An Extension of the Differential Approach for Bayesian Network Inference to Dynamic Bayesian Networks

Boris Brandherm
Department of Computer Science, Saarland University
P. O. Box 15 11 50, D-66041 Saarbrücken, Germany
brandherm@cs.uni-sb.de

Anthony Jameson
DFKI, German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
jameson@dfki.de

Abstract

We extend the differential approach to inference in Bayesian networks (BNs) (Darwiche, 2000) to handle specific problems that arise in the context of dynamic Bayesian networks (DBNs). We first summarize Darwiche's approach for BNs, which involves the representation of a BN in terms of a multivariate polynomial. We then show how procedures for the computation of corresponding polynomials for DBNs can be derived. These procedures permit not only an exact roll-up of old time slices but also a constant-space evaluation of DBNs. The method is applicable to both forward and backward propagation, and it does not presuppose that each time slice of the DBN has the same structure. It is compatible with approximative methods for roll-up and evaluation of DBNs. Finally, we discuss further ways of improving efficiency, referring as an example to a mobile system in which the computation is distributed over a normal workstation and a resource-limited mobile device.

1 Introduction

Dynamic Bayesian networks (DBNs) are an extension of Bayesian networks (BNs). With a DBN, it is possible to model dynamic processes: Each time the DBN receives new evidence, a new time slice is added to the existing DBN. Each time slice must satisfy the Markov assumption: The future is conditionally independent of the past, given the current state. In principle, DBNs can be evaluated with the same inference procedures as normal BNs; but their dynamic nature places heavy demands on computation time and memory. Therefore, it is necessary to apply roll-up procedures that cut off old time slices without eliminating their influence on the newer time slices. For this condition to be fulfilled, a probability table representing the *belief state* has to be maintained: a probability distribution over the state of the system at a given time. This belief state includes at least all of the nodes that are parents of nodes in the next time slice. The representation of this belief state as a probability table raises complexity problems: This table cannot in general be given a factorized representation without loss of information (see, e.g., Boyen & Koller, 1998). Roll-up procedures may be exact (see, e.g., Kjærulff, 1995) or approximative (see, e.g., Boyen & Koller, 1999).

With his differential approach to inference in Bayesian networks, Darwiche (1999, 2000, 2003) presents an algorithm that compiles a BN into a multivariate polynomial that can be processed efficiently: After the polynomial and its partial derivatives have been computed, a large class of probabilistic computations, such as classical inference and sensitivity analysis, can be computed in constant time.

In this paper, we extend the differential approach so that DBNs can be handled relatively efficiently, like (static) BNs. In particular, once an approximate factorization of the belief state has been computed, the effects of this approximation on the values of particular nodes can be analyzed with the differential approach.

In Section 2, we summarize the approach of Darwiche (2000) for BNs, looking at the canonical and factorial representation of multivariate polynomials that represent a BN. We will indicate which queries to the BN can be answered with the polynomial. In Section 3 we will show how the approach can be extended to DBNs, developing a procedure within the differential approach that makes it possible to perform roll-ups and inference in DBNs with constant space requirements. In Section 4, we briefly introduce the system READY, in which our procedures for solving DBNs are being applied. Using this system as an example, we discuss further ways of improving efficiency, which are also applicable in other domains.

2 Polynomial Representation of Bayesian Networks

A BN $\mathcal{N}$ can be represented graphically as a directed acyclic graph (see Figure 1). The nodes of the graph represent the variables[1] $X_1, \ldots, X_n$. The edges joining the nodes represent the dependencies among them. For each node X_i, there is a conditional probability table (CPT) $\boldsymbol{\theta}_{(X_i|\mathbf{pa}(X_i))}$ that quantifies these dependencies. (The parent nodes of X_i are denoted by $\mathbf{pa}(X_i)$.) A number of different procedures exist for evaluating BNs (see, e.g., Pearl, 1988; Jensen, 2001).

2.1 Canonical Representation of the Polynomial

Any BN that contains only discrete variables can be represented by a polynomial. In Figure 1, for example, let us look first only at the binary nodes A and B and their associated CPTs $\boldsymbol{\theta}_{(A)}$ and $\boldsymbol{\theta}_{(B|A)}$. The two possible values of A are denoted by a_1 and a_2, while those of B are denoted by b_1 and b_2. Suppose, for example, that we have evidence $\mathbf{e} = a_1$ and that we are interested in $\Pr(\mathbf{e})$. By multiplying the two CPTs, we obtain

[1]In this paper, we use the terms *node* and *variable* interchangeably.

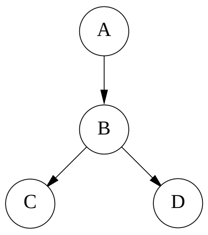


Figure 1: Example of a Bayesian network with four binary nodes. ($\Pr(A,B,C,D)$ = $\Pr(A) * \Pr(B \mid A) * \Pr(C \mid B) * \Pr(D \mid B)$.)

an overall probability table $\boldsymbol{\theta}_{(A,B)}$ that represents the joint probability distribution of the nodes A and B:

$$\boldsymbol{\theta}_{(A,B)} = \boldsymbol{\theta}_{(A)}\ \boldsymbol{\theta}_{(B|A)} = \begin{bmatrix} \theta_{a_1} \\ \theta_{a_2} \end{bmatrix} \begin{bmatrix} \theta_{b_1|a_1} & \theta_{b_1|a_2} \\ \theta_{b_2|a_1} & \theta_{b_2|a_2} \end{bmatrix} = \begin{bmatrix} \theta_{a_1}\theta_{b_1|a_1} & \theta_{a_2}\theta_{b_1|a_2} \\ \theta_{a_1}\theta_{b_2|a_1} & \theta_{a_2}\theta_{b_2|a_2} \end{bmatrix} .$$

Now we need only to add up the table entries (probabilities) that are consistent with a_1, and we obtain the desired result:

$$\Pr(\mathbf{e}) = \theta_{a_1}\theta_{b_1|a_1} + \theta_{a_1}\theta_{b_2|a_1}.$$

For each variable in the BN, it is possible that evidence has been obtained. Such evidence can be represented by an *evidence vector* λ_{X_i} for the node in question. The vector contains a 1 for the state which is realized and a 0 for each other state of the variable. (When no evidence concerning a given variable is available, each element of the evidence vector is 1.)

We can use the evidence vectors to parameterize the probability table, so that it takes into account the available evidence, as follows:

$$\boldsymbol{\theta}_{(A,B)}\lambda_A\lambda_B = \boldsymbol{\theta}_{(A)}\lambda_A\ \boldsymbol{\theta}_{(B|A)}\lambda_B = \begin{bmatrix} \theta_{a_1}\theta_{b_1|a_1}\lambda_{a_1}\lambda_{b_1} & \theta_{a_2}\theta_{b_1|a_2}\lambda_{a_2}\lambda_{b_1} \\ \theta_{a_1}\theta_{b_2|a_1}\lambda_{a_1}\lambda_{b_2} & \theta_{a_2}\theta_{b_2|a_2}\lambda_{a_2}\lambda_{b_2} \end{bmatrix} .$$

By adding up all of the table entries, we obtain a multivariate polynomial:

$$\begin{aligned} \mathcal{F}(\lambda_A, \lambda_B, \boldsymbol{\theta}_{(A)}, \boldsymbol{\theta}_{(B|A)}) = {} & \theta_{a_1}\theta_{b_1|a_1}\lambda_{a_1}\lambda_{b_1} + \theta_{a_2}\theta_{b_1|a_2}\lambda_{a_2}\lambda_{b_1} + \\ & \theta_{a_1}\theta_{b_2|a_1}\lambda_{a_1}\lambda_{b_2} + \theta_{a_2}\theta_{b_2|a_2}\lambda_{a_2}\lambda_{b_2} \ . \end{aligned}$$

Now we can determine through the evidence vectors which table entries are to be added up. The λ_{x_i} that are consistent with the evidence $\mathbf{e}$ are set to 1, while the other λ_{x_i} are set to 0. In our example, therefore, $\lambda_{a_2} = 0$ and $\lambda_{a_1} = \lambda_{b_1} = \lambda_{b_2} = 1$.

With this polynomial, for example, we can compute the beliefs for particular nodes in the network, and we can perform various sensitivity analyses. Much more detail can be found in the articles by Darwiche.

In general, the multivariate polynomial is computed as follows: First we multiply all of the CPTs of the nodes of the BN and their evidence vectors λ_{X_i} to obtain a table $\mathbf{T}$ that contains the probabilities of the individual hypothesis combinations, that is:

$$\mathbf{T} = \Pi_{i=1}^{n}\ \boldsymbol{\theta}_{(X_i|\mathbf{pa}(X_i))}\lambda_{X_i} \ .$$

Adding up all of the table entries in $\mathbf{T}$, we obtain the multivariate polynomial in canonical form (see also Darwiche, 2000).

$$\mathcal{F}(\lambda_{X_i}, \boldsymbol{\theta}_{(X_i|\mathbf{pa}(X_i))}) = \Sigma_{\mathbf{X}}\, \Pi_{i=1}^{n}\, \boldsymbol{\theta}_{(X_i|\mathbf{pa}(X_i))}\lambda_{X_i} \ .$$

The size of a multivariate polynomial in canonical form is always exponential in the number of its variables. A way of avoiding the resulting complexity is offered by the factored representation of the polynomial, which is discussed in the following subsection.

2.2 Factored Representation of the Polynomial

In finding a factorization of the multivariate polynomial, we want to take into account the structure of the Bayesian network; that is, we want to exploit the factored representation of the joint probability distribution in order to reduce the complexity of the multivariate polynomial. We will not simply compute the product of all of the CPTs with their evidence vectors. Instead, we will first find an ordering of the nodes according to which the CPTs and their corresponding evidence vectors λ_{X_i} are to be multiplied. Early in this process, we will marginalize as many nodes as possible out of this product, so as to obtain the multivariate polynomial in factored instead of canonical form.

As an illustration, let us take once again the example of the nodes A and B. Suppose that the order of processing of the nodes is "First B, then A". This order is also called the *elimination order* $\pi = \langle B, A\rangle$. If we marginalize the nodes out as soon as possible, we obtain:

$$\begin{aligned}
&\mathcal{F}(\lambda_A, \lambda_B, \boldsymbol{\theta}_{(A)}, \boldsymbol{\theta}_{(B|A)}) = \mathcal{F}_{\langle B,A\rangle}(\lambda_A, \lambda_B, \boldsymbol{\theta}_{(A)}, \boldsymbol{\theta}_{(B|A)}) = \\
&\quad \Sigma_A\, \boldsymbol{\theta}_{(A)}\lambda_A\, (\Sigma_B\, \boldsymbol{\theta}_{(B|A)}\lambda_B) = \theta_{a_1}\lambda_{a_1}\, (\theta_{b_1|a_1}\lambda_{b_1} + \theta_{b_2|a_1}\lambda_{b_2}) + \\
&\qquad\qquad\qquad\qquad\qquad\qquad \theta_{a_2}\lambda_{a_2}\, (\theta_{b_1|a_2}\lambda_{b_1} + \theta_{b_2|a_2}\lambda_{b_2}) \ .
\end{aligned}$$

When we compare the polynomial in factored form with the polynomial in canonical form, we notice that there are some multiplications in the factored form that do not need to be computed. In connection with the notation of the elimination order, it should be noted that the following holds:

$$\Pi_{\langle B,A\rangle}\, \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X = \boldsymbol{\theta}_{(A)}\lambda_A\, \boldsymbol{\theta}_{(B|A)}\lambda_B$$

and

$$\begin{aligned}
&\Sigma_{\langle B,A\rangle}\, \boldsymbol{\theta}_{(A)}\lambda_A\, \boldsymbol{\theta}_{(B|A)}\lambda_B = \\
&\Sigma_{\langle A\rangle}\Sigma_{\langle B\rangle}\, \boldsymbol{\theta}_{(A)}\lambda_A\, \boldsymbol{\theta}_{(B|A)}\lambda_B = \\
&\Sigma_A\, \boldsymbol{\theta}_{(A)}\lambda_A\, (\Sigma_B\, \boldsymbol{\theta}_{(B|A)}\lambda_B).
\end{aligned}$$

We will use the following abbreviated notation for the multivariate polynomial in factored form:

$$\begin{aligned}
&\mathcal{F}_{\text{elim}(\{X\in\mathcal{N}\})}(\lambda_X, \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}) = \\
&\quad \Sigma_{\text{elim}(\{X\in\mathcal{N}\})}\, \Pi_{\text{elim}(\{X\in\mathcal{N}\})}\, (\boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X) = \\
&\quad \Sigma_{\langle X_1,\ldots,X_n\rangle}\, \Pi_{\langle X_1,\ldots,X_n\rangle}\, (\boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X) \ .
\end{aligned}$$

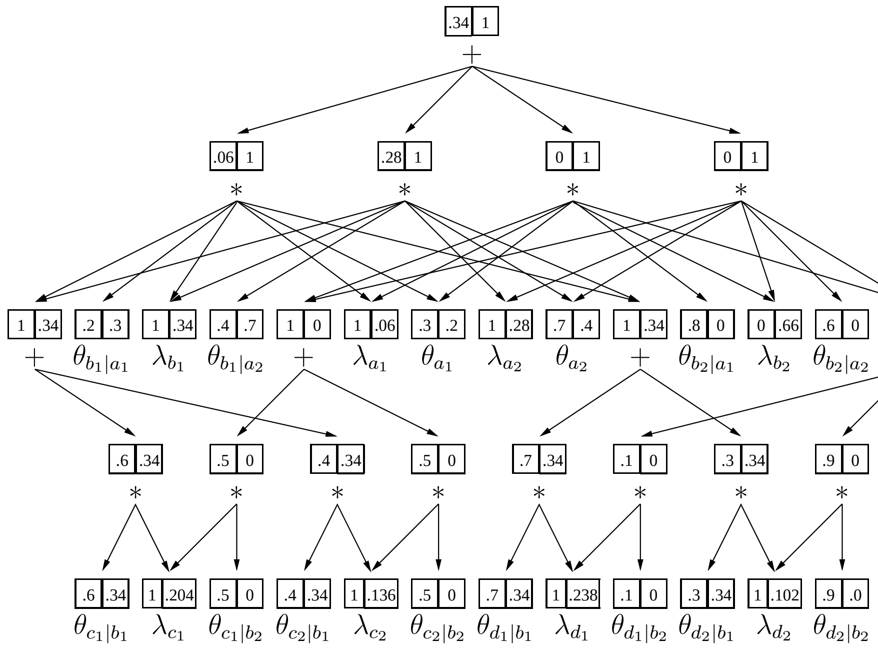


Figure 2: An arithmetic circuit which computes the polynomial $\Sigma_{A,B}\ \boldsymbol{\theta}_A\lambda_A\ \boldsymbol{\theta}_{B|A}\lambda_B\ (\Sigma_C\ \boldsymbol{\theta}_{C|B}\lambda_C)\ (\Sigma_D\ \boldsymbol{\theta}_{D|B}\lambda_D)$. It has been evaluated (upward pass) and differentiated (downward pass) under evidence b_1. Registers *vr* are shown on the left, and registers *dr* are shown on the right.

Here, $\mathrm{elim}(\{X \in \mathcal{N}\})$ is a function that computes the elimination order of the variables.[2]

In Figure 2 an arithmetic circuit is shown that computes the polynomial $\Sigma_{A,B}\ \boldsymbol{\theta}_A\lambda_A\ \boldsymbol{\theta}_{B|A}\lambda_B$ $(\Sigma_C\ \boldsymbol{\theta}_{C|B}\lambda_C)\ (\Sigma_D\ \boldsymbol{\theta}_{D|B}\lambda_D)$ that represents the Bayesian network in Figure 1. In (Darwiche, 2003) it is shown how we obtain such an arithmetic circuit given a Bayesian network. This particular representation of the network polynomial facilitates its evaluation and differentation. Once the arithmetic circuit is evaluated and differentiated, a large number of probabilistic queries can be retrieved immediately (see Darwiche, 2003).

We will present here only a simple algorithm for evaluation and differentiation of an arithmetic circuit. The technical background and much more information about algorithms for evaluation and differentiation can be found in (Darwiche, 2003; Park & Darwiche, 2002). To every circuit node v, two registers *vr*(v) and *dr*(v) are assigned. In the upward pass, the values of the *vr*(v) registers are computed and the circuit is evaluated. In the downward pass, the values of the *dr*(v) registers are computed and the circuit is differentiated.

- Initialization: Set *dr*(v) to 1 for root v; for all other v set *dr*(v) to 0.
- Upward pass:

[2]The question of which such function is to be used does not concern us here; cf. Kjærulff (1995).

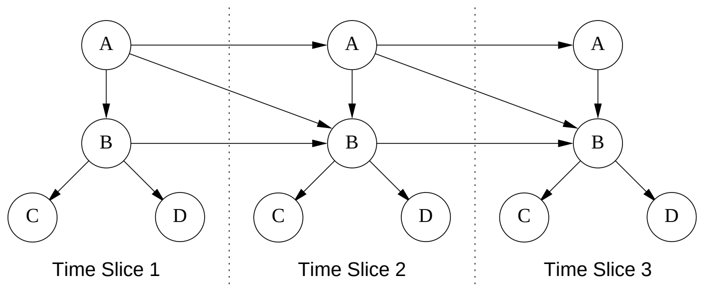


Figure 3: Example dynamic Bayesian network discussed in the text. ($\Pr(A_1, B_1, \ldots) = \Pr(A_1) * \Pr(B_1 \mid A_1) * \ldots$.)

  - If node p is labeled with an addition sign:
    Set the *vr* register of node p to Σ_v*vr*(v), where v are the children of p.
  - If node p is labeled with a multiplication sign:
    Set the *vr* register of node p to Π_v*vr*(v), where v are the children of p.

- Downward pass:
  - If node p is labeled with an addition sign:
    For all children v of p: Increment *dr*(v) by *dr*(p).
  - If node p is labeled with a multiplication sign:
    For all children v of p: Increment *dr*(v) by *dr*$(p)\Pi_{v'}$*vr*(v') , where v' are the other children of p.

In Figure 2 the values of the registers *vr* and *dr* are shown after the upward and downward pass under evidence b_1. With these values we can compute the belief value of a node. For example, for node C we get $\left(\frac{0.204}{0.34}, \frac{0.136}{0.34}\right) = (0.6, 0.4)$.

3 Computation of the Polynomial in DBNs

Now we will see how these ideas of Darwiche (2000) can be extended and applied to DBNs. Let us look at the DBN in Figure 3, which at this point comprises three time slices. This DBN includes two building blocks (so-called *time slice schemas*), shown in Figure 4. The first schema, used for the first time slice, is simply a BN. The second schema, which is used for each of the remaining time slices, is a BN plus a specification of the parent variables that are taken from the preceding time slice. Suppose that we have evidence $\mathbf{e} = \mathbf{e}_1, \ldots, \mathbf{e}_3$ for Time Slices 1 through 3 and that we want to know the *belief vector* for a node in Time Slice 2—i.e., the vector that expresses how likely each possible value of the variable is, given the currently available evidence.

We now have to distinguish between (a) forward propagation, which brings forward the impact of the evidence from Time Slice 1 to Time Slice 2; and (b) backward propagation, which brings the impact of evidence from Time Slice 3 back to Time Slice 2.

In the following we will define procedures that will allow us to determine the beliefs for nodes in an arbitrary time slice t (with $1 \leq t \leq L$); to cut off old time slices; and to

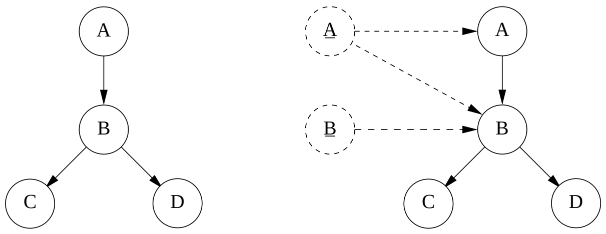


Figure 4: Time Slice Schemas 1 and 2 for the dynamic Bayesian network shown in Figure 3.

add new time slices. These computations can be performed in constant space (depending on the structure of the time slice schemas). We perform a roll-up by assigning values to the variables of the polynomial and then simplifying the polynomial by evaluating it. We add new time slices efficiently by recycling polynomials and computations from the preceding time slices.

Typically, we will not use a single arbitrary elimination order for the whole DBN; instead we will use an elimination order that is restricted to one time slice. In this way, we will obtain for each time slice schema a general procedure for the partial polynomial that corresponds to this time slice schema.

3.1 Forward Propagation

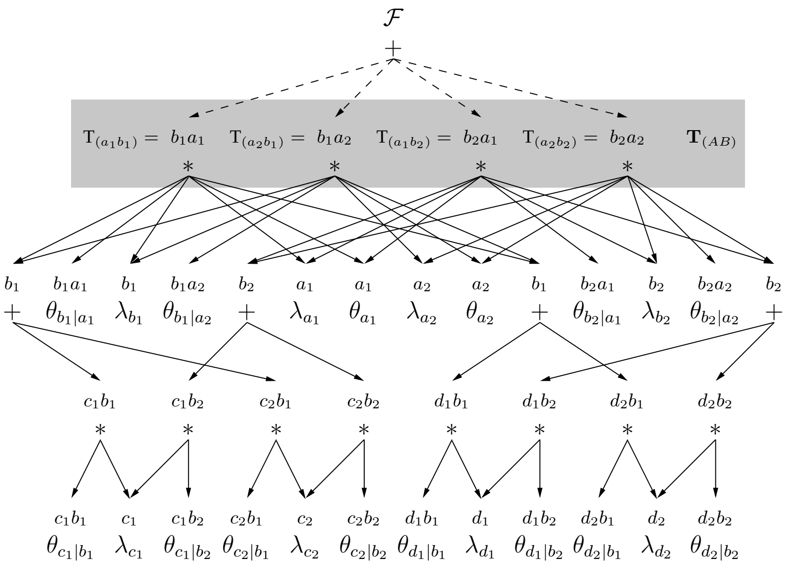
First, we will look at the case of forward propagation. To simplify exposition, we assume that we have only one time slice schema available for the instantiation of the initial time slice and also only one time slice schema for the instantiation of the following time slices, as in our example DBN. Later, we will sketch how it is possible to generalize the procedure to deal with an arbitrary number of time slice schemas for the instantiation of the initial and the succeeding time slices.

For the first time slice, we determine a procedure that permits a simple and efficient extension of the old polynomial when a new time slice is added to the DBN. Let us look at Time Slices 1 and 2 in Figure 3. We can see that in Time Slice 1 the parent nodes of the nodes in Time Slice 2 cannot be marginalized out until Time Slice 2 has been added. The result of the procedure for the first time slice is a table over the nodes of the current time slice that could not be marginalized out. Because only one time slice schema can be instantiated for the following time slices, we know which nodes in the current time slice will become parent nodes of nodes in the following time slices, and we can determine the nodes that belong to the belief state. (The set of these nodes is denoted by $\mathbf{bs}(.)$.) In our example, these are the nodes A_1 and B_1, so $\mathbf{bs}(1) = \{A_1, B_1\}$. The indices in the nodes denote the time slice to which they belong.

In our example DBN, we obtain as a procedure for forward propagation for the first time slice:

$$\text{fwd}_{\text{init}} = \boldsymbol{\theta}_{(A_1)}\lambda_{A_1}\ \boldsymbol{\theta}_{(B_1|A_1)}\lambda_{B_1}\ (\Sigma_{C_1}\ \boldsymbol{\theta}_{(C_1|B_1)}\lambda_{C_1})\ (\Sigma_{D_1}\ \boldsymbol{\theta}_{(D_1|B_1)}\lambda_{D_1})\ .$$

As was already mentioned, the result is a table over the Cartesian product of the hypothe-



$\mathcal{F}$
$\mathrm{T}_{(a_1b_1)} = b_1a_1$
$\mathrm{T}_{(a_2b_1)} = b_1a_2$
$\mathrm{T}_{(a_1b_2)} = b_2a_1$
$\mathrm{T}_{(a_2b_2)} = b_2a_2$
$\mathbf{T}_{(AB)}$
b_1 b_1a_1 b_1 b_1a_2 b_2 a_1 a_1 a_2 a_2 b_1 b_2a_1 b_2 b_2a_2 b_2
$\theta_{b_1|a_1}$ λ_{b_1} $\theta_{b_1|a_2}$ λ_{a_1} θ_{a_1} λ_{a_2} θ_{a_2} $\theta_{b_2|a_1}$ λ_{b_2} $\theta_{b_2|a_2}$
c_1b_1 c_1b_2 c_2b_1 c_2b_2 d_1b_1 d_1b_2 d_2b_1 d_2b_2
c_1b_1 c_1 c_1b_2 c_2b_1 c_2 c_2b_2 d_1b_1 d_1 d_1b_2 d_2b_1 d_2 d_2b_2
$\theta_{c_1|b_1}$ λ_{c_1} $\theta_{c_1|b_2}$ $\theta_{c_2|b_1}$ λ_{c_2} $\theta_{c_2|b_2}$ $\theta_{d_1|b_1}$ λ_{d_1} $\theta_{d_1|b_2}$ $\theta_{d_2|b_1}$ λ_{d_2} $\theta_{d_2|b_2}$

Figure 5: An arithmetic circuit that represents the procedure for forward propagation (solid lines) and the polynomial (dashed and solid lines) for Time Slice Schema 1 in Figure 4. (The table $\mathbf{T}_{(AB)}$ is highlighted with a gray background.)

ses of the nodes A_1 and B_1, that is $\mathrm{fwd}_{\text{init}} = \mathbf{T}_{(A_1B_1)}$. The table entries of $\mathbf{T}_{(A_1B_1)}$ itself are polynomials. We obtain the polynomial for the first time slice by summing over all of these table entries—that is, over all of the nodes that have been left out so far. With this polynomial, it is possible, for example, to compute the belief value of a node. In Figure 5, an arithmetic circuit is shown that represents the procedure for forward propagation (solid lines) and the polynomial (dashed and solid lines) for the first time slice. The table entries of $\mathbf{T}_{(A_1B_1)}$ are shown in Figure 5 as $\mathrm{T}_{(a_1b_1)}, \ldots, \mathrm{T}_{(a_2b_2)}$ at the top. A label (e.g., c_1b_1) is associated with each node in the arithmetic circuit. Consistent labels in each level span a table. For example, a_1 and a_2 span the table $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$, and $b_1a_1, \ldots, b_2a_2$ span the table $\begin{bmatrix} b_1a_1 & b_1a_2 \\ b_2a_1 & b_2a_2 \end{bmatrix}$.

In $\mathbf{T}_{(A_1B_1)}$, on the other hand, all of the information is contained that must be passed to a later time slice so as to permit exact inference. So far, no evidence has been taken into account through instantiation in the polynomial or in $\mathbf{T}_{(A_1B_1)}$. To simplify the polynomial or $\mathbf{T}_{(A_1B_1)}$, all of the λ_{x_i} of the noninteresting nodes can be set to the given evidence or to 1 if no evidence is given.

Now let us look at Time Slice 2 with its predecessor Time Slice 1 and its successor Time Slice 3 in Figure 3. We want to determine a procedure which takes into account the effects of the earlier time slice on the following time slices and which allows a simple and efficient extension of the old polynomial when a new time slice is added.

As was the case with the procedure for the first time slice, the nodes of the belief state cannot be marginalized out until the following time slice has been added. In our example, the nodes in question are A_2 and B_2.

We obtain the following as a procedure for forward propagation in the second time slice in our example in Figure 3:

$$\begin{aligned}&\mathrm{fwd}_2(\mathbf{T}_{(A_1B_1)}) = \\ &\quad(\Sigma_{C_2}\ \boldsymbol{\theta}_{(C_2|B_2)}\lambda_{C_2})\ (\Sigma_{D_2}\ \boldsymbol{\theta}_{(D_2|B_2)}\lambda_{D_2}) \\ &\quad(\Sigma_{A_1}\ \boldsymbol{\theta}_{(A_2|A_1)}\lambda_{A_2}\ (\Sigma_{B_1}\ \boldsymbol{\theta}_{(B_2|A_2,A_1,B_1)}\lambda_{B_2}\mathbf{T}_{(A_1B_1)})) = \mathbf{T}_{(A_2B_2)}\ .\end{aligned}$$

An arithmetic circuit that represents this procedure is shown in Figure 6. In this figure, hypotheses and nodes from the preceding time slice are underlined and indices indicating the time slice number are omitted. To save memory, both arithmetic circuits shown in Figure 5 and 6 can be merged to share common structures, for example $(\Sigma_C\ \boldsymbol{\theta}_{(C|B)}\lambda_C)$.

It is easy to see that the procedures for the following time slices are all identical and that the procedures for the time slices i (with $i \geq 1$) can be generalized to:

$$\begin{aligned}&\mathrm{fwd}_i(\mathbf{T}_{(A_{i-1}B_{i-1})}) = \\ &\quad(\Sigma_{C_i}\ \boldsymbol{\theta}_{(C_i|B_i)}\lambda_{C_i})\ (\Sigma_{D_i}\ \boldsymbol{\theta}_{(D_i|B_i)}\lambda_{D_i}) \\ &\quad(\Sigma_{A_{i-1}}\ \boldsymbol{\theta}_{(A_i|A_{i-1})}\lambda_{A_i}\ (\Sigma_{B_{i-1}}\ \boldsymbol{\theta}_{(B_i|A_i,A_{i-1},B_{i-1})}\lambda_{B_i}\mathbf{T}_{(A_{i-1}B_{i-1})})) = \mathbf{T}_{(A_iB_i)}\ .\end{aligned}$$

In the case where $i = 1$, the variables with the index 0 disappear, and we have:

$$\mathrm{fwd}_1(\mathbf{T}_{(A_0B_0)}) = \mathrm{fwd}_1(1).$$

The polynomial for Time Slices 1 through L is as follows:

$$\begin{aligned}\mathcal{F}_{\mathrm{elim}(\{X\in\mathcal{N}\})}(\lambda_{X_i}, \boldsymbol{\theta}_{(X_i|\mathbf{pa}(X_i))}) &= \Sigma_{\mathrm{elim}(\{A_L,B_L\})}(\mathrm{fwd}_L(\ldots\mathrm{fwd}_2(\mathrm{fwd}_1(1))\ldots)) \\ &\quad\Sigma_{\mathrm{elim}(\{A_L,B_L\})}(\mathrm{fwd}_L \circ \ldots \circ \mathrm{fwd}_2 \circ \mathrm{fwd}_1(1))\ .\end{aligned}$$

Before we show how the polynomial can be evaluated with constant space requirements, we want to capture the results obtained so far in a general notation. As a general procedure for forward propagation from the first time slice, we obtain:

$$\begin{aligned}\mathrm{fwd}_{\text{init}} = {}&\Sigma_{\mathrm{elim}(\{X|X\in\mathrm{TS}(1)\setminus\mathbf{bs}(1)\})} \\ &\Pi_{X\in\mathrm{TS}(1)}\ \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X = \\ &\Sigma_{\mathrm{elim}(\{X|X\in\mathrm{TS}(1)\setminus\mathbf{bs}(1)\})} \\ &\Pi_{X\in\mathrm{TS}(1)}\ \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X\ 1 = \mathrm{fwd}_1(1)\ .\end{aligned}$$

$\mathrm{TS}(1)$ denotes the nodes of the first time slice, and $\mathbf{bs}(1)$ denotes the set of nodes that belong to the belief state of the first time slice. To simplify exposition (especially for the algorithm in the next column) we have introduced here the notation $\mathrm{fwd}_1(1)$ for $\mathrm{fwd}_{\text{init}}$.

The polynomial for the first time slice is therefore as follows:

$$\begin{aligned}&\mathcal{F}_{\mathrm{elim}(\{X\in\mathcal{N}\})}(\lambda_{X_i}, \boldsymbol{\theta}_{(X_i|\mathbf{pa}(X_i))}) = \\ &\quad\Sigma_{\mathrm{elim}(\mathbf{bs}(1))}(\mathrm{fwd}_{\text{init}}) = \Sigma_{\mathrm{elim}(\mathbf{bs}(1))}(\mathrm{fwd}_1(1)) = \\ &\quad\Sigma_{\mathrm{elim}(\mathbf{bs}(1))}(\Sigma_{\mathrm{elim}(\{X|X\in\mathrm{TS}(1)\setminus\mathbf{bs}(1)\})}\ \Pi_{X\in\mathrm{TS}(1)}\ \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X\ 1)\ .\end{aligned}$$

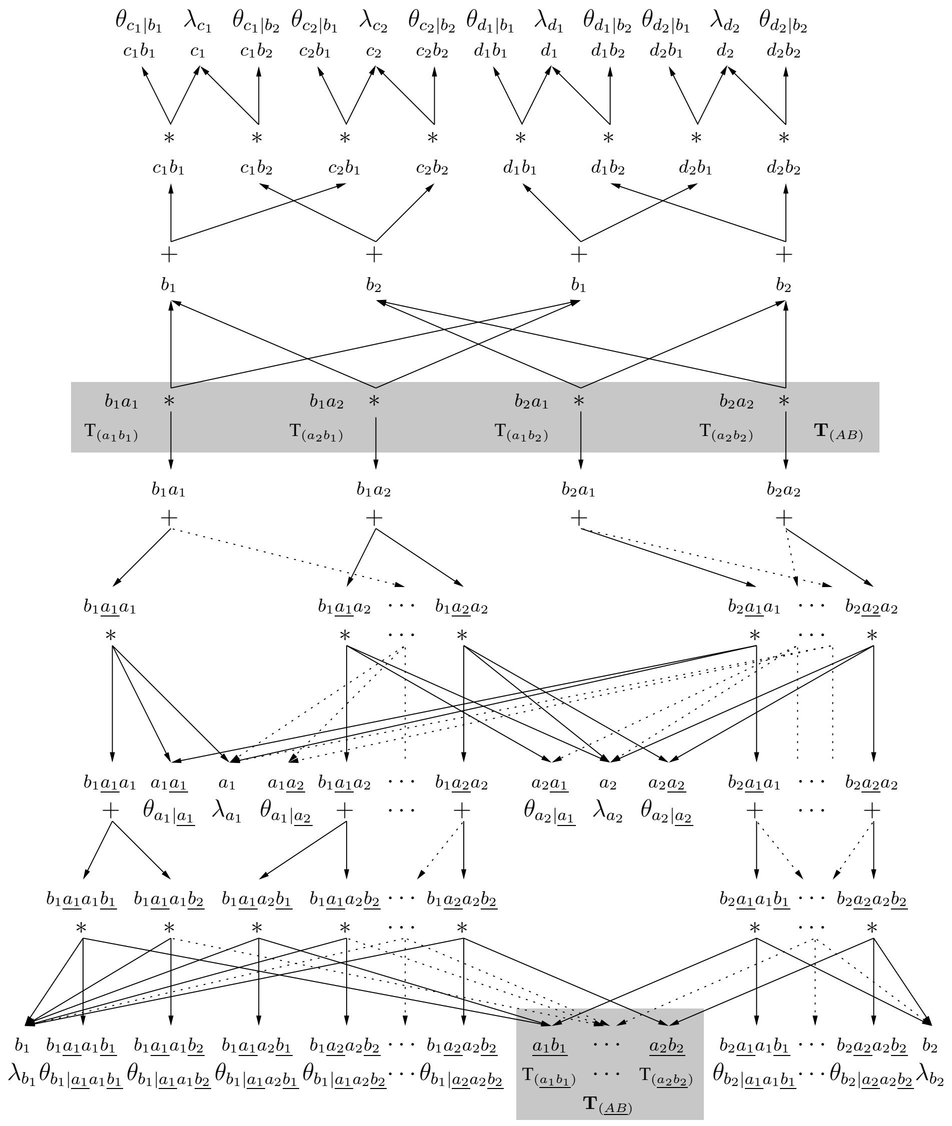


Figure 6: An arithmetic circuit that represents the procedure for forward propagation for Time Slice Schema 2 in Figure 4. (Hypotheses and nodes from the preceding time slice are underlined. The tables $\mathbf{T}_{(\underline{AB})}$ and $\mathbf{T}_{(AB)}$ are highlighted with a gray background.)

Now let us look at the case in which a time slice is to be added to the existing DBN. The situation is different from the one for the first time slice: The table over all nodes that were not marginalized out by the procedure that was just applied is now passed on to the

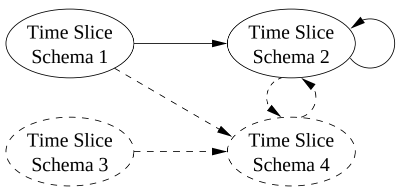


Figure 7: Directed graph that determines in which orders the time slice schemas can be instantiated.

general procedure for the ith time slice:

$$\begin{aligned}\mathrm{fwd}_i(\mathbf{T}) = {} & \Sigma_{\mathrm{elim}(\{X \mid X \in \mathrm{TS}(i) \setminus \mathbf{bs}(i)\})} \\ & \Sigma_{\mathrm{elim}(\{X \mid X \in \mathbf{bs}(i-1)\})} \Pi_{X \in \mathrm{TS}(i)} \,\boldsymbol{\theta}_{(X \mid \mathbf{pa}(X))} \lambda_X \; \mathbf{T} \,.\end{aligned}$$

$\mathbf{T}$ is a table over the set of nodes $\{X \mid X \in \mathbf{bs}(i-1)\}$, which in general was computed by the procedure of the preceding time slice. The result of the function fwd_i is a table over the set of nodes $\{X \mid X \in \mathbf{bs}(i)\}$.

The polynomial for Time Slices 1 through L is as follows:

$$\begin{aligned}\mathcal{F}_{\mathrm{elim}(\{X \in \mathcal{N}\})}(\lambda_{X_i}, \boldsymbol{\theta}_{(X_i \mid \mathbf{pa}(X_i))}) = {} & \Sigma_{\mathrm{elim}(\mathbf{bs}(L))}(\mathrm{fwd}_L(\ldots \mathrm{fwd}_2(\mathrm{fwd}_1(1)) \ldots)) = \\ & \Sigma_{\mathrm{elim}(\mathbf{bs}(L))}(\mathrm{fwd}_L \circ \ldots \circ \mathrm{fwd}_2 \circ \mathrm{fwd}_1(1)) \,.\end{aligned}$$

The evaluation of the polynomial with constant space requirements for the time slice L is obtained as follows:

1. $\mathbf{T} = 1$;
2. For $i = 1$ to L: $\quad \mathbf{T}_{\text{new}} = \mathrm{fwd}_i(\mathbf{T})$; $\quad \mathbf{T} = \mathbf{T}_{\text{new}}$;
3. $\mathcal{F}(\mathbf{e}_1, \ldots, \mathbf{e}_L)$ = $\Sigma_{\mathrm{elim}(\mathbf{bs}(L))}(\mathbf{T})$;

In Step 1, for the initial time slice, the value 1 is passed on to the table $\mathbf{T}$. In Step 2, a loop is traversed with the index running from 1 to L. In the first iteration, a new table $\mathbf{T}_{\text{new}}$ is computed by the initial time slice, that is $\mathbf{T}_{\text{new}} = \mathrm{fwd}_1(1)$. In the ith iteration, $\mathbf{T}_{\text{new}} = \mathrm{fwd}_i(\mathbf{T})$ is computed, which takes into account the effects of the earlier time slice. In the computation within the loop, already existing variables are reused efficiently. In particular, for the determination of $\mathrm{fwd}_i(\mathbf{T})$, the corresponding general procedures are applied. Consequently, in a subsequent pass through the loop, no further memory allocation is performed if all evidence vectors of the current time slice have been given numerical values. Finally, in Step 3 the polynomial is computed.

The polynomial $\mathcal{F}(\mathbf{e}_1, \ldots, \mathbf{e}_L)$ can be used for the computation of the belief values of the nodes in time slice t, as well as for other computations, such as sensitivity analyses, as was mentioned briefly in Section 2.1 and as is discussed in more detail by Darwiche (2000).

The procedure can also be generalized to DBNs in which more than one time slice schema is available for the instantiation of each time slice. A directed graph can be used to model the dependencies among the time slice schemas that determine the orders that they can be instantiated in, so that the number of procedures can be minimized. In Figure 7 the solid lines indicate the directed graph that models the dependencies among the time slice schemas (see Figure 4) for the example DBN (see Figure 3). The whole directed graph (with dashed and solid lines) in Figure 7 comprises two time slice schemas (1 and 3) that can be instantiated as the initial time slice of the DBN and two other time slice schemas (2 and 4) that can be instantiated as following time slices of the DBN. After Time Slice Schema 1, either Time Slice Schema 2 or Time Slice Schema 4 can be instantiated as the following time slice. After Time Slice Schema 3, only Time Slice Schema 4 can be instantiated as the following time slice.

The number of general procedures for a time slice schema depends on the number of possible preceding and succeeding time slice schemas.

In the following, $\text{strct}(i)$ denotes the nodes of time slice schema i and $\text{tss}(t)$ denotes those of the time slice schema that was instantiated at time t. In our example DBN (see Figure 4), we have $\text{tss}(1) = \text{strct}(1)$ and $\text{tss}(t) = \text{strct}(2)\ \forall t \geq 2$. The belief state depends not only on the current time slice but also on the succeeding time slice. So $\mathbf{bs}(\text{tss}(i), \text{tss}(i+1))$ denotes the set of nodes that belong to the belief state of time slice i when the succeeding time slice is $t+1$, while $\mathbf{bs}(\text{strct}(i), \text{strct}(j))$ denotes the set of nodes that belong to the belief state of time slice schema i when the succeeding time slice schema is j.

The general procedure for forward propagation for a time slice schema that instantiates the first time slice is therefore as follows:

$$\text{fwd}_{\text{tss}(1),\text{tss}(2)} = \Sigma_{\text{elim}(\{X \mid X \in \text{tss}(1) \setminus \mathbf{bs}(\text{tss}(1),\text{tss}(2))\})}\ \Pi_{X \in \text{tss}(1)} \theta_{(X \mid \mathbf{pa}(X))} * \lambda_X \ .$$

The general procedures for Time Slice Schema 1 in Figure 7 are $\text{fwd}_{\text{strct}(1),\text{strct}(2)}$ and $\text{fwd}_{\text{strct}(1),\text{strct}(4)}$. The procedure for Time Slice Schema 3 is $\text{fwd}_{\text{strct}(3),\text{strct}(4)}$.

Now let us look at the case in which a time slice is to be added to the existing DBN. The table over all nodes that were not marginalized out by the procedure that was just applied is now passed on to the general procedure for the ith time slice:

$$\begin{aligned}
&\text{fwd}_{\text{tss}(i-1) \to \text{tss}(i),\text{tss}(i+1)}(\mathbf{T}) = \\
&\quad \Sigma_{\text{elim}(\{X \mid X \in \text{tss}(i) \setminus \mathbf{bs}(\text{tss}(i),\text{tss}(i+1))\})} \\
&\quad \Sigma_{\text{elim}(\{X \mid X \in \mathbf{bs}(\text{tss}(i-1),\text{tss}(i))\})}\ \Pi_{X \in \text{tss}(i)} \theta_{(X \mid \mathbf{pa}(X))} * \lambda_X * \mathbf{T} \ .
\end{aligned}$$

$\mathbf{T}$ is a table over the set of nodes $\{X \mid X \in \mathbf{bs}(\text{tss}(i-1), \text{tss}(i))\}$, which in general was computed by the procedure of the preceding time slice. The result of the function fwd is a table over the set of nodes $\{X \mid X \in \mathbf{bs}(\text{tss}(i), \text{tss}(i+1))\}$. The general procedures for Time Slice Schema 2 in Figure 7 are $\text{fwd}_{\text{strct}(1) \to \text{strct}(2),\text{strct}(2)}$, $\text{fwd}_{\text{strct}(1) \to \text{strct}(2),\text{strct}(4)}$, $\text{fwd}_{\text{strct}(2) \to \text{strct}(2),\text{strct}(2)}$, $\ldots$, $\text{fwd}_{\text{strct}(4) \to \text{strct}(2),\text{strct}(4)}$, and the general procedures for Time Slice Schema 4 in Figure 7 are $\text{fwd}_{\text{strct}(1) \to \text{strct}(4),\text{strct}(2)}$, $\text{fwd}_{\text{strct}(2) \to \text{strct}(4),\text{strct}(2)}$ and $\text{fwd}_{\text{strct}(3) \to \text{strct}(4),\text{strct}(2)}$.

Again, to simplify exposition (especially for the following algorithm in 3.3) we introduce here the notation $\text{fwd}_{\text{strct}(0) \to \text{strct}(1),\text{strct}(2)}(1)$ for $\text{fwd}_{\text{tss}(1),\text{tss}(2)}$.

The polynomial for Time Slices 1 through L (with $\text{tss}(0) = \emptyset$) is therefore as follows:

$$\begin{aligned}\mathcal{F}_{\text{elim}(\{X \in \mathcal{N}\})}(\lambda_{x_i}, \theta_{(x_i|\mathbf{pa}(x_i))}) = \Sigma_{\text{elim}(\mathbf{bs}(\text{tss}(L),\text{tss}(L+1)))}(&\\ \text{fwd}_{\text{tss}(L-1)\to\text{tss}(L),\text{tss}(L+1)}(\ldots&\\ \text{fwd}_{\text{tss}(1)\to\text{tss}(2),\text{tss}(3)}(&\\ \text{fwd}_{\text{tss}(0)\to\text{tss}(1),\text{tss}(2)}(1))\ldots))&\ .\end{aligned}$$

3.2 Backward Propagation

With the procedures for forward propagation, the impact of evidence in new time slices can be transported to older time slices. First, an upward pass has to be performed from the older time slice to the new one whose evidence is to be taken into account; then, a downward pass from the newer time slice to the older one is required. Time slices can be rolled up only after the downward pass. Hence, a constant-space evaluation with only the general procedures for forward propagation is not possible.

Let us look at Time Slices 2 and 3 in Figure 3. We obtain as a procedure for backward propagation for the last time slice:

$$\begin{aligned}\text{bwd}_{\text{tss}(3)\to\text{tss}(2)} = {}& \Sigma_{A_3,B_3}\boldsymbol{\theta}_{(A_3|A_2)}\lambda_{A_3}\ \boldsymbol{\theta}_{(B_3|A_3,A_2,B_2)}\lambda_{B_3}\\ & (\Sigma_{C_3}\ \boldsymbol{\theta}_{(C_3|B_3)}\lambda_{C_3})\ (\Sigma_{D_3}\ \boldsymbol{\theta}_{(D_3|B_3)}\lambda_{D_3})\ .\end{aligned}$$

In contrast to the general procedures for forward propagtion, we marginalize out all of the nodes that belong to the time slice in focus. The result is a table $\mathbf{T}$ over the Cartesian product of the hypotheses of the parent nodes that belong to the preceding time slice—in our example, the nodes A_2 and B_2.

Now we want to specify a procedure which takes into account the effects of the following time slice on the preceding time slice. As was the case with the procedure for the last time slice, we can marginalize out all of the nodes that belong to the time slice in focus, but to keep demands on computation time and memory as low as possible, we first marginalize out the nodes that do not belong to the belief state of this time slice, that is $\mathbf{bs}(\text{tss}(2), \text{tss}(3))$, then we multiply the table $\mathbf{T}$ with the nodes of the belief state of this time slice and then we marginalize out the nodes of the belief state of this time slice.

We obtain the following as a procedure for backward propagation for the second time slice in our example in Figure 3:

$$\begin{aligned}&\text{bwd}_{\text{tss}(3),\text{tss}(2)\to\text{tss}(1)}(\mathbf{T}_{(A_2B_2)}) =\\ &(\Sigma_{A_2}\ \boldsymbol{\theta}_{(A_2|A_1)}\lambda_{A_2}\ (\Sigma_{B_2}\ \boldsymbol{\theta}_{(B_2|A_2,A_1,B_1)}\lambda_{B_2}\ \mathbf{T}_{(A_2B_2)}\\ &\quad(\Sigma_{C_2}\ \boldsymbol{\theta}_{(C_2|B_2)}\lambda_{C_2})\ (\Sigma_{D_2}\ \boldsymbol{\theta}_{(D_2|B_2)}\lambda_{D_2}))) = \mathbf{T}_{(A_1B_1)}\ .\end{aligned}$$

Now we want to capture the results obtained so far in a general notation. We omit the general procedure for backward propagation from the last time slice, which is only a special case of the general procedure for backward propagation from the ith time slice ($1 \leq i \leq L$):

$$\begin{aligned}&\text{bwd}_{\text{tss}(i+1),\text{tss}(i)\to\text{tss}(i-1)}(\mathbf{T}) =\\ &\ \Sigma_{\text{elim}(\{X|X\in\mathbf{bs}(\text{tss}(i),\text{tss}(i+1))\})}\ \Pi_{X\in\mathbf{bs}(\text{tss}(i),\text{tss}(i+1))}\ \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X\ \mathbf{T}\\ &\quad(\Sigma_{\text{elim}(\{X|X\in\text{tss}(i)\setminus\mathbf{bs}(\text{tss}(i),\text{tss}(i+1))\})}\ \Pi_{X\in\text{tss}(i)\setminus\mathbf{bs}(\text{tss}(i),\text{tss}(i+1))}\ \boldsymbol{\theta}_{(X|\mathbf{pa}(X))}\lambda_X)\ .\end{aligned}$$

The following hold:

- in the case $i = L$: tss$(L+1) = \emptyset$ and $\mathbf{T} = \mathbf{1}$;
- in the case $i = 1$: tss$(0) = \emptyset$.

In the next subsection we will show how forward and backward propagation can be combined.

3.3 Combined Forward and Backward Propagation

If we are interested in the belief values of nodes in time slice t of a DBN from the Time Slices 1 through L (with $1 \leq t \leq L$), we can combine the procedures for forward and backward propagation: We multiply the procedures for the polynomial for forward propagation from Time Slice 1 through t with the procedures for the polynomial for backward propagation from Time Slices L through $t+1$. Then we marginalize out the nodes $\mathbf{bs}(\text{tss}(t), \text{tss}(t+1))$.

The polynomial to be evaluated at time t with evidence $\mathbf{e}_1, \ldots, \mathbf{e}_t, \ldots, \mathbf{e}_L$ is as follows:

$$\begin{aligned}\mathcal{F}_t(\mathbf{e}_1, \ldots, \mathbf{e}_t, \ldots, \mathbf{e}_L) = \Sigma_{\text{elim}(\mathbf{bs}(\text{tss}(t),\text{tss}(t+1)))}\big(&\\ &\text{fwd}_{\text{tss}(t-1)\to\text{tss}(t),\text{tss}(t+1)}\big(\\ &\quad\text{fwd}_{\text{tss}(t-2)\to\text{tss}(t-1),\text{tss}(t)}\big(\cdots\\ &\qquad\text{fwd}_{\text{tss}(0)\to\text{tss}(1),\text{tss}(2)}(\mathbf{1})\ldots\big)\big)\\ &\text{bwd}_{\text{tss}(t+2),\text{tss}(t+1)\to\text{tss}(t)}\big(\\ &\quad\text{bwd}_{\text{tss}(t+3),\text{tss}(t+2)\to\text{tss}(t+1)}\big(\cdots\\ &\qquad\text{bwd}_{\text{tss}(L+1),\text{tss}(L)\to\text{tss}(L-1)}(\mathbf{1})\ldots\big)\big)\big)\ .\end{aligned}$$

The evaluation of the polynomial with constant space requirements for the time slice t is obtained as follows:

1. $\mathbf{F} = 1$;
2. For $i = 1$ to t: $\mathbf{F}_{\text{neu}} = \text{fwd}_{\text{tss}(i-1)\to\text{tss}(i),\text{tss}(i+1)}(\mathbf{F})$; $\mathbf{F} = \mathbf{F}_{\text{neu}}$;
3. $\mathbf{B} = \mathbf{1}$;
4. For $i = L$ downto $t+1$: $\mathbf{B}_{\text{neu}} = \text{bwd}_{\text{tss}(i+1),\text{tss}(i)\to\text{tss}(i-1)}(\mathbf{B})$; $\mathbf{B} = \mathbf{B}_{\text{neu}}$;
5. $\mathcal{F}_t(\mathbf{e}_1, \ldots, \mathbf{e}_t, \ldots, \mathbf{e}_L) = \Sigma_{\text{elim}(\mathbf{bs}(\text{tss}(t),\text{tss}(t+1)))}(\mathbf{F} * \mathbf{B})$;

Here the same remarks hold as for the preceding algorithm. We now call the table $\mathbf{F}$ in the case of forward propagation and $\mathbf{B}$ in the case of backward propagation.

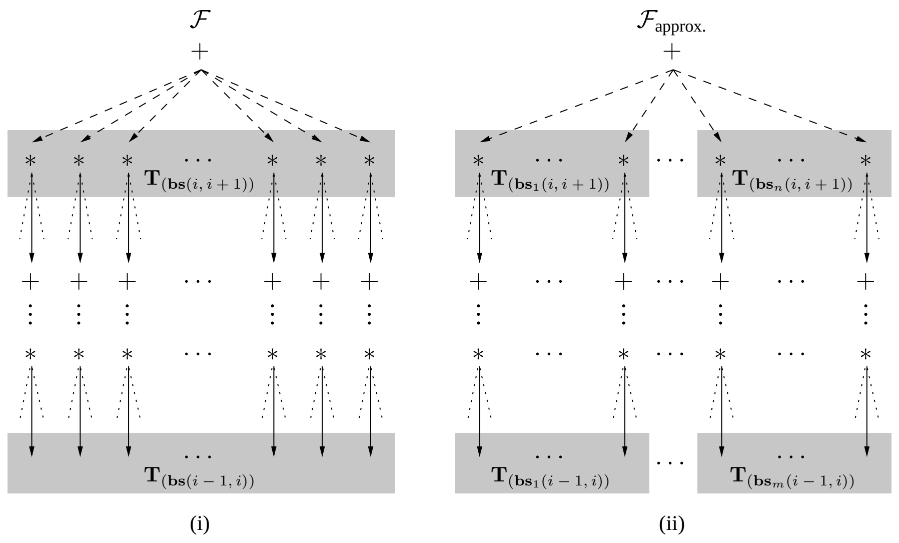

$\mathcal{F}$
$\mathcal{F}_{\text{approx.}}$
$\mathbf{T}_{(\mathbf{bs}(i,\,i+1))}$
$\mathbf{T}_{(\mathbf{bs}(i-1,\,i))}$
$\mathbf{T}_{(\mathbf{bs}_1(i,\,i+1))}$
$\mathbf{T}_{(\mathbf{bs}_n(i,\,i+1))}$
$\mathbf{T}_{(\mathbf{bs}_1(i-1,\,i))}$
$\mathbf{T}_{(\mathbf{bs}_m(i-1,\,i))}$
(i)
(ii)

Figure 8: Two arithmetic circuits that represent the procedure for forward propagation (solid and dotted lines) and the polynomial (dotted, dashed and solid lines)—with and without approximation, respectively. For the sake of clarity in the figure, the notation $\mathbf{bs}(\text{tss}(i-1), \text{tss}(i))$ has been replaced with $\mathbf{bs}(i-1, i)$.

3.4 Approximation

The procedures can be adapted so that we do not have to compute the table $\mathbf{T}$, which passes on all information without approximation. Instead, several smaller tables can be computed which are much less complex than the overall table $\mathbf{T}$ and whose product approximates the overall table. Boyen and Koller (1999) describe how the overall table can be split up into smaller tables with minimal loss of information. They specify several criteria (*weak interaction*, *conditional weak interaction* and *sparse interaction*) on the basis of which it is possible to partition the set of nodes into several smaller sets, so as to minimize the resulting information loss.

They "...show that the error in a belief state contracts exponentially as the process evolves. Thus, even with multiple approximations, the error in our process remains bounded indefinitely" (Boyen & Koller, 1998).

We will now show how this approximation can be incorporated into the proposed algorithm. Figure 8(i) shows schematically the general procedure for forward propagation in the ith time slice ($\text{fwd}_{\text{tss}(i-1)\to\text{tss}(i),\text{tss}(i+1)}$) without approximation. For the sake of greater clarity in the figure, the notation $\mathbf{bs}(\text{tss}(i-1), \text{tss}(i))$ is replaced with the shorter notation $\mathbf{bs}(i-1, i)$, which we will continue to use in the following. The table $\mathbf{T}_{(\mathbf{bs}(i-1,\,i))}$ was computed via the general procedure for forward propagation for the previous time slice. This table now serves as input to the general procedure for forward propagation in the ith time slice, which yields the table $\mathbf{T}_{(\mathbf{bs}(i,\,i+1))}$ as a result.

Figure 8(ii) shows the general procedure for forward propagation with approximation in the ith time slice. The table $\mathbf{T}_{(\mathbf{bs}(i-1,\,i))}$ has been split up into the tables

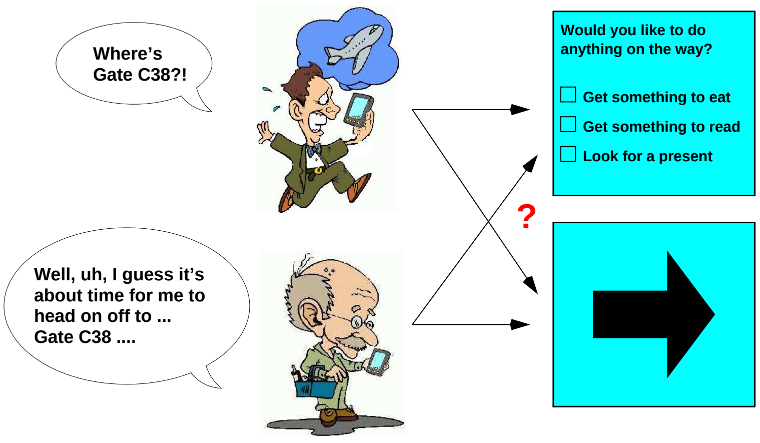

Where's Gate C38?!
Well, uh, I guess it's about time for me to head on off to ... Gate C38 ....
Would you like to do anything on the way?
☐ Get something to eat
☐ Get something to read
☐ Look for a present
?

Figure 9: Example of how the system READY adapts its behavior to the user's perceived resource limitations.

$\mathbf{T}_{(\mathbf{bs}_1(i-1,i))}, \ldots, \mathbf{T}_{(\mathbf{bs}_m(i-1,i))}$, which serve as input to the general procedure with approximation. This procedure yields as a result the table $\mathbf{T}_{(\mathbf{bs}_1(i,i+1))}, \ldots, \mathbf{T}_{(\mathbf{bs}_n(i,i+1))}$.

The approximation method illustrated here for forward propagation can be applied analogously for backward propagation. The approximate procedures require less memory and less computation; both of these advantages speed up the overall inference process.

4 Illustration of Practical Applicability

4.1 Example Application Domain

One promising aspect of the method presented above is the possibility of optimizing DBNs for resource-limited devices. In this section, we will show how this process can work in practice. We first briefly introduce the prototype mobile dialog system READY, whose requirements originally inspired the development of our method. We will then discuss how the properties of our method for handling DBNs can be exploited in the context of a system like READY.

Our overall goal is to have a mobile assistance system present information to a user in a way that is adapted to his or her current time pressure and cognitive load (cf. Bohnenberger, Brandherm, Großmann-Hutter, Heckmann, & Wittig, 2002). READY assesses these resource limitations probabilistically on the basis of *symptoms* in the user's behavior, as well as on the basis of physiological signals.

The nature of the assistance provided by READY can best be explained with a concrete example. Figure 9 shows two travelers at a large international airport. The first traveler is experiencing both time pressure and distraction because of the impending departure of his flight. For him, the second presentation shown in the figure is probably suitable. Since the second traveler seems to have a lot of time and attention available, the first presentation may be more suitable. While directing this second passenger to the

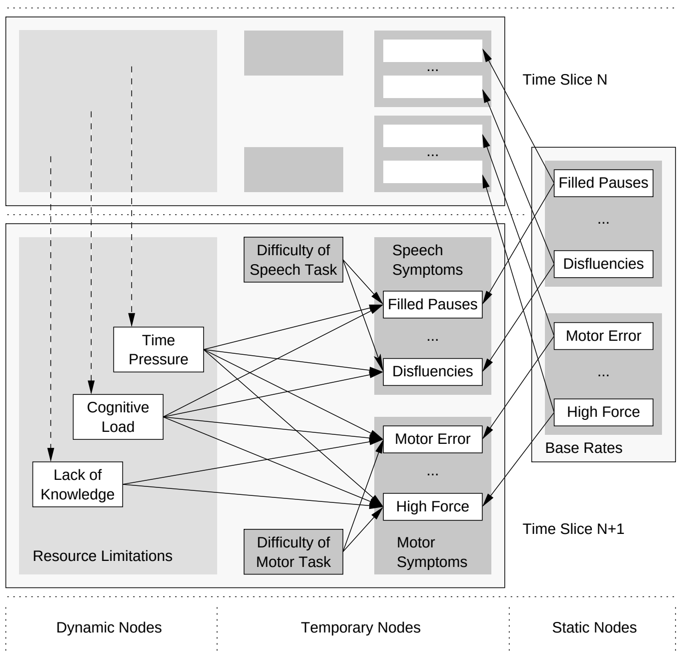


Figure 10: Overview of a dynamic Bayesian network for the recognition of a user's resource limitations. (Explanation in text.)

gate, the system can lead him past selected shops, so that he can get some useful things done on the way to the gate.

In this example, the differences between the two travelers can be recognized mainly on the basis of features of their speech such as the length of the utterances, the presence of pauses, and the rate of articulation.

The first person is speaking fast, loud and tersely. These are indicators of both time pressure and cognitive load. In the READY project, several BNs have been developed for the assessment of time pressure and cognitive load. These BNs serve as time slices for the DBN. A BN for the interpretation of speech symptoms was learned on the basis of two experiments (see Müller, Großmann-Hutter, Jameson, Rummer, & Wittig, 2001; Kiefer, 2002), while another BN for the interpretation of features of manual input behavior was constructed on the basis of a literature study (see Lindmark, 2000). The combination of these two BNs allows READY to make inferences on the basis of multimodal imput. Other BNs to handle data from an eye tracker and from physiological sensors are currently being developed. Figure 10 shows two time slices of a DBN that handles speech and motor symptoms. In the current version of the system, a time slice typically comprises about 40 nodes, each of which has 2, 3, or 4 states.

Let us look more closely at the nodes in time slice $N + 1$. The nodes Cognitive Load and Time Pressure model the resource limitations of the user. These nodes are realized as *dynamic nodes*, because they have an impact on the next slice. These user properties vary over time and cannot be observed directly. They can be estimated on the basis of symptoms in the behavior of the user, such as pauses in speech or the use of high force in tapping on the screen. The nodes that represent symptoms in behavior are *temporary nodes*. The *static nodes* shown in the figure are *base rates* of the user—for example, how often this user typically produced filled pauses (e.g., *uhm*). These properties do not change over time, but they are estimated with increasing accuracy as more information arrives. They have the same impact on each time slice.

Evidence of the types shown in the figure is acquired each time the user produces a spoken utterance or provides manual input. A greater processing challenge is raised by data from physiological sensors, which yield signals concerning variables like the user's heart beat, skin conductance, and muscle tension.

Data of ths type is typically acquired at a high rate. Instead of being used directly for instantiation of variables in the DBN, the signals must first be subjected to preprocessing that yields somewhat higher-level data. For example, preprocessing of the electrocardiographic signal yields a measure of heart rate. But even these higher-level variables can require a potentially large number of instantiations of the corresponding temporary nodes of the DBN. Therefore, we need a fast inference algorithm for the solution of the DBNs. And if we want to evaluate a DBN fast on a PDA, we have to adapt the computation to the limitations of the PDA (e.g., with respect to computation time and memory).

Ramos, Cozman, and Ide (2002) present an algorithm for the processing of static BNs on a PDA that can deal with dynamically changing limitations in the availability of the resources time and memory. The approach involves the combination of various algorithms that operate on different parts of the network, in a way that depends on the availability of resources. These algorithms exhibit anytime and anyspace behavior.

By contrast, our procedure does not exhibit anytime or anyspace behavior. The DBN has to be adapted in advance to the resource limitations of the PDA, so as to ensure as far as possible that the inference can be performed within the limits imposed by the device. In the next subsection, we will show how such adaptation can be achieved with the methods presented in this paper.

4.2 Exploiting the Extended Approach

Figure 11 gives an overview of the various types of computation that need to be performed in a system like READY. Two types of computation can be distinguished: those that are performed offline on a PC and those that are performed online on a PDA used by the user.

Let us consider first the offline computations. If we have a set of time slice schemas, the dependencies among them, and the set of possible evidence and query nodes, we can compute a set of raw polynomials with the polynomial generation algorithm that was presented above. In READY, the set of time slice schemas would comprise the Bayesian networks for the recognition of time pressure and cognitive load on the basis of speech, manual input behavior, and physiological signals. The dependencies among the time slice schemas are represented in a directed graph like the one in Figure 7, which determines the orders in which the time slice schemas can be instantiated.

In the airport scenario introduced above, we are interested only in the belief values of the dynamic nodes Cognitive Load and Time Pressure. Therefore, all that is needed is an upward pass in the corresponding arithmetic circuit and the computation of the table

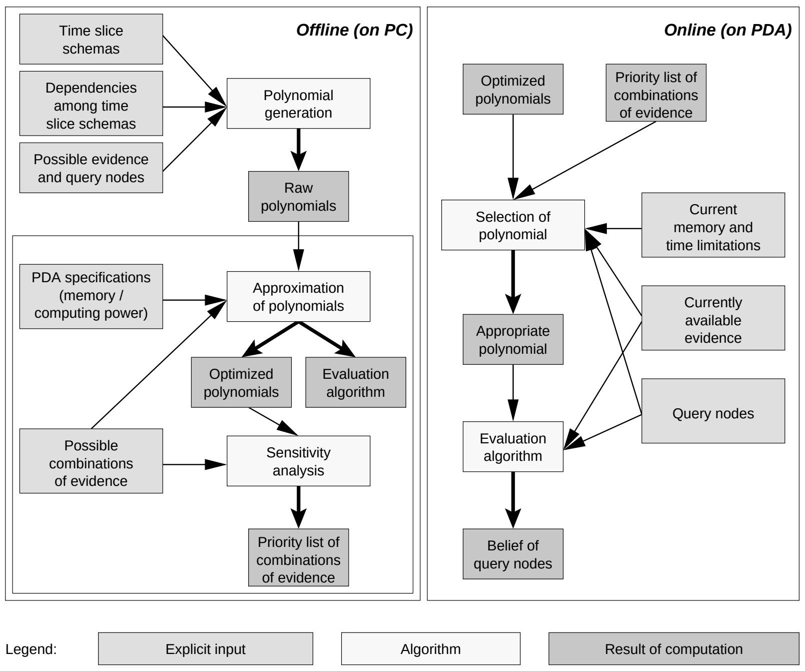

Offline (on PC)
Time slice schemas
Dependencies among time slice schemas
Possible evidence and query nodes
Polynomial generation
Raw polynomials
PDA specifications (memory / computing power)
Approximation of polynomials
Optimized polynomials
Evaluation algorithm
Possible combinations of evidence
Sensitivity analysis
Priority list of combinations of evidence
Online (on PDA)
Optimized polynomials
Priority list of combinations of evidence
Selection of polynomial
Current memory and time limitations
Currently available evidence
Appropriate polynomial
Query nodes
Evaluation algorithm
Belief of query nodes
Legend:
Explicit input
Algorithm
Result of computation

Figure 11: Online and offline computations for the resource-limited processing of DBNs.

T. From the table entries, the desired belief values can be obtained. In this case, the downward pass can be omitted.

In a second step in the offline computations, the set of raw polynomials can be adapted to the resource limitations of a PDA through approximation. The relevant properties of the PDA must have been entered manually, for example on the basis of the manufacturer's specifications. We obtain a set of optimized polynomials and an evaluation algorithm. For the set of optimized polynomials, the system can perform sensitivity analyses[3] to compute a priority list of combinations of pieces of evidence ordered by their impact on particular query nodes: We want to avoid the instantiation of combinations of evidence that will have little or no impact.

The set of optimized polynomials, the priority list, and the evaluation algorithm are then passed to the PDA. Depending on the current memory and time limitations, the currently available evidence, and the selected query nodes, an appropriate polynomial is selected out of the set of optimized polynomials by the selection algorithm.

Suppose that, during the user's interaction with the PDA, more evidence is received within a given time interval than the DBN is capable of processing. In this case, a subset of pieces of evidence should be used for instantiation; this subset should be the one that is likely to exert the greatest influence on the query nodes, according to the priority list. If this situation occurs frequently, the reason may be that the specification of the computing

[3]The possibility of performing sensitivity analyses of this sort is one of the general strengths of Darwiche's differential approach. The discussion of how the sensitivity analyses would look in this particular case would exceed the scope of this paper.

resources of the PDA was too optimistic; in this case, it may be better to create a revised specification of the resource limitations, which will in turn lead to a greater degree of approximation of the polynomials themselves.

During the interaction of the user with the PDA, it may be possible for the system to recognize typical features of the way in which the current user uses the device. For example, a given user may consistently avoid using speech recognition, restricting himself entirely to pen input. The next time the PDA is connected to the PC, it can transmit its accumulated knowledge about the user to the PC. The system on the PC can then update its list of possible combinations of evidence as well as its assessment of the PDA's resource limitations, and it can compute a new approximation of the polynomials as well as a new priority list of combinations of evidence. When the user starts using the PDA again, the system will have adapted itself better to the requirements and properties of the user.

5 Summary

We have shown how Darwiche's differential approach to the evaluation of Bayesian networks can be extended to dynamic Bayesian networks. We have specified the procedures that can be used to determine the relevant polynomials for arbitrarily large DBNs. Computations for partial polynomials can be reused.

Through the use of these formulas, we can perform forward and backward propagation (as well as a combination of the two). We can also roll up older time slices and other superfluous network structures while ensuring constant space requirements in the evaluation of the polynomials. The other advantages of the differential approach are now also available for DBNs, for example, efficient methods for sensitivity analysis (see Darwiche, 2000, 1999).

We have also described how the method presented can be used to adapt the processing of DBNs to the resource limitations of a mobile device.

Acknowledgements

This research is being supported by the German Science Foundation (DFG) in its Collaborative Research Center on Resource-Adaptive Cognitive Processes, SFB 378, Project EM 5, READY. We thank the three anonymous reviewers for helpful comments on an earlier version of the manuscript.

References

Bohnenberger, T., Brandherm, B., Großmann-Hutter, B., Heckmann, D., & Wittig, F. (2002). Empirically grounded decision-theoretic adaptation to situation-dependent resource limitations. *Künstliche Intelligenz*, *3*, 10–16.

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. In G. F. Cooper & S. Moral (Eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference* (pp. 33–42). San Francisco: Morgan Kaufmann.

Boyen, X., & Koller, D. (1999). Exploiting the architecture of dynamic systems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (pp. 313–320). Orlando, FL.

Darwiche, A. (1999). *A differential approach to inference in Bayesian networks* (Tech. Rep. No. D-108). Computer Science Department, UCLA.

Darwiche, A. (2000). A differential approach to inference in Bayesian networks. In C. Boutilier & M. Goldszmidt (Eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference* (pp. 123–132). San Francisco: Morgan Kaufmann.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the Association for Computing Machinery*, *50*(3), 280–305.

Jensen, F. V. (2001). *Bayesian networks and decision graphs.* New York: Springer.

Kiefer, J. (2002). *Auswirkungen von Ablenkung durch gehörte Sprache und eigene Handlungen auf die Sprachproduktion [Effects on speech production of distraction through overheard speech and one's own actions].* Unpublished master's thesis, Department of Psychology, Saarland University, Germany.

Kjærulff, U. (1995). dHugin: A computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, *11*, 89–111.

Lindmark, K. (2000). *Interpreting symptoms of cognitive load and time pressure in manual input.* Unpublished master's thesis, Department of Computer Science, Saarland University, Germany.

Müller, C., Großmann-Hutter, B., Jameson, A., Rummer, R., & Wittig, F. (2001). Recognizing time pressure and cognitive load on the basis of speech: An experimental study. In M. Bauer, P. Gmytrasiewicz, & J. Vassileva (Eds.), *UM2001, User Modeling: Proceedings of the Eighth International Conference* (pp. 24–33). Berlin: Springer.

Park, J. D., & Darwiche, A. (2002). A differential semantics for jointree algorithms. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems 15.* Cambridge, MA: MIT Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Ramos, F. T., Cozman, F. G., & Ide, J. S. (2002). Embedded Bayesian networks: Anyspace, anytime probabilistic inference. In *AAAI/KDD/UAI Workshop on Real-Time Decision Support and Diagnosis Systems* (pp. 13–19). Edmonton, Canada.