relations induced by evidence can be exploited.

2.1 Definitions

Definition barren variable

A variable in a Bayesian network is said to be a barren variable if it is not a query variable, does not have evidence and all of its descendants are barren variables.
By the unity-potential axiom, barren variables have no impact on the calculation of the posterior probability distribution of the query variables.

Definition unity-potential axiom

$$\Sigma_H P(H|T) = 1_T$$

H is the head variable or conditioned variable and T is the tail variable or conditioning variable. The unity-potential axiom also applies, if H is a set of variables.

Definition d-separation/d-connection

Variables X and Y in a Bayesian network $N = (G, P)$ are d-separated if for every path connecting X and Y there is an intermediate variable Z such that one of the following statement is true:

- *Z is the middle variable in a serial (Figure 1(a)) or a diverging (Figure 1(c)) connection, and Z is instantiated by evidence.*
- *Z is the middle variable in a converging connection (Figure 1(b)), and neither* Z *nor any of its descendants have received evidence.*

X and Y are d-connected by Z if and only if they are not d-separated by Z.

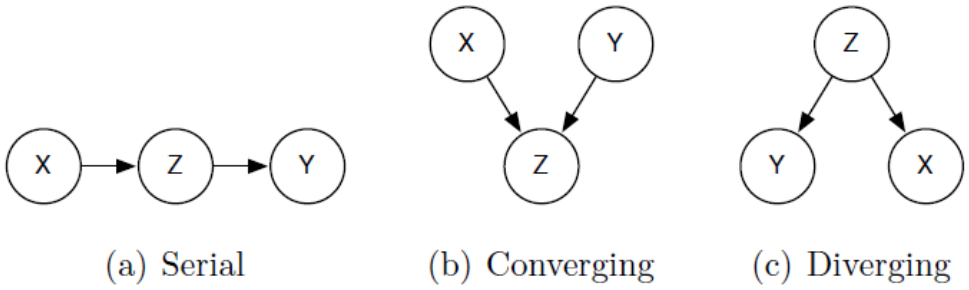

Figure 1 d-separation connection types.

2.2 Message Passing

This process consists of two phases: evidence collection and evidence distribution.

When "collection" is invoked on a clique C_j from an adjacent clique C_i, then C_j invokes "collection" on all other adjacent cliques. When "distribution" is invoked on a clique C_j from a neighbour C_i, C_j absorbs evidence from C_i and invokes distribution on all other adjacent cliques.

Evidence is passed between adjacent cliques by absorption. Figure 1 shows how absorption proceeds. Absorption from clique C_j to clique C_i over separator S amounts to decomposing ϕ_S into potentials associated with C_j and the neighbouring separators except S.

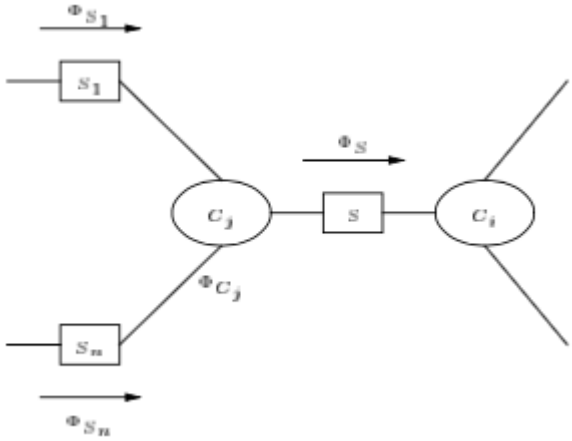

Figure 2: Absorption from clique C_j to clique C_i over separator S

In message passing, we have to avoid the received information passing back to the source clique. That is to say, the information contained in the message from C_j to C_i should not be contained in the message passed in the opposite direction. Regarding this issue, HUGIN, Shafer-Shenoy and LAZY propagation act differently. HUGIN architecture solves this problem by dividing separator potentials, while in the Shafer-Shenoy architecture it is solved by computing the message to pass over S from the potentials associated with C_i and

neighboring separators except S. LAZY propagation avoids this problem by discarding the potentials passed to C_i over S from the set of potentials passed in the opposite direction. Thus, LAZY propagation combines the two methods.

2.3 Evidence

LAZY propagation takes full advantage of the independence relations induced by evidence. It incorporates the evidence by associating evidence on a variable X with all cliques containing X. For example, if the evidence is $X = x$, then the domain of every potential containing X will be reduced so that the representation only includes configurations of the domain where $X = x$ only. Such function is referred to as an *evidence function*.

2.4 Internal Elimination

The computational structure of the LAZY propagation is based on a junction tree. The structure of the junction tree imposes a partial elimination order. The separator S between two adjacent cliques C_i and C_j is the intersection of the two cliques, which indicates the variables to be eliminated when passing messages between C_i and C_j. It is assumed that all potentials R_s associated with C_j and its neighbours except S are relevant for computing ϕ_s^*. However, that is often not the case. LAZY propagation uses a different mechanism to find relevant potentials. First, R_s should be found which contains all potentials d-connected to S in the reduced domain and the next step should be to remove from R_s all potentials containing only barren head variables by the unity-potential axiom.

2.5 Posterior Marginals

In the LAZY propagation, posterior marginals are readily computed. As the clique and separator potentials are factorized multiplicatively, computing a posterior marginal may involve combination of potentials and variable elimination.

Algorithm Posterior Marginal

Let ϕ be the set of potentials representing the joint distribution from which the posterior marginal of Y to be calculated. The posterior marginal P (Y| ε) can be done in the following steps:

1. Find relevant potentials on ϕ to obtain R_Y.
2. For each variable X in $\{X \in dom(\phi) | \ \phi \in \ R_Y, X \neq Y\}$
   (a) Marginalize out X

3. Let ϕ_Y be the set of potentials obtained.

4. Calculate $P\ (Y|\varepsilon) = \frac{\prod_{\phi\in\ \phi_Y}\ \phi}{\sum_Y \prod_{\phi\in\ \phi_Y} \phi}$

2.6 Example

Figure 3 is an example of a junction tree. BCDEF is predetermined to be the root and the given evidence is $\varepsilon = \{D = d\}$. When "evidence collection" is invoked on BCDEF, message will pass from the leaves to BCDEF. The intersection between the BCDEF clique and ABC clique is BC, so A has to be eliminated by marginalization. The same is for the variable G, but no calculations are involved in this step because G is a barren variable.

$$\phi(.|E,F) = \Sigma_G\, P(G|E,F) = 1_{E,F}$$

When "evidence distribution" is invoked on BCDEF, messages will pass from BCDEF to the leaves. The set of potentials associated with BCDEF after evidence collection is $\{\phi(E|d), \phi(F|d), \phi(d|B,C), \phi\ (B,C)\}$. The set of potentials relevant for computing the message to pass to ABC is $\{\ \phi(d|B,C), \phi(B,C)\}$.

Since $\phi(B,C)$ is the same as the message passing in the evidence collection phase and the domain of $\phi(d|B,C)$ is equal to that of the separator, there is no need for computation to obtain the message to ABC.

The set of potentials relevant for computing the message to EFG is $\phi(E|d)$ and $\phi(F|d)$. Again, there is no need for computations as the domains of the relevant potentials are subsets of the separator.

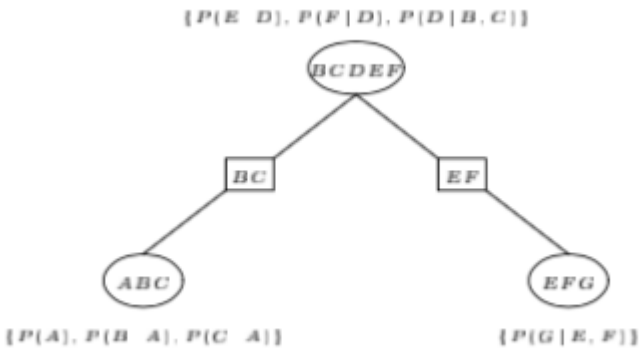

Figure 3: A Junction Tree

The evidence on D has been propagated by performing only one marginalization and two combinations of potentials, which is much more efficient than both HUGIN and Shafer-Shenoy propagation.

For the proof of the correctness of LAZY propagation, please refer to LAZY Propagation: A Junction Tree Inference Algorithm Based on LAZY Evaluation (Madsen & Jensen, 1999).

2.7 Independence of Causal Independence

LAZY Propagation enhances the efficiency by exploiting the independence of causal influence. Different approaches in exploiting independence of causal influence with a junction tree can be considered.

It is simple actually to extend LAZY propagation to take advantage of any of those junction tree inference algorithms, such as factorized representation, LAZY parent divorcing, and heterogeneous factorization. The above-mentioned algorithms offer a decomposition of conditional probability distributions and LAZY propagation exploits a decomposition of clique and separator potentials. It seems straightforward that LAZY propagation can readily be extended to take advantage of the more fine-grained decomposition offered by factorized representation, LAZY parent divorcing, and heterogeneous factorization, respectively. The factorized representation and LAZY parent divorcing do not impose any constraint on the elimination order while heterogeneous factorization does impose.

2.8 Variation

Symbolic Probabilistic Inference (SPI) and Arc-Reversal (AR) can be used for computation of clique-to-clique messages in addition to the traditional use of Variable Elimination (VE).

SPI is a direct computation algorithm, which is fundamentally different from VE. The idea of SPI is to solve a query as a combinatorial optimization problem (Li & D'Ambrosio, 1994). Instead of focusing on the elimination order, SPI focuses on the order in which potentials should be combined.

AR is more fine-grained than both VE and SPI. The basic idea of AR when computing a single marginal is to perform a sequence of arc-reversals and barren variable eliminations.

The results of an empirical evaluation of the performance of LAZY propagation are compared using VE, SPI, and AR as the message computation algorithm. The results of the empirical evaluation show that for most networks, the performance of inference did not depend on the choice of message computation algorithm, but for some randomly generated networks the choice had an impact on both space and time performance. In the cases where the choice had an impact, AR produced the best results.

2.9 Comparison with HUGIN and Shafer-Shenoy

In the HUGIN architecture, each clique sends a message to separators between it and its neighbors. When a separator receives a message from one of its neighboring clique, it sends a message to its other neighbors. When all messages have been sent, the potential at each clique and at each separator is the marginal of the joint for that node. Each clique and each separator in the junction tree stores a potential. Computations are done by each clique and by each separator in the junction tree. The process of belief update is as follow:

1. Calculate the updated separator potential: $\phi_S^* = \sum_{C_j \setminus S} \phi_{C_j}$.
2. Update the clique potential of C_i : $\phi_{C_i}^* = \phi_{C_i} \frac{\phi_S^*}{\phi_S}$.
3. Associate the updated potential with the separator: $\phi_S = \phi_S^*$.

In the Shafer-Shenoy architecture, nodes needed for calculating the marginals request messages from all their neighbors. When a node receives a request for a message, it in turn requests messages from all its other neighbors. When all requested messages have been delivered, the marginals are computed at the desired nodes. A node may store either no potential, or one potential or two or more potentials. Each separator may store one or two potentials. Computations are done only by nodes and not by separators. It can be done as follow:

1. $\phi_S^* = \sum_{C_j \setminus S} \phi_{C_j} \prod_{S' \epsilon ne\,(C_j \{S\}} \phi_{S'}$.
2. $\phi_{C_i}^* = \phi_{C_i} \prod_{S \epsilon ne\{C_i)} \phi_S$.

The LAZY propagation differs from HUGIN and Shafer-Shenoy algorithms in the following manner:

1. Instead of storing only one potential in each clique, it stores a list of potentials (or conditional probability tables). It performs products on some potentials only when necessary, i.e., when they contain a variable that is to be marginalized-out.

2. It recognizes summations like $\Sigma_H P(H|T) = 1T$, the result of which is known for sure to be 1.

3. It uses d-separation to avoid sending a message from one clique to another if the random variables of these cliques are independent due to some evidence.

2.10 Sample Empirical Results

We measured the performance of LAZY propagation relative to Shafer-Shenoy and HUGIN algorithms in terms of time and space cost.

Network	cliques	Clique state space size			Clique neighbours		
		min	max	μ	min	max	μ
Diabetes	337	495	190080	30906.3	1	3	2.0
KK	38	40	5806080	397780.2	1	4	1.9
ship-ship	35	8	4032000	693102.1	1	3	1.9

Table 1: Information on the junction trees

The tests were performed on a number of large Bayesian networks with evidence sets of different sizes.

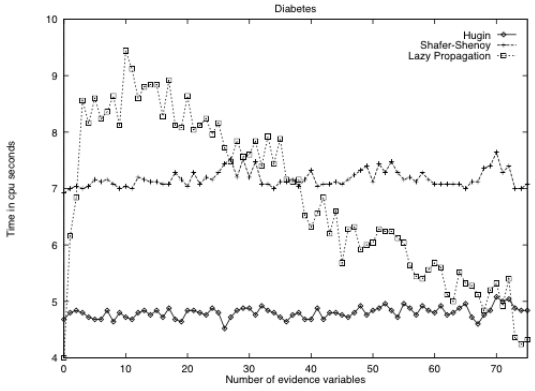

Figure 4: Diabetes

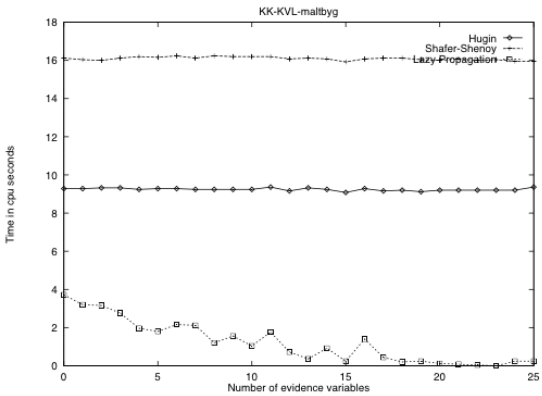

Figure 5: KK

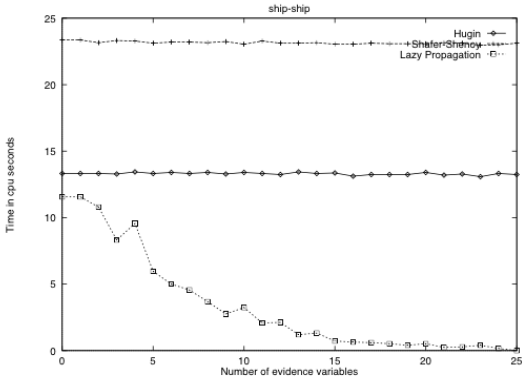

Figure 6: ship-ship

Figure 4, 5, 6 show plots of the average time for propagation of evidence for the Diabetes, KK-KVL-maltbyg (KK), and ship-ship networks, respectively.

The plots clearly show that with the increase of the number of instantiated variables, the time cost of LAZY propagation will decrease. Figure 5 and 6 show LAZY propagation costs less time than the other algorithms even when no variables are instantiated. Figure 4 shows the time cost of LAZY propagation is higher than that of HUGIN and Shafer-Shenoy for the small set of instantiated variables, but as the number of instantiated variables reached 10, the time cost of LAZY propagation decreases.

2.11 Various Speed-up Improvements

LAZY propagation implements a speed-up improvement that is referred to as expression trees and is related to the use of the unity-potential axiom. Expression trees are used in order to recognize situations where elimination of a set of variables will lead to a unity potential. For example, for the junction tree in figure 7, it should be recognized that the elimination of variables A and B from the potential $\phi\ (A, B|D)$ when calculating the message to pass from ABDE to EF produces a unity potential.

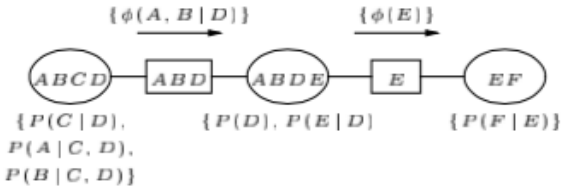

Figure 7

A number of different improvements to standard junction tree propagation architectures have been presented through the years, such as zero compression, binary join trees and nested junction trees.

3 Conclusion

We have presented how the LAZY propagation works and compared the performance of LAZY propagation, HUGIN and Shafer-Shenoy architectures. Compared to HUGIN and Shafer-Shenoy architectures, LAZY propagation is more efficient in aspects of time and space. Although the implementations of the HUGIN and Shafer-Shenoy architectures are simple, LAZY propagation would be useful in some cases where the efficiency of belief update may be insufficient using other algorithms, such as HUGIN and Shafer-Shenoy algorithms.

The LAZY propagation architecture outperforms HUGIN and Shafer-Shenoy algorithms when dealing with Bayesian networks in terms of time and space cost. In this way, we can say LAZY propagation enlarges the class of tractable Bayesian networks.

Bayesian networks are not restricted to the variables of either discrete or continuous. Instead, it can be mixed. In recent years, Bayesian networks with a mixture of continuous and discrete variables also have received an increasing level of attention. For example, it has been applied to the restricted class of mixture Bayesian networks known as conditional linear Gaussian Bayesian networks (CLG Bayesian networks). LAZY propagation works very well for this type of Bayesian networks too.

References

[1] J. Pearl, Probabilistic Reasoning in Intelligence Systems, Series in Representation and Reasoning, *Morgan Kaufman Publishers*, 1988.

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

[3] 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.

[4] N.L. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, *Journal of Artificial Intelligence Research*, 5 (1996) 301–328.

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

[6] F. Bacchus, S. Dalmao, T. Pitassi, Value elimination: Bayesian inference via backtracking search, *in: Proc. of UAI*, 2003, pp. 20–28.

[7] 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.

[8] 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.

[9] G.R. Shafer, P.P. Shenoy, Probability propagation, *Annals of Mathematics and Artificial Intelligence*, 2 (1990) 327–351.

[10] Li, Z. and D'Ambrosio, B. Efficient Inference in Bayes Networks As A Combinatorial Optimization Problem. *International Journal of Approximate Reasoning*, 11, (1) (1994), 55–81.

[11] Madsen A L, F.V. Jensen, LAZY Propagation: A Junction Tree Inference Algorithm Based on Lazy Evaluation*, Artificial Intelligence*, 113, (1-2), 203–45.

[12] Madsen A L. An empirical evaluation of possible variations of LAZY propagation, *Uncertainty in Artificial Intelligence*, 2004: 366-373.