

# The Bucket Elimination Algorithm for Belief Net Inference

R Greiner  
October 22, 2006

In general, the input to the BUCKELIM algorithm is

- a belief net (structure plus CPTables)  
here: the Belief Net shown in Figure ??
- a specific query  
here:  $P(G = 1) = ?$

A simplified version of the BUCKELIM algorithm appears in Figure ??; we will discuss some enhancements below. The first challenge is computing an ordering of the variables. The algorithm will produce the correct answer for any ordering, however the *time* required to produce the answer will be much longer with some orderings, versus others. Here, we will use  $\rho = \langle A, C, B, F, D, G \rangle$ ; we will view each variable as *indexed* by its position here. Hence, the index for  $A$  is  $\rho^{-1}(A) = 1$  as it is in  $\rho$ 's first position, the index for  $B$  is  $\rho^{-1}(B) = 2, \dots$ , and through  $\rho^{-1}(G) = 6$ .

The BN in Figure ?? has 6 variables; we therefore create  $6 + 1$  buckets, one for each variable and one for  $\{\}$ ; hence,  $\{\beta[\{\}], \beta[A], \beta[C], \beta[B], \beta[F], \beta[D], \beta[G]\}$ . We now take the 6 CPTable functions (shown in Figure ??), and put each in exactly one bucket — the bucket associated with the largest index of any of the arguments. For example,  $\lambda_{B|A}(b, a)$  involves  $B$  and  $A$ , with indices  $\rho^{-1}(B) = 3$  and  $\rho^{-1}(A) = 1$ ; as  $\rho^{-1}(B) > \rho^{-1}(A)$ , we store this CPTable-function in the 3<sup>th</sup> bucket (origin 0),  $\beta[B]$ . Similarly, as  $\max\{\rho^{-1}(D), \rho^{-1}(A), \rho^{-1}(B)\} = 5$ , we store  $\lambda_{D|A,B}(d, a, b)$  in  $\beta[D]$ , etc. Recall we are dealing with the instantiation  $G = 1$ ; we therefore

BUCKELIM(  $\langle N, E, \Lambda \rangle$ : BeliefNet;  $P(X = x)$ : query )

1. Let  $\rho$  be an ordering of the variables  $N$   
Create  $n + 1$  buckets, labeled  $\{\beta[\{\}], \beta[X_{\rho[1]}], \dots, \beta[X_{\rho[n]}\}]$
2. Instantiate the CPTable-functions, based on  $X = x$  evidence.
3. Place each instantiated CPTable-function  $\lambda_{X_i|X_{j_1}, \dots, X_{j_n}}(\dots)$  into bucket for  $X_k = \operatorname{argmax}_\ell \{X_i, X_{j_1}, \dots, X_{j_n}\}$
4. Go through buckets  $\beta[X_i]$  in *reverse* order  $\langle X_{\rho[n]}, X_{\rho[n-1]}, \dots, X_{\rho[2]}, X_{\rho[1]}, \{\} \rangle$   
For bucket  $X$ , with  $\beta[X] = \{\lambda_{X,1}, \dots, \lambda_{X,k}\}$ :  
Let  $\vec{Y} = (\bigcup_i \operatorname{args}(\lambda_{X,i})) - X$   
 $\lambda(\vec{Y}) = \sum_x \lambda_{X,1} \bowtie \dots \bowtie \lambda_{X,k}$   
Drop  $\lambda(\vec{Y})$  into  $\beta[Y_{max}]$  ( $Y_{max} = \operatorname{argmax}_{Y \in \vec{Y}} \{\rho[Y]\}$  has highest  $\rho$ -index within  $\vec{Y}$ ;  $\beta[\{\}]$  if  $|\vec{Y}| = 0$ )
5. Return  $\prod \beta[\{\}]$

Figure 1: BUCKELIM algorithm

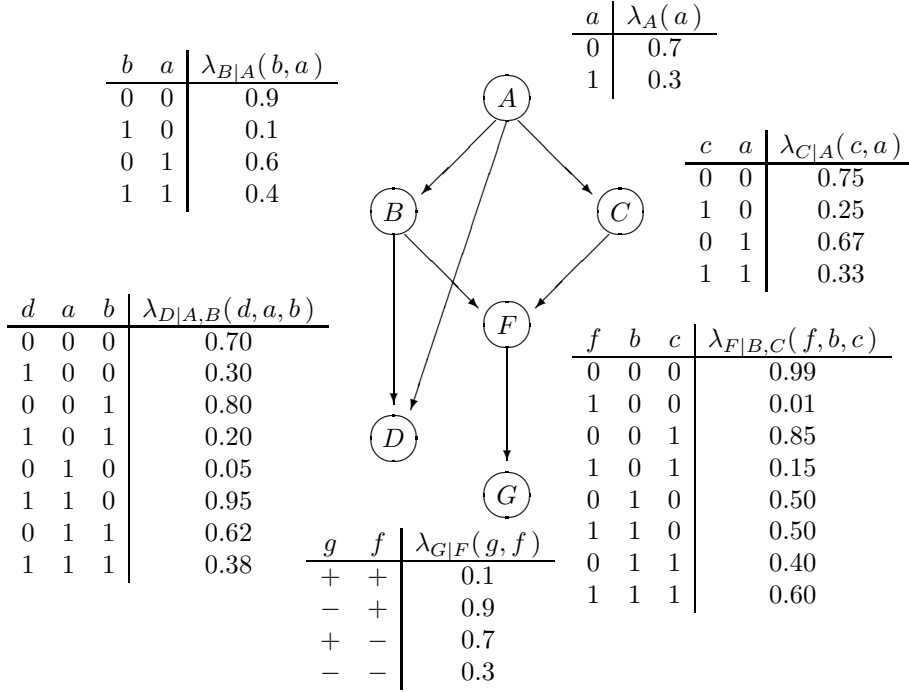


Figure 2: Simple Belief Net — Structure + Parameters

consider just the “ $G = 1$ ” sub-function of the

$$\lambda_{G|F}(g, f) = \left\{ \begin{array}{c|c} \frac{g}{f} & \lambda_{G|F}(g, f) \\ \hline + & 0.1 \\ - & 0.9 \\ + & 0.7 \\ - & 0.3 \end{array} \right\}$$

function, which we write as

$$\lambda_{+G|F}(f) = \left\{ \begin{array}{c|c} \frac{f}{\lambda_{+G|F}(f)} \\ \hline + & 0.1 \\ - & 0.7 \end{array} \right\}$$

As this function depends only on  $F$  (and not  $G$ ), we place it in the  $\beta[F]$  bucket. The results appear in first row of Figure ??, labeled  $\#0$ . (Note we have boxed this  $\lambda_{+G|F}(f)$  function.) Observe that the CPtable associated with the node  $X$  is not always stored in the bucket associated with  $X$ ,  $\beta[X]$ .

Now for step 4 of Figure ??: Here BUCKELIM processes the buckets in the *reverse* order from  $\rho$  — *i.e.*,  $\langle \beta[G], \beta[D], \beta[F], \beta[B], \beta[C], \beta[A] \rangle$ . As the first of these buckets,  $\beta[G]$ , is empty, BUCKELIM progresses to the second,  $\beta[D]$ .

Here there is only a single function,  $\lambda_{D|A,B}(d, a, b)$  (which is shown explicitly in Figure ??). When we sum out  $D$ , we get the (degenerate)

$$\lambda_{\bar{D}|A,B}(a, b) = \left\{ \begin{array}{c|c} \frac{a}{b} & \lambda_{\bar{D}|A,B}(a, b) \\ \hline 0 & 0.70 + 0.30 = 1.0 \\ 0 & 0.80 + 0.20 = 1.0 \\ 1 & 0.05 + 0.95 = 1.0 \\ 1 & 0.62 + 0.38 = 1.0 \end{array} \right\}$$

	$\beta[\{\}]$	$\beta[A]$	$\beta[C]$	$\beta[B]$	$\beta[F]$	$\beta[D]$	$\beta[G]$
#0	-	$\lambda_A(a)$	$\lambda_{C A}(c, a)$	$\lambda_{B A}(b, a)$	$\lambda_{F B,C}(f, b, c)$ $\lambda_{+G F}(f)$	$\lambda_{D A,B}(d, a, b)$	-

% Let  $\lambda_{\tilde{D}:A,B}(a, b) = \sum_d \lambda_{D|A,B}(d, a, b)$

#1	-	$\lambda_A(a)$	$\lambda_{C A}(c, a)$	$\lambda_{B A}(b, a)$ $\lambda_{\tilde{D}:A,B}(a, b)$	$\lambda_{F B,C}(f, b, c)$ $\lambda_{+G F}(f)$	-	-
----	---	----------------	-----------------------	--	---	---	---

% Let  $\lambda_{\tilde{F}:B,C}(b, c) = \sum_f \lambda_{F|B,C}(f, b, c) \times \lambda_{+G|F}(f)$

#2	-	$\lambda_A(a)$	$\lambda_{C A}(c, a)$ $\lambda_{\tilde{D}:A,B}(a, b)$ $\lambda_{\tilde{F}:B,C}(b, c)$	$\lambda_{B A}(b, a)$	-	-	-
----	---	----------------	---	-----------------------	---	---	---

% Let  $\lambda_{\tilde{B}:A,C}(a, c) = \sum_b \lambda_{B|A}(b, a) \times \lambda_{\tilde{D}:A,B}(a, b) \times \lambda_{\tilde{F}:B,C}(b, c)$

#3	-	$\lambda_A(a)$	$\lambda_{C A}(c, a)$ $\lambda_{\tilde{B}:A,C}(a, c)$	-	-	-	-
----	---	----------------	--	---	---	---	---

% Let  $\lambda_{\tilde{C}:A}(a) = \sum_c \lambda_{C|A}(c, a) \times \lambda_{\tilde{B}:A,C}(a, c)$

#4	-	$\lambda_A(a)$ $\lambda_{\tilde{C}:A}(a)$	-	-	-	-	-
----	---	--	---	---	---	---	---

% Let  $\lambda_{\tilde{A}}() = \sum_a \lambda_A(a) \times \lambda_{\tilde{C}:A}(a)$

#5	$\lambda_{\tilde{A}}()$	-	-	-	-	-	-
----	-------------------------	---	---	---	---	---	---

Figure 3: Buckets during BUCKELIM's computation

(We will later discuss optimizations that allow us to ignore buckets that would produce such degenerate functions.)

This new function is (formally) a function of  $A$  and  $B$ ; it is placed in the bucket associated with  $B$  as  $\rho^{-1}(B) = 3 > \rho^{-1}(A) = 1$ . The result appears in the row labeled #1 in Figure ??.

BUCKELIM next processes  $\beta[F]$ , which contains two functions:  $\lambda_{F|B,C}(f, b, c)$  and  $\lambda_{+G|F}(f)$ . We first take the “join” of these two functions, to produce a new function of the variables  $\{F, B, C\} \cup \{F\}$ : This is done by component-by-component multiplication:

$$\lambda_{F|B,C}(f, b, c) \bowtie \lambda_{+G|F}(f) = \left\{ \begin{array}{c|c} \begin{array}{ccc} f & b & c \end{array} & \lambda_{F|B,C}(f, b, c) \times \lambda_{+G|F}(f) \\ \hline 0 & 0 & 0 & 0.99 \times 0.7 = 0.694 \\ 1 & 0 & 0 & 0.01 \times 0.1 = 0.001 \\ 0 & 0 & 1 & 0.85 \times 0.7 = 0.595 \\ 1 & 0 & 1 & 0.15 \times 0.1 = 0.015 \\ 0 & 1 & 0 & 0.50 \times 0.7 = 0.350 \\ 1 & 1 & 0 & 0.50 \times 0.1 = 0.050 \\ 0 & 1 & 1 & 0.40 \times 0.7 = 0.280 \\ 1 & 1 & 1 & 0.60 \times 0.1 = 0.060 \end{array} \right.$$

We then sum-out on  $F$ , on this new  $\lambda_{F|B,C}(f, b, c) \bowtie \lambda_{+G|F}(f)$  function, to produce the function of  $B$  and  $C$ :

$$\lambda_{\tilde{F}:B,C}(b, c) = \left\{ \begin{array}{c|c} \begin{array}{cc} b & c \end{array} & \sum_f \lambda_{F|B,C}(f, b, c) \times \lambda_{+G|F}(f) \\ \hline 0 & 0 & 0.694 + 0.001 = 0.695 \\ 0 & 1 & 0.595 + 0.015 = 0.610 \\ 1 & 0 & 0.350 + 0.050 = 0.400 \\ 1 & 1 & 0.280 + 0.060 = 0.340 \end{array} \right.$$

As  $\rho^{-1}(B) = 3 > \rho^{-1}(C) = 2$ , BUCKELIM moves this function to  $\beta[B]$ ; see row #2 in Figure ??.

(Notice this tabular function is not a probability distribution, and so these values do *not* have to add up to 1. Also the result of running BUCKELIM on this new set of buckets, will produce the same results as running BUCKELIM on the original set.)

BUCKELIM then deals with this  $\beta[B]$  bucket, which includes 3 functions. Again it first computes the join, which will be over the variables  $\{B, A\} \cup \{A, B\} \cup \{B, C\} = \{B, C, A\}$ :

$$\lambda_{B|A}(b, a) \bowtie \lambda_{\tilde{D}|A,B}(a, b) \bowtie \lambda_{\tilde{F}:B,C}(b, c) = \left\{ \begin{array}{c|c} \begin{array}{ccc} b & a & c \end{array} & \lambda_{B|A}(b, a) \times \lambda_{\tilde{D}|A,B}(a, b) \times \lambda_{\tilde{F}:B,C}(b, c) \\ \hline 0 & 0 & 0 & 0.9 \times 1.0 \times 0.695 = 0.625 \\ 1 & 0 & 0 & 0.1 \times 1.0 \times 0.400 = 0.040 \\ 0 & 0 & 1 & 0.9 \times 1.0 \times 0.610 = 0.549 \\ 1 & 0 & 1 & 0.1 \times 1.0 \times 0.340 = 0.034 \\ 0 & 1 & 0 & 0.6 \times 1.0 \times 0.695 = 0.417 \\ 1 & 1 & 0 & 0.4 \times 1.0 \times 0.400 = 0.160 \\ 0 & 1 & 1 & 0.6 \times 1.0 \times 0.610 = 0.366 \\ 1 & 1 & 1 & 0.4 \times 1.0 \times 0.340 = 0.136 \end{array} \right.$$

then sums out the  $B$  variable, to produce

$$\lambda_{\tilde{B}:A,C}(a,c) = \left\{ \begin{array}{c|c} a & c \\ \hline 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array} \left| \begin{array}{l} \sum_b \lambda_{B|A}(b,a) \times \lambda_{\tilde{D}|A,B}(a,b) \times \lambda_{\tilde{F}:B,C}(b,c) \\ 0.625 + 0.040 = 0.665 \\ 0.549 + 0.034 = 0.583 \\ 0.417 + 0.160 = 0.577 \\ 0.366 + 0.136 = 0.502 \end{array} \right. \right.$$

This is deposited to  $\beta[C]$ ; see #3.

Onto bucket  $\beta[C]$ , with its two functions. The value of  $\lambda_{\tilde{C}:A}(a) = \sum_c \lambda_{C|A}(c,a) \times \lambda_{\tilde{B}:A,C}(a,c)$  is

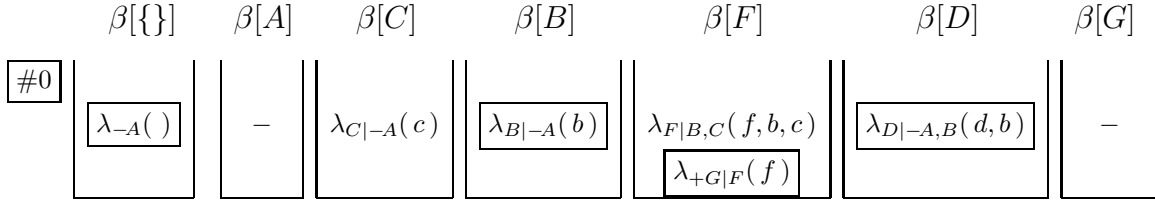
$$\lambda_{\tilde{C}:A}(a) = \left\{ \begin{array}{c|c} a & \sum_c \lambda_{C|A}(c,a) \times \lambda_{\tilde{B}:A,C}(a,c) \\ \hline 0 & 0.665 \times 0.75 + 0.583 \times 0.25 = 0.643 \\ 1 & 0.577 \times 0.67 + 0.502 \times 0.33 = 0.551 \end{array} \right.$$

This function is deposited in bucket  $\beta[A]$ ; see row #4 of Figure ???. Finally, we compute the value  $\lambda_{\tilde{A}}() = \sum_a \lambda_A(a) \times \lambda_{\tilde{C}:A}(a) = 0.7 \times 0.643 + 0.3 \times 0.551 = 0.615$ , which is placed in  $\beta[\{\}]$ .

BUCKELIM is now on step 5 of Figure ???: it simply returns this 0.615 value. (In general, there may have been several numbers here; if so, BUCKELIM would return their product.)

## 1 More complicated queries

We can use this same approach to deal with  $> 1$  variables: For example, to compute  $P(A = 0, G = 1)$ , we would start with



Notice several functions are reduced to have fewer arguments, and some of these reductions move the function to a different bucket (e.g.,  $\lambda_A(a)$  was in  $\beta[A]$ , but the reduced  $\lambda_{-A}()$  is in  $\beta[\{\}]$ ).

We can use this BUCKELIM system to deal with *conditional* queries: In general, we need just deal with unconditional expressions (using the BUCKELIM algorithm shown above), then use addition and division. For example, suppose we wanted to compute  $P(A = a | G = 1)$ . Given that

$$P(A = 0 | C = 1) = \frac{P(A = 0, C = 1)}{P(C = 1)} = \frac{P(A = 0, C = 1)}{P(A = 0, C = 1) + P(A = 1, C = 1)}$$

we would just use BUCKELIM to compute  $P(\underline{A} = a, G = 1)$  for all  $a$ , etc.

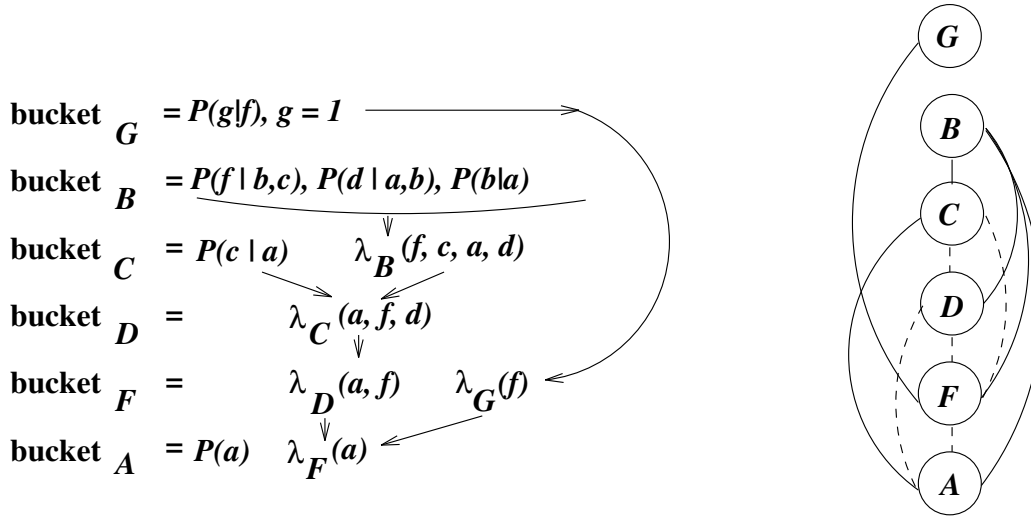


Figure 4: Another Ordering

## 2 Efficiency Tricks

There are several of ways to make the computation relatively efficient. (Of course, this is only relative — the underlying problem remains NP-hard.)

The cost of the computation depends critically on the “ordering”  $\rho$ , which is (somehow) determined in Step1 of Figure ???. One heuristic trick, incorporated above, is to move the instantiated variables (such as  $G$ , given the query involved  $G = 1$ ) to the end of the ordering.

Another idea is to move the “varying variables” to the front of the ordering. For example, when the query is  $P(A = 0 | G = 1)$ , we will be considering all values of  $A$ . There is a part of the computation that does not involve  $A$ ; a clever-er version of BUCKELIM could first do this sub-computation once, then use it for each value of  $A$ . That is, we should move  $A$ , and the variables whose CPTable functions involve  $A - D, B, C$  — to the front of the ordering.

Also, as we saw above, we can ignore any bucket(s) that produce a function that is identically 1. This happened for the  $\beta[D]$  above, as we were summing out  $D$  for  $\lambda_{D|A,B}(d, a, b)$ .

The general idea is: BUCKELIM *can ignore any bucket that is  $d$ -separated from the query* — *i.e.*, we can skip bucket  $\beta[X_i]$  if it contains no

- evidence variables
- query variable,
- newly-computed  $\lambda_X(Y)$

Finally, notice the time and space efficiency of this entire process is *exponential* in the size of largest bucket, and so depends on this “high-water mark” quantity; this is called the “**Induced width**” of an ordering (wrt the given structure). Here, for example, every bucket has  $\leq 3$  variables, which means this system is  $O(D^3)$  where  $D = |Dom(X_i)|$  is size of domain.

This quantity, in turn, depends on the ordering of the variables selected. For example, consider the ordering  $\rho_2 = \langle A, F, D, C, B, G \rangle$ . As shown Figure ??, this  $\rho_2$  produces a bucket with 4 functions, and so has complexity  $O(|Dom|^4)$ ; this is more than the earlier  $\rho$ , which had complexity  $O(|Dom|^3)$ ,

Note it is NP-hard to find best ordering  $\rho$  — *i.e.*, the ordering producing the minimal induced width.