Bayesian Network Inference Using Marginal Trees

Cory J. Butz[1⋆], Jhonatan de S. Oliveira[2 ⋆⋆], and Anders L. Madsen[3 4]

[1] University of Regina, Department of Computer Science,
Regina, S4S 0A2, Canada
butz@cs.uregina.ca
[2] Federal University of Viçosa, Electrical Engineering Department,
Viçosa, 36570-000, Brazil
jhonatan.oliveira@gmail.com
[3] HUGIN EXPERT A/S, Aalborg, Denmark
Aalborg, DK-9000, Denmark
anders@hugin.com
[4] Aalborg University, Department of Computer Science,
Aalborg, DK-9000, Denmark

Abstract. *Variable Elimination* (VE) answers a query posed to a *Bayesian network* (BN) by manipulating the conditional probability tables of the BN. Each successive query is answered in the same manner. In this paper, we present an inference algorithm that is aimed at maximizing the reuse of past computation but does not involve precomputation. Compared to VE and a variant of VE incorporating precomputation, our approach fairs favourably in preliminary experimental results.

Keywords: Bayesian network, inference, marginal trees

1 Introduction

Koller and Friedman [1] introduce readers to inference in *Bayesian networks* (BNs) [2] using the *Variable Elimination* (VE) [3] algorithm. The main step of VE is to iteratively eliminate all variables in the BN that are not mentioned in the query. Subsequent queries are also answered against the BN meaning that past computation is not reused. The consequence is that some computation may be repeated when answering a subsequent query.

Cozman [4] proposed a novel method attempting to reuse VE's past computation when answering subsequent queries. Besides the computation performed by VE to answer a given query, Cozman's method also performs precomputation that may be useful to answer subsequent queries. While Cozman's approach is meritorious in that it reduces VE's duplicate computation, one undesirable feature is that precomputation can build tables that are never used.

⋆ Supported by NSERC Discovery Grant 238880.
⋆⋆ Supported by CNPq - Science Without Borders.

In this paper, we introduce *marginal tree inference* (MTI) as a new exact inference algorithm in discrete BNs. MTI answers the first query the same way as VE does. MTI answers each subsequent query in a two-step procedure that can readily be performed in a new secondary structure, called a *marginal tree*. First, determine whether any computation can be reused. Second, only compute what is missing to answer the query. One salient feature of MTI is that it does not involve precomputation, meaning that every probability table built is necessarily used in answering a query. In preliminary experimental results, MTI fairs favourably when compared to VE and Cozman [4].

The remainder is organized as follows. Section 2 contains definitions. Marginal trees are introduced in Section 3. Section 4 presents MTI. Related work is discussed in Section 5. Section 6 describes advantages and give preliminary experimentals results. Conclusions are given in Section 7.

2 Definitions

2.1 Bayesian Network

Let U be a finite set of variables. Each variable $v_i \in U$ has a finite domain, denoted $dom(v_i)$. A *Bayesian network* (BN) [2] on U is a pair (B, C). B is a *directed acyclic graph* (DAG) with vertex set U and C is a set of *conditional probability tables* (CPTs) $\{p(v_i|P(v_i)) \mid v_i \in U\}$, where $P(v_i)$ denotes the parents (immediate predecessors) of $v_i \in B$. For example, Fig. 1 shows the *extended student Bayesian network* (ESBN) [1], where CPTs are not shown. The product of the CPTs in C is a joint probability distribution $p(U)$. For $X \subseteq U$, the *marginal distribution* $p(X)$ is $\sum_{U-X} p(U)$. Each element $x \in dom(X)$ is called a *row* (configuration) of X. Moreover, $X \cup Y$ may be written as XY. We call B a BN, if no confusion arises.

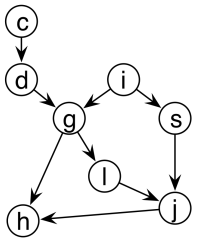

Fig. 1. The DAG of the ESBN [1].

2.2 Variable Elimination

Variable elimination (VE) [3] computes $p(X|E = e)$, where X and E are disjoint subsets of U, and E is observed taking value e. In VE (Algorithm 1), Φ is the

set of CPTs for B, X is a list of query variables, E is a list of observed variables, e is the corresponding list of observed values, and σ is an elimination ordering for variables $U - (X \cup E)$. Evidence may not be denoted for simplified notation.

Algorithm 1. VE(Φ, X, E, e, σ)
Delete rows disagreeing with $E = e$ from $\phi \in \Phi$
While σ is not empty:
  Remove the first variable v from σ
  Φ = sum-out(v, Φ)
$p(X, E = e) = \prod_{\phi \in \Phi} \phi$
return $p(X, E = e) / \sum_X p(X, E = e)$

The sum-out algorithm eliminates v from a set Φ of *potentials* [1] by multiplying together all potentials involving v, then summing v out of the product.

Example 1. [1] Suppose $p(j|h = 0, i = 1)$ is the query issued to the ESBN in Fig. 1. One possible elimination order is $\sigma = (c, d, l, s, g)$. The evidence $h = 0$ and $i = 1$ are incorporated into $p(h|g, j)$, $p(i)$, $p(g|d, i)$, and $p(s|i)$. VE then computes:

$$p(d) = \sum_c p(c) \cdot p(d|c), \tag{1}$$

$$p(g|i) = \sum_d p(d) \cdot p(g|d, i), \tag{2}$$

$$p(j|g, s) = \sum_l p(l|g) \cdot p(j|l, s), \tag{3}$$

$$p(j|g, i) = \sum_s p(s|i) \cdot p(j|g, s), \tag{4}$$

$$p(h, j|i) = \sum_g p(g|i) \cdot p(h|g, j) \cdot p(j|g, i). \tag{5}$$

Next, the product of the remaining potentials is taken, $p(h, i, j) = p(h, j|i) \cdot p(i)$. VE answers the query by normalizing on the evidence variables, $p(j|h, i) = p(h, i, j) / \sum_j p(h, i, j)$.

3 Marginal Trees

We begin by motivating the introduction of marginal trees.

Example 2. Suppose $p(s|h = 0, i = 1)$ is the second query issued to VE. One possible elimination order is $\sigma = (c, d, l, j, g)$. VE performs:

$$p(d) = \sum_c p(c) \cdot p(d|c), \tag{6}$$

$$p(g|i) = \sum_d p(d) \cdot p(g|d, i), \tag{7}$$

$$
\begin{aligned}
p(j|g,s) &= \sum_{l} p(l|g) \cdot p(j|l,s), \qquad (8)\\
p(h|g,s) &= \sum_{j} p(j|g,s) \cdot p(h|g,j),\\
p(h|i,s) &= \sum_{g} p(h|g,s) \cdot p(g|i),\\
p(s,h,i) &= p(s|i) \cdot p(i) \cdot p(h|i,s),\\
p(s|h,i) &= p(s,h,i) / \sum_{s} p(s,h,i).
\end{aligned}
$$

Note that VE's computation in (1)-(3) for the first query is repeated in (6)-(8) for the second query. We seek to avoid recomputation.

The second query $p(s|h = 0, i = 1)$ can be answered from the following factorization of the marginal $p(h, i, s)$:

$$p(h,i,s) = p(s|i) \cdot p(i) \cdot \sum_{g} p(g|i) \cdot \sum_{j} p(j|g,s) \cdot p(h|g,j),$$

where the past calculation of $p(g|i)$ in (2) and $p(j|g,s)$ in (3) are reused.

We introduce marginal trees as a representation of past computation. This secondary structure not only facilitates the identification of that computation which can be reused, but also enables the determination of what missing information needs to be constructed. It is based on the fact that VE can be seen as one-way propagation in a join tree [5]. A *join tree* [5] is a tree having sets of variables as nodes with the property that any variable in two nodes is also in any node on the path between the two.

Definition 1. *Given a Bayesian network B defining a joint probability distribution $p(U)$. A marginal tree M is a join tree on $X \subseteq U$ with CPTs of B assigned to nodes of M and showing constructed messages in one-way propagation to a chosen root node R of M yielding $p(R)$.*

The initial marginal tree has one node N, which has all variables from U. All the CPTs from the BN are assigned to N. For example, the initial marginal tree for the ESBN is depicted in Fig. 2 (i).

Each time a variable v is eliminated, a new marginal tree is uniquely formed by replacing one node with two nodes and the CPT (containing only those rows agreeing with the evidence) built during elimination is passed from one new node to the other. We say that a CPT from N_1 to N_2 is *outgoing* from N_1 and *incoming* to N_2.

Whenever v is eliminated, there exists a unique node N containing v without outgoing messages. Let Γ be the set of all assigned CPTs and incoming messages to N, Ψ is the set of all CPTs containing v, and τ the CPT produced by summing out v. Replace N by nodes N_1 and N_2. N_1 has Ψ assigned CPTs and N_2 has $\Gamma - \Psi$ assigned CPTs. The variables in nodes N_1 and N_2 are defined by the variables

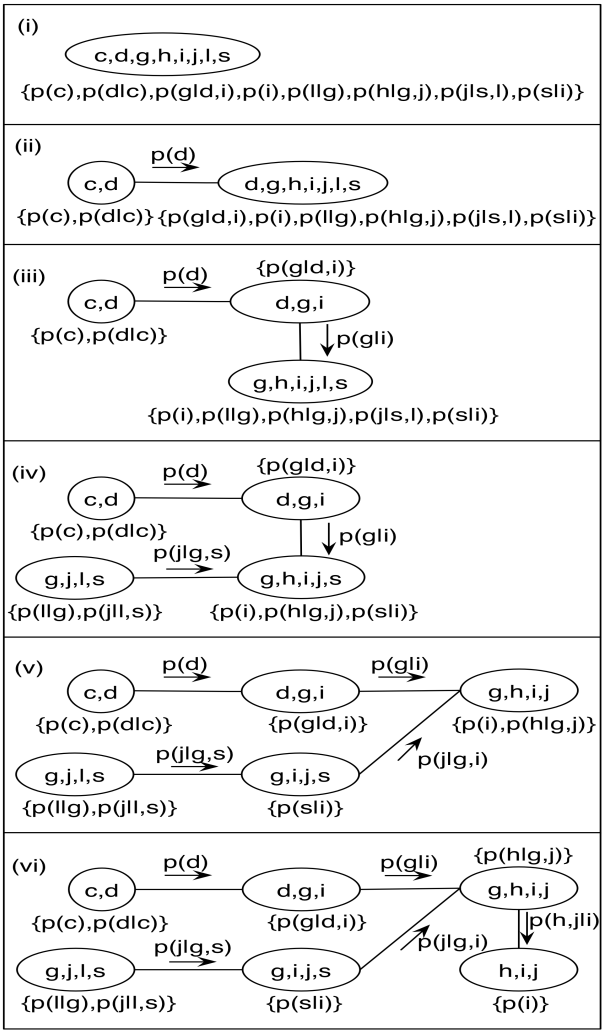

Fig. 2. In Example 3, the initial marginal tree is in (i). The respective marginal trees (ii)-(vi) formed by eliminating c,d,l,s and g are in (1)-(5), respectively.

appearing in the CPTs assigned to N_1 and N_2, respectively. For each incoming message m from node N_i to N, if $m \in \Psi$, set m as incoming message from N_i to N_1; otherwise, set m as incoming message from N_i to N_2. The outgoing message from N_1 to N_2 is τ.

Example 3. In Example 1, the initial marginal tree is shown in Fig. 2 (i). Here, $\Gamma = \{p(c),\ p(d|c),\ p(g|d,i),\ p(i),\ p(l|g),\ p(h|g,j),\ p(j|s,l),\ p(s|i)\}$. The CPTs needed to eliminate variable c are $\Psi = \{p(c), p(d|c)\}$. A new marginal tree is

uniquely formed as shown in Fig. 2 (ii). Node $N = \{c, d, g, h, i, j, l, s\}$ is replaced by N_1 with assigned CPTs Ψ and N_2 with $\Gamma - \Psi$. Thus, $N_1 = \{c, d\}$ and $N_2 = \{d, i, g, s, l, h, j\}$. As seen in (1), $\tau = p(d)$ is the outgoing message from N_1 to N_2. The new marginal tree with N_1 and N_2 can be seen in Fig. (2) (ii).

The subsequent elimination of variable d yields the marginal tree in Fig. (2) (iii). The unique node containing d and having no outgoing messages is $N = \{d, g, h, i, j, l, s\}$. All assigned CPTs and incoming messages to N are $\Gamma = \{p(d), p(g|d, i), p(i), p(l|g), p(h|g, j), p(j|s, l), p(s|i)\}$. The CPTs used to eliminate variable d are $\Psi = \{p(d), p(g|d, i)\}$. N is replaced by N_1 with assigned CPTs Ψ and N_2 with $\Gamma - \Psi$. Then, $N_1 = \{d, g, i\}$ and $N_2 = \{g, h, i, j, l, s\}$. There is one incoming message $p(d)$ from $N_i = \{c, d\}$ to N. Since $p(d) \in \Psi$, N_1 has $p(d)$ as an incoming message. When summing out d, $\tau = p(g|i)$ is the outgoing message from N_1 to N_2. The new marginal tree with N_1 and N_2 is Fig. (2) (iii).

It can be verified that the elimination of variables l, s and g yield the marginal trees shown in Fig. 2 (iv), (v) and (vi), respectively.

Observe in Example 3 that past computation is saved in marginal trees. For example, the computation to answer query $p(j|h = 0, i = 1)$ is saved in the marginal tree of Fig. 2 (vi). Instead of processing a new query against the given BN, we present in the next section a method for reusing computation saved in a marginal tree.

4 Marginal Tree Inference

There are two general steps needed when answering a subsequent query. First, determine which of the past computation can be reused. Second, compute what is missing (in addition to the reused computation) to answer the query. In our marginal tree representation, the former step requires modification of a marginal tree, while the latter boils down to computing missing messages in the modified marginal tree.

4.1 Determine Reusable Computation

Let the new query be $p(X|E = e)$. We select nodes with reusable computation in the marginal tree M with respect to the new query using the *selective reduction algorithm* (SRA) [6], described as follows. Mark variables XE in a copy M' of M. Repeatedly apply the following two operations until neither can be applied: (i) delete an unmarked variable that occurs in only one node; (ii) delete a node contained by another one.

Example 4. Let M be the marginal tree in Fig. 2 (vi) and M' be the copy in Fig. 3 (i). Let $N_1 = \{c, d\}$, $N_2 = \{d, g, i\}$, $N_3 = \{g, j, l, s\}$, $N_4 = \{g, i, j, s\}$, $N_5 = \{g, h, i, j\}$ and $N_6 = \{h, i, j\}$. Let the new query be $p(s|h = 0, i = 1)$. Variables s, h and i are first marked. Variable c can be deleted as it occurs only in N_1. Therefore, $N_1 = \{d\}$. Now N_1 can be deleted, since $N_1 \subseteq N_2$. It can be verified that after applying steps (i) and (ii) repeatedly, all that remains is $N_4 = \{g, i, j, s\}$ and $N_5 = \{g, h, i, j\}$, as highlighted in Fig. 3 (ii).

The SRA output is the portion of VE's past computation that can be reused. For instance, Example 4 indicates that the computation (1)-(3) for answering query $p(j|h = 0, i = 1)$ can be reused when subsequently answering the new query $p(s|h = 0, i = 1)$.

Now, we need to construct a marginal tree $M^{'}$ to be used to answer the new query $p(X|E = e)$ while at the same time reusing past computation saved in M.

Algorithm 3. Rebuild(M, XE)
Let $M^{'}$ be a copy of M
$M^{''}$ = SRA($M^{'}$, XE)
N = $\cup_{N_i \in M''} N_i$
Delete all messages between $N_i \in M^{''}$ from $M^{'}$
Delete all nodes $N_i \in M^{''}$ from $M^{'}$
Adjust messages as incoming to N accordingly
return $M^{'}$

Example 5. Supposing that the new query is $p(s|h = 0, i = 1)$, then we call Rebuild(M,$\{s, h, i\}$), where M is the marginal tree in Fig. 2 (vi). Let $M^{'}$ be the copy in Fig. 3 (i) and $M^{''} = \{N_4 = \{g, i, j, s\}, N_5 = \{g, h, i, j\}\}$. In Rebuild, the next step sets $N = N_4 \cup N_5 = \{g, h, i, j, s\}$.

The message $p(j|g, i)$ from N_4 to N_5 in Fig. 3 (i) is ignored in Fig. 3 (iii). Moreover, all incoming messages to N_4 and N_5 remain as incoming messages to the new node N. The output from Rebuild is the modified marginal tree depicted in Fig. 3 (iv).

The key point is that the modified marginal tree has a node N containing all variables in the new query. Thus, the query could be answered by one-way probability propagation towards N in the modified marginal tree. For example, query $p(s|h = 0, i = 1)$ can be answered in the modified marginal tree in Fig. 4 (i) by propagating towards node $N = \{g, h, i, j, s\}$. However, some computation can be reused when answering a new query. For example, it can be seen in Fig. 4 (ii) that messages $p(d)$, $p(g|i)$ and $p(j|g, s)$ have already been built in (1)-(3) and do not have to be recomputed.

4.2 Determine Missing Computation

Given a marginal tree M constructed for a query and a new query $p(X|E = e)$ such that XE is contained within at least one node $N \in M$, the *partial-one-way-propagation* (POWP) algorithm determines which messages of M can be reused when answering $p(X|E = e)$, namely, it determines what missing computation is needed to answer $p(X|E = e)$. POWP works by determining the messages needed for one-way propagation to root node N in M [7, 8], and then ignoring those messages that can be reused.

Example 6. Fig. 5 illustrates how POWP determines missing information to answer a new query $p(s|l = 0)$ from the marginal tree M in (i) previously built when answering query $p(j|h = 0, i = 1)$. In (ii), we consider node $\{g, j, l, s\}$ as

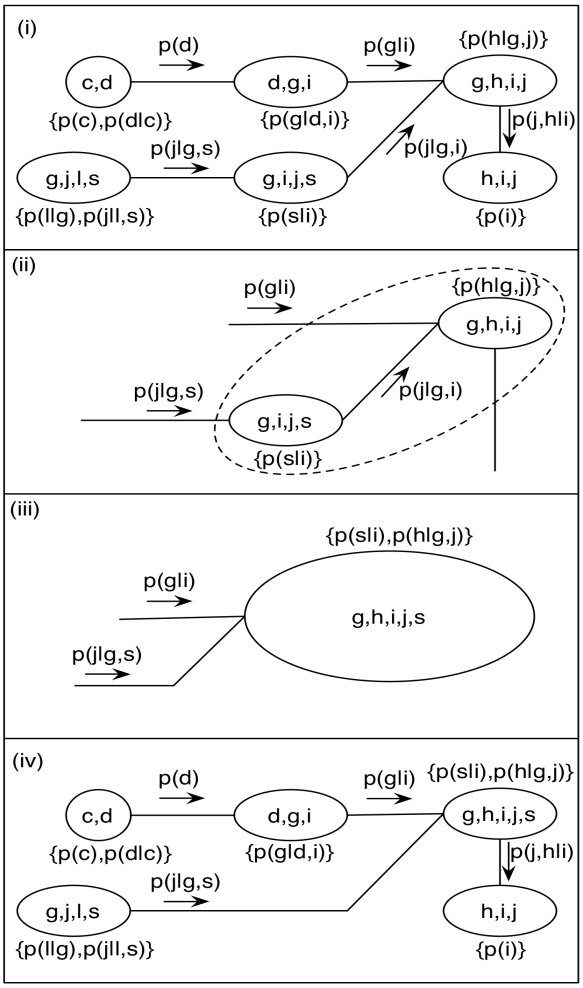

Fig. 3. (i) the marginal tree for Example 4. (ii) RSA outputs $N_4 = \{g, i, j, s\}$ and $N_5 = \{g, h, i, j\}$. (iii) $N = \{g, i, j, s\} \cup \{g, h, i, j\}$. (iv) the modified marginal tree built by RSA.

the new root node N, since the query variables s and l are contained in N. One-way propagation towards N is shown in Fig. 5 (ii). POWP determines that messages $p(d)$ and $p(g|i)$ can be reused from (1)-(2), and are hence ignored as shown in Fig. 5 (iii).

The important point is that the new query can be answered at the root node when POWP finishes.

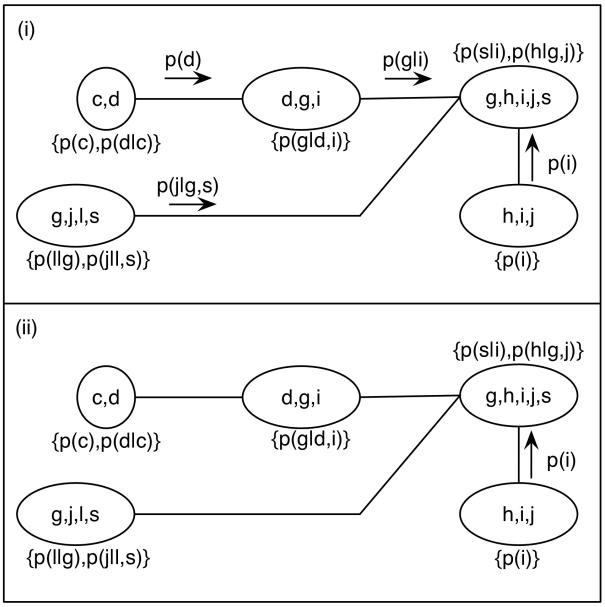

Fig. 4. (i) new messages needed to build a marginal factorization on new root $N = \{g, h, i, j, s\}$. (ii) messages $p(d)$, $p(g|i)$ and $p(j|g, s)$ built when answering a previous query in (1)-(3) can be reused.

Example 7. In Fig. 5 (iii), once POWP to the root node $N = \{g, j, l, s\}$ completes, $p(s|l = 0)$ can be answered from the marginal factorization of $p(N)$: $p(g, j, l, s) = p(l|g) \cdot p(j|l, s) \cdot p(g, s)$.

As another example, given the initial query $p(j|h = 0, i = 1)$ and marginal tree M in Fig. 3 (i), the following takes place to answer a new query $p(s|h = 0, i = 1)$. The modified marginal tree $M^{'}$ is depicted in Fig. 4 (i). The POWP algorithm determines the messages to propagate to the root node $N = \{g, h, i, j, s\}$, chosen as root since it contains the variables in the new query $p(s|h = 0, i = 1)$. POWP determines that messages $p(d)$, $p(g|i)$, and $p(j|g, s)$ in Fig. 3 (i) can be reused in Fig. 4 (i). Thus, only message $p(i)$ needs to be computed in Fig. 4 (ii). And, lastly, the query $p(s|h = 0, i = 1)$ can be answered from the marginal factorization of $p(N)$:

$$p(g, h, i, j, s) = p(g|i) \cdot p(j|g, s) \cdot p(i) \cdot p(s|i) \cdot p(h|g, j).$$

The culmination of the ideas put forth thus far is formalized as the *marginal tree inference* (MTI) algorithm.

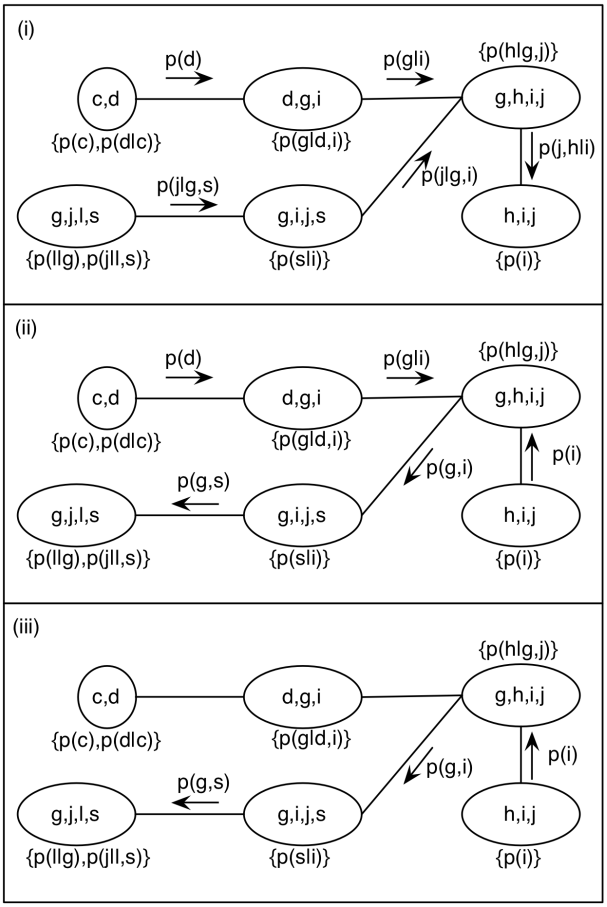

Fig. 5. (i) messages built answering the initial query $p(j|h=0,i=1)$ in node $\{h,i,j\}$. (ii) new root $N=\{g,j,l,s\}$. (iii) messages $p(d)$ and $p(g|i)$ can be reused while messages $p(g,s)$, $p(g,i)$ and $p(i)$ need to be built.

Algorithm 4. MTI(M, $p(X|E=e)$)
$M^{'}$ = SRA(M, XE)
Rebuild($M^{'}$, XE)
POWP($M^{'}$, XE)
Compute $p(X|E=e)$ at the root node of $M^{'}$
Return $p(X|E=e)$

MTI takes as input a marginal tree M built from a previous query and a new query $p(X|E=e)$.

For instance, answering $p(j|h=0,i=1)$ in Example 1 builds the marginal tree M in Fig. 2 (vi), as described in Example 3. Let $p(s|l=0)$ be the new query.

SRA outputs two nodes highlighted in Fig. 3 (ii), as illustrated in Example 4. In Example 5, Rebuild constructs $M^{'}$, as depicted in Fig. 3 (iv). Next, POWP determines reusable messages in Example 6, as illustrated in Fig. 4. Finally, $p(s|l = 0)$ can be calculated at the new root $N = \{g, j, l, s\}$, as discussed in Example 7.

5 VE with Precomputation

Cozman [4] gives a novel algorithm in an attempt to reuse VE's past computation. The crux of his algorithm can be understood as follows. As previously mentioned, Shafer [5] has shown that VE's computation to answer a query can be seen as one-way propagation to a root node in a join tree. It is also known [5] that conducting a subsequent outward pass from the root node to the leaves of the join tree, called two-way propagation, builds marginals $p(N)$ for every node N in the join tree. Cozman's method is to perform the outward second pass after VE has conducted the inward pass for a given query. The marginals built during the outward pass may be useful when answering a subsequent query.

As shown in [4], the VE algorithm can be described as follows. Given a query $p(X|E)$:

1. Compute the set of non-observed and non-query variables $N = U - XE$.
2. For each $v_i \in N$
   (a) Create a data structure B_i, called a *bucket*, containing:
      – the variable v_i, called the *bucket variable*;
      – all potentials that contain the bucket variable, called *bucket potentials*;
   (b) Multiply the potentials in B_i. Store the resulting potential in B_i; the constructed potential is called B_i*'s cluster*.
   (c) Sum out v_i from B_i's cluster. Store the resulting potential in B_i; this potential is called B_i*'s separator*.
3. Collect the potentials that contain the query variables in a bucket B_q. Multiply the potentials in B_q together and normalize the result.

Denote the bucket variable for B_i as v_i, the variables in B_i's separator by S_i, and the evidence contained in the sub-tree above and including bucket B_i by E_i. The outward pass is given by computing the marginal probability for every variable in a BN. In order to update buckets immediately above the root, denoted B_a, with the normalized potential containing v_a and some of the variables in X, compute:

$$p(v_a|E) \;=\; \sum_{X} p(v_a|X, E_a) \cdot p(X|E). \tag{9}$$

Similarly, to update the buckets away from the root, say B_b, compute:

$$p(v_b|E) \;=\; \sum_{S_b} p(v_b|S_b, E_b) \cdot p(S_b|E). \tag{10}$$

Example 8. Given query $p(j|h=0,i=1)$, Cozman's method runs VE to answer it. This inward pass generates a tree of buckets as shown in Fig. 6 (i). Next, Cozman performs an outward pass with the following computation. In order to use (9), first we need to compute $p(v_a|X,E_a)$:

$$p(g|h,i,j) = p(g,h,j|i)/\sum_g p(g,h,j|i). \tag{11}$$

Now apply (9) to determine:

$$p(g|h,i) = \sum_j p(g|j,h,i)\cdot p(j|h,i). \tag{12}$$

Similarly, in order to use (10) we first compute $p(v_b|S_b,E_b)$:

$$p(d|g,i) = p(d,g|i)/\sum_d p(d,g|i). \tag{13}$$

Now apply (10) to compute:

$$p(d|h,i) = \sum_g p(d,g|i)\cdot p(g|h,i). \tag{14}$$

The remainder of the example is as follows:

$$p(c|d) = p(c,d)/\sum_c p(c,d), \tag{15}$$

$$p(c|h,i) = \sum_d p(c|d)\cdot p(d|h,i), \tag{16}$$

$$p(s|g,i,j) = p(j,s|g,i)/\sum_s p(j,s|g,i), \tag{17}$$

$$p(s|h,i) = \sum_{g,j} p(s|g,i,j)\cdot (p(g|h,i)\cdot p(j|h,i)), \tag{18}$$

$$p(l|g,j,s) = p(l,j|g,s)/\sum_l p(l,j|g,s), \tag{19}$$

$$p(l|h,i) = \sum_{g,j,s} p(l|g,j,s)\cdot (p(g|h,i)\cdot p(j|h,i)\cdot p(s|h,i)). \tag{20}$$

The outward pass can be illustrated in Fig. 6 (ii), where all buckets from (i) were updated.

Whereas VE's inward pass in Fig. 2 (vi) constructed $p(j|h=0,i=1)$ at the root node, Cozman's outward pass constructed posteriors for all nodes in Fig. 2 (vi), namely, $p(g|h=0,i=1)$ in (12), $p(d|h=0,i=1)$ in (14), $p(c|h=0,i=1)$ in (16), $p(s|h=0,i=1)$ in (18), and $p(l|h=0,i=1)$ in (20).

The precomputation performed during the outward pass in [4] can be exploited when answering subsequente queries as demonstrated in Example 9.

Example 9. Given a new query $p(s|h=0,i=1)$, Cozman's method can readily answer it using (18).

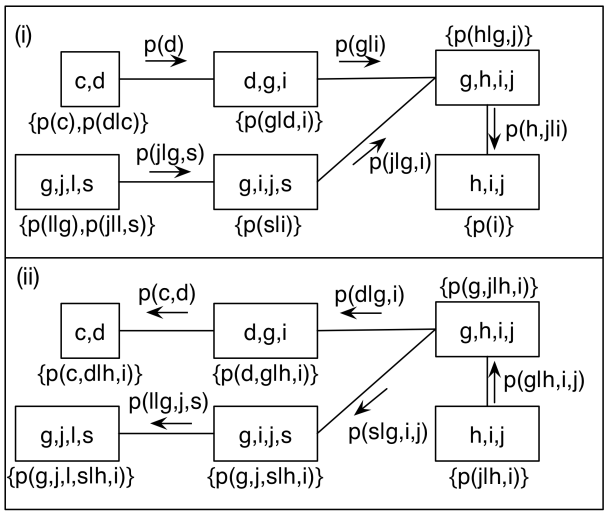

Fig. 6. In Example 8, (i) shows a tree of buckets for the inward pass and (ii) for the outward pass as described in [4].

6 Experimental Results

In the section, we compare and contrast VE, Cozman's approach, and marginal tree inference.

While VE is a simple approach to BN inference, repeatedly applying VE on the original BN can result in computation being duplicated, as previously discussed in Section 3.

Cozman [4] can alleviate some repeated computation. For example, as shown in Example 9, the computation in (17)-(18) for answering an initial query can be reused to answer a subsequent query in Example 9.

On the other hand, the price to pay is that precomputation can build tables that are never used. For instance, a second look at Example 9 reveals that the tables $p(l|g,j,s)$ in (19) and $p(l|h,i)$ in (20) are not reused. This means that the computation performed in (19)-(20) was wasteful.

We attempt here to stake out middle ground. One aim is to exploit VE as it is such a simple and clear approach to inference. Like Cozman, we seek to avoid repeated computation in VE. Unlike Cozman, however, our philosophy is to never build a probability table that goes unused.

MTI saves VE's past computation in marginal trees. Each time a new query is issued, MTI checks to see whether VE's past computation can be reused. If yes, MTI determines both that computation which can be reused and that computation which is missing. MTI proceeds to build only the missing computation. In this way, the query is answered and all tables built by MTI are used.

A preliminary empirical analysis is conducted as follows. We assume binary variables. We consider the following six queries:

$$
\begin{aligned}
Q_1 &= p(j|h=0, i=1),\\
Q_2 &= p(s|h=0, i=1),\\
Q_3 &= p(d|h=0, i=1),\\
Q_4 &= p(j|h=0, i=1, l=0),\\
Q_5 &= p(d|h=0, i=1, l=0),\\
Q_6 &= p(s|l=0).
\end{aligned}
$$

Observe that the queries involve the same evidence, additional evidence, and retracted evidence in this order. Table 1 shows the number of multiplications, divisions, and additions for each approach to answer the six queries.

Table 1. Number of ($\cdot$, / , $+$) to answer six queries in VE, MTI and Cozman's algorithm.

$\frac{Algorithm}{Query}$	VE	MTI	Cozman
Q_1	(76,8,34)	(76,8,34)	(220,68,80)
Q_2	(72,8,34)	(44,8,20)	(0,0,24)
Q_3	(84,8,38)	(48,8,20)	(0,0,8)
Q_4	(84,16,38)	(80,16,32)	(212,68,100)
Q_5	(132,16,50)	(128,16,48)	(0,0,16)
Q_6	(56,4,24)	(44,4,22)	(136,48,74)
Total	(504,60,218)	(420,60,210)	(568, 184, 302)

Table 1 shows several interesting points. First, the number of multiplications, divisions, and additions for MTI will never exceed those for VE, respectively. MTI seems to show a small but noticeable improvement over VE. The usefulness of Cozman's approach is clearly evident from queries Q_2, Q_3 and Q_5, but is overshadowed by the wasteful precomputation for the other queries. Based on these encouraging results, future work includes a rigorous comparison on numerous real world and benchmark BNs, as well as establishing theoretical properties of marginal tree inference.

7 Conclusions

Applying VE against a given Bayesian network for each query can result in repeated computation. Cozman [4] alleviates some of this repeated computation, but at the expense of possibly building tables that remain unused. Our approach in this paper is to stake out middle ground.

Marginal tree inference seeks to maximize the reuse of VE's past computation, while at the same time ensuring that every table built is used to answer a query. This is consistent with [3], where it is emphasized that VE does not support precomputation.

Future work will investigate relationships and provide empirical comparisons between MTI and join tree propagation [5], including Lazy propagation [9–12] and various extensions such as using prioritized messages [13], message construction using multiple methods [14–16], and exploiting semantics [7, 8].

References

1. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
2. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
3. Zhang, N.L., Poole, D.: A simple approach to Bayesian network computations. In: Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference. (1994) 171–178
4. Cozman, F.G.: Generalizing variable elimination in Bayesian networks. In: Workshop on Probabilistic Reasoning in Artificial Intelligence, Atibaia, Brazil (2000)
5. Shafer, G.: Probabilistic Expert Systems. Volume 67. Philadelphia: Society for Industrial and Applied Mathematics (1996)
6. Tarjan, R., Yannakakis., M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal on Computing **13**(3) (1984) 566–579
7. Butz, C.J., Yao, H., Hua, S.: A join tree probability propagation architecture for semantic modeling. Journal of Intelligent Information Systems **33**(2) (2009) 145–178
8. Butz, C.J., Yan, W.: The semantics of intermediate cpts in variable elimination. In: Fifth European Workshop on Probabilistic Graphical Models. (2010)
9. Madsen, A.L., Butz, C.J.: Ordering arc-reversal operations when eliminating variables in Lazy AR propagation. International Journal of Approximate Reasoning **54**(8) (2013) 1182–1196
10. Madsen, A.L., Jensen, F.: Lazy propagation: A junction tree inference algorithm based on Lazy evaluation. Artificial Intelligence **113**(1-2) (1999) 203–245
11. Madsen, A.L.: Improvements to message computation in Lazy propagation. International Journal of Approximate Reasoning **51**(5) (2010) 499–514
12. Madsen, A.L., Butz, C.J.: On the importance of elimination heuristics in Lazy propagation. In: Sixth European Workshop on Probabilistic Graphical Models (PGM). (2012) 227–234
13. Butz, C.J., Hua, S., Konkel, K., Yao, H.: Join tree propagation with prioritized messages. Networks **55**(4) (2010) 350–359

14. Butz, C., Hua, S.: An improved Lazy-ar approach to Bayesian network inference. In: Nineteenth Canadian Conference on Artificial Intelligence (AI). (2006) 183–194
15. Butz, C.J., Konkel, K., Lingras, P.: Join tree propagation utilizing both arc reversal and variable elimination. International Journal of Approximate Reasoning **52**(7) (2011) 948–959
16. Butz, C.J., Chen, J., Konkel, K., Lingras, P.: A formal comparison of variable elimination and arc reversal in Bayesian network inference. Intelligent Decision Technologies **3**(3) (2009) 173–180