# Probabilistic Processing of Interval-valued Sensor Data

Sander Evers
University of Twente
The Netherlands
everss@cs.utwente.nl

Maarten M. Fokkinga
University of Twente
The Netherlands
fokkinga@cs.utwente.nl

Peter M.G. Apers
University of Twente
The Netherlands
p.m.g.apers@utwente.nl

## ABSTRACT

When dealing with sensors with different time resolutions, it is desirable to model a sensor reading as pertaining to a time interval rather than a unit of time. We introduce two variants on the Hidden Markov Model in which this is possible: a reading extends over an arbitrary number of hidden states. We derive inference algorithms for the models, and analyse their efficiency. For this, we introduce a new method: we start with an inefficient algorithm directly derived from the model, and visually optimize it using a *sum-factor diagram*.

## Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Statistical databases—*sensor data*

## General Terms

Algorithms, Theory, Performance

## Keywords

Intervals, Probabilistic Inference, Hidden Markov Model

## 1. INTRODUCTION

Consider the following sensor setup: at 15 fixed locations in a building, a Bluetooth transceiver ('scanner') is installed, and performs regular scans in order to track the position of a mobile device. The scanning range is such that the mobile device can be seen by 2 or 3 different scanners at most places. To estimate its position, a Hidden Markov Model (HMM) is used: this probabilistic model relates a sequence of observations (scan results) at times $t = 1..T$ to a sequence of hidden states (positions) at the same times. An advantage of a HMM is that it includes a *transition model*, in which domain knowledge can be encoded about possible position changes: for example, that it is impossible to move directly from room A to room B due to a wall[6].

A disadvantage of a HMM is that it synchronizes hidden states with observations. In the above setup, which we implemented in our lab, scans take about 10 seconds (it is unknown when in this interval the mobile device really responds) and are performed 4 times per minute. For several reasons, it is desirable to have a higher time resolution for the hidden states: (a) to encode higher-precision possible transitions, (b) to integrate readings from sensors with other time resolutions, and (c) to handle the situation where the 10-second scans are not synchronized with each other.

To accomodate a higher hidden time resolution within this probabilistic framework, we define a generalization of the HMM which we name HMM-AO (Aggregate Observations). In this model, each observation can stretch over an arbitrary number of hidden states, possibly overlapping with other observations. Furthermore, it is not required that observations occur at regular intervals, which provides a straightforward way to deal with missing sensor readings. We also define a restricted variant of HMM-AO named HMM-NOR (Noisy-OR) that is more specific to the above setup.

An important property of the HMM is that *inference* of the probability distributions over the hidden states can be done in linear time w.r.t. the sequence length (which enables *forward filtering* or *fixed-lag smoothing* in a streaming way[9]). We derive inference algorithms for our models for which this is also the case. For HMM-AO, however, the running time is exponential in the interval length, which renders it unsuitable for high time resolutions. The running time for HMM-NOR is independent of interval length.

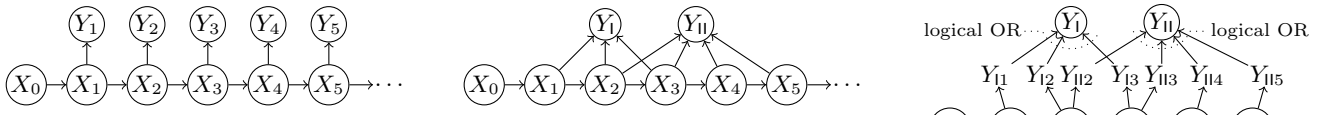
A substantial contribution of this article is a new method for deriving these inference algorithms: we start with a highly inefficient algorithm directly derived from the model's definition, and rewrite it for efficiency using a so-called *sum-factor diagram* and a set of visual transformation rules. At the moment, we have optimized the number of basic calculations, but we see potential for more complex cost models.

The rest of this article is organized as follows: in Sect. 2, we define the models; in Sect. 3, we explain inference with sum-factor diagrams, which we apply to our models in Sect. 4. Related work and conclusions are presented in Sect. 5 and 6.

## 2. HMM, HMM-AO & HMM-NOR MODELS

To do probabilistic sensor data processing, one needs a *probabilistic model* of the observed phenomenon and the sensing process. This model defines a number of stochastic variables (written in capitals, e.g. $X$,$Y$) and the relations between them. Every sensor reading is represented by an *observable* variable (also called *evidence variable*). For example, we define the variable $Y_t$ to represent the Bluetooth scan at time $t$. When the scan is performed and results in


*DMSN'08, August 24, 2008, Auckland, New Zealand*




(a) Standard HMM: observation $Y_t$ is synchronized with state $X_t$

(b) HMM-AO: observation $Y_\mathsf{i}$ stretches over an arbitrary interval $X_r, \ldots, X_s$. HMM-NOR: idem, but $Y_\mathsf{i}$ is binary and $\mathsf{P}(y_\mathsf{i}|x_{r..s})$ is restricted (see text)

(c) HMM-NOR with Noisy-OR structure elaborated using extra variables

Figure 1: The three different Hidden Markov Models drawn as a Bayesian network.

a value $y_t$, we have observed that $Y_t = y_t$. The other variables in the model, e.g. the location of the mobile device, are *unobservable* (also called *hidden*). Each variable has a well-defined domain (the set of values it can take); in this article, we only consider variables with a discrete, finite domain.

A probabilistic model is characterized by the *joint probability distribution* (jpd) that it defines over its variables. In a model with $n$ variables $X_1$–$X_n$, the jpd defines a probability $\mathsf{P}(X_1 = x_1, \ldots, X_n = x_n)$ for every possible combination of $x_i$ values. All information about the probabilistic relations between variables can be deduced from this jpd.

The most simple method of defining a probabilistic model is providing a table with all the values in the jpd. However, the size of this table grows exponentially with the number of variables. A popular alternative is to define the model by means of a Bayesian network[3], which is what we will do in this section.

## 2.1 Notation

Where a probabilistic event (e.g. $X{=}x,Y{=}y,Y{=}0$) is expected, we often omit the stochastic variable $(X, Y)$ and write only the concrete or abstract value $(x, y, 0)$. It should always be clear which variable we have omitted: values 0, 1 and $y$ always belong to a $Y$ variable, and value $x$ to an $X$ variable. When a value has a subscript, this transfers to the variable: $\mathsf{P}(y_1)$ means $\mathsf{P}(Y_1 = y_1)$. In case of the concrete values 0 and 1, the subscript only refers to the variable: $\mathsf{P}(1_t)$ means $\mathsf{P}(Y_t = 1)$.

In a summation, the range of the variable is omitted. We always sum over all possible values that the stochastic variable can take. The scope of a summation extends as far as possible: in $\sum_{x_1} f_1(x_1) \sum_{x_2} f_2(x_2)$, the first summation sums over the whole expression. For summing only over $f_1(x_1)$, we write $\left[\sum_{x_1} f_1(x_1)\right] \left[\sum_{x_2} f_2(x_2)\right]$.

In the context of a probabilistic event, dots in a subscript mean the conjunction of probabilistic events: $\mathsf{P}(x_{1..3})$ means $\mathsf{P}(X_1 = x_1, X_2 = x_2, X_3 = x_3)$. In a summation, they mean a nested summation over all variables: $\sum_{x_{1..3}} f(x_1, x_2, x_3)$ means $\sum_{x_1} \sum_{x_2} \sum_{x_3} f(x_1, x_2, x_3)$.

## 2.2 The three models as Bayesian networks

A Bayesian network consists of two components:

- A directed acyclic graph, in which the nodes represent the stochastic variables. The edges define a *parent relation* by pointing from parent to child; we write $\mathit{Parents}(X)$ for the set of parents of variable $X$.
- A conditional probability distribution (cpd) for each stochastic variable $X$, defining $\mathsf{P}(x|\mathit{parents}(X))$ for all possible values of $x$ and $\mathit{parents}(X)$. We use $\mathit{parents}(X)$ (notice the lowercase $p$) as an abbreviation for a set of abstract values for the stochastic variables $\mathit{Parents}(X)$. For example, if $\mathit{Parents}(X_{45}) = \{X_3, X_{22}, X_{29}\}$, then $\mathsf{P}(\mathit{parents}(X_{45}))$ means $\mathsf{P}(x_3, x_{22}, x_{29})$.

In a Bayesian network over $n$ variables $X_1$–$X_n$, the joint probability distribution over all the variables is given by

$$\mathsf{P}(x_{1..n}) = \prod_{i=1..n} \mathsf{P}(x_i|\mathit{parents}(X_i)) \qquad \text{(Fact-BN)}$$

Advantages of using a Bayesian network are:

- If the cpds are defined by tables of probabilities (which provides maximum generality), the number of parameters that define a model is exponential in the maximal number of *parents* that a variable has, rather than in the number of variables. It is also possible to define a cpd with less parameters, e.g. the cpd (Cpd-NOR) that we will shortly define.
- The fact that the jpd is *factorized* into $n$ factors speeds up inference (see Sect. 3.1).
- The directed graph provides an intuitive method for modelling cause-and-effect relations: if $X$ can directly cause $Y$ (with an amount of uncertainty), one draws an arrow from $X$ to $Y$. Formally, this translates into *conditional independence* properties of the model. For example, in Fig. 1a, the only direct cause of $Y_1$ is $X_1$. Knowing the value of $X_1$ is enough to fix the probability distribution over $Y_1$. Knowing the value of any other variable, e.g. $x_0$, does not result in another distribution: $\mathsf{P}(y_1|x_1, x_0) = \mathsf{P}(y_1|x_1)$. The conditional independence properties can be directly deduced from the graph; see Charniak's tutorial[3].

We now define the three different models as Bayesian networks: the regular HMM as a reference, followed by our extended models HMM-AO and HMM-NOR.

As shown in Fig. 1a, the regular HMM consists of a layer of observation variables $Y_t$ and a layer of state variables $X_t$. For every $X_t$, the cpd $\mathsf{P}(x_t|x_{t-1})$ is equal, and is called the *transition model*. Also, for every $Y_t$, the cpd $\mathsf{P}(y_t|x_t)$ is equal, and is called the *sensor model*. Initial state $X_0$ does not have any parents or an associated $Y_0$.

In the HMM with aggregate observations (HMM-AO), we replace the observation layer by a set of variables $Y_\mathsf{I}, Y_\mathsf{II}, \ldots$ indexed with Roman numerals, over which we range using $\mathsf{i}$ instead of $t$. Each $Y_\mathsf{i}$ is connected to an arbitrary number of subsequent state variables $X_r, \ldots, X_s$ (see Fig. 1b). Their cpds can all be different in principle.

The structure of the HMM with Noisy-OR aggregate observations (HMM-NOR) is equal to that of the HMM-AO, but its variables $Y_\mathsf{i}$ are restricted to binary variables (taking values 0 or 1), and their cpds to a 'Noisy-OR' form

$$\mathsf{P}(0_\mathsf{i}|x_{r..s}) = \prod_{t=r..s} f_{\mathsf{i}t}(x_t) \qquad \mathsf{P}(1_\mathsf{i}|x_{r..s}) = 1 - \prod_{t=r..s} f_{\mathsf{i}t}(x_t) \tag{CPD-NOR}$$

Again, the cpd for each observation $Y_\mathsf{i}$ can be different (hence the $\mathsf{i}$ subscript in $f_{\mathsf{i}t}$).

The cpds in the HMM-NOR are a slight generalization of the original Noisy-OR model[8]. To get this original model back, the input variables $x_t$ should be binary as well, with $f_{\mathsf{i}t}(0) = 1$ and $f_{\mathsf{i}t}(1) = q_{\mathsf{i}t}$ (where $q_{\mathsf{i}t}$ is the probability that a 1 on input $x_t$ is 'inhibited').

To better visualise the conditional independence structure of the Noisy-OR cpd, it is possible to split it up by introducing additional unobservable binary variables $Y_{\mathsf{i}r}, \ldots, Y_{\mathsf{i}s}$ in the model (see Fig. 1c). Their cpds are defined to be

$$\mathsf{P}(0_{\mathsf{i}t}|x_t) = f_{\mathsf{i}t}(x_t) \qquad \mathsf{P}(1_{\mathsf{i}t}|x_t) = 1 - f_{\mathsf{i}t}(x_t)$$

The cpd of $Y_\mathsf{i}$, now conditioned on these variables, is degenerate (the probabilities can only be 0 or 1), and represents the logical OR function: $\mathsf{P}(1_\mathsf{i}|y_{\mathsf{i}r..\mathsf{i}s}) = \bigvee_{t=r..s} y_{\mathsf{i}t}$. The reader may check that $\mathsf{P}(y_\mathsf{i}|x_{r..s})$ then equals (CPD-NOR).

Although the additional $Y_{\mathsf{i}t}$ variables clarify the model, we do not need them in deriving the inference algorithm.

## 2.3 Bluetooth localization with HMM-NOR

We will briefly explain how we have applied the HMM-NOR model to our Bluetooth localization task. Each $X_t$ variable represents the location of the mobile device at time $t$, where time is discretized into granules of 2.5 seconds. Space is also discretized into 50 locations, each of which correspond to an office or a section of the corridor. Each $Y_\mathsf{i}$ variable corresponds to one 10-second scan, and stretches over 4 subsequent $X$ variables (because there are 15 scanners that all produce 4 scans per minute, there is a lot of overlap). A *positive observation* $Y_\mathsf{i} = 1$ represents the event that the mobile device was detected during that scan, and a *negative observation* $Y_\mathsf{i} = 0$ represents the event that it was not.

The cpd $\mathsf{P}(x_t|x_{t-1})$ encodes the possible transitions that a mobile device can make within 2.5 seconds, and are estimated from the topology of the building and human walking speed. The cpd $\mathsf{P}(y_\mathsf{i}|x_{r..s})$ represents the probability that the mobile device has been scanned given its locations within the 10-second interval $[r, s]$. It is different for each of the 15 scanners, and can be estimated in the following way: the mobile device is kept in location $a$ for some time to record the number of detections by the scanner in question. The fraction of detections out of performed scans is denoted $p_{ca}$, and used as an estimate for the probability $\mathsf{P}(1_\mathsf{i}|X_r = \ldots = X_s = a)$ for scanner $c$.

We further make the assumption that $f_{\mathsf{i}t}$ is independent of the moment that the scan took place, and also of the moment $t$ within the interval. It is only dependent on the scanner $c$ that performed the scan. Therefore, $f_{\mathsf{i}r} = \ldots = f_{\mathsf{i}s} = f_c$. Then, using (CPD-NOR), we derive that $f_c(a) = \sqrt[4]{1 - p_{ca}}$. Repeating this procedure for every scanner-location combination gives us all the parameters of our sensor model.

Notice that HMM-NOR model has another advantage over HMM-AO except for efficient inference: given the above assumption that $f_{\mathsf{i}t}$ is the same for each moment $t$ within the interval, the number of parameters remains constant if we increase the time resolution. If we want to use 1-second granules, we simply use $f_c(a) = \sqrt[10]{1 - p_{ca}}$ to define our new sensor model (we do not address the question of how the transition model is affected here).

The price to pay for the HMM-NOR model are the conditional independence assumptions between the $Y_{\mathsf{i}t}$ variables. We do not elaborate on this now, but we would like to point out that these independence assumptions are similar to those in the original HMM model (which can also perform well in situations where these assumptions are not entirely faithful).

# 3. USING SUM-FACTOR EXPRESSIONS

## 3.1 Inference over a factored distribution

Given a probabilistic model over a set of variables, *inference* is the general task of deriving the probability distribution over a subset of *query variables*, given the observed values of another subset called the *evidence variables*. Formally, we partition the model's variables into $Q_{1..m}$, $E_{1..n}$ and (the remaining variables) $R_{1..p}$, and the goal of the inference task is to calculate $\mathsf{P}(q_{1..m}|e_{1..n})$ for all values $q_{1..m}$, given certain values $e_{1..n}$. Using basic probability axioms, this can be written in terms of the joint distribution (where $\alpha = 1/\mathsf{P}(e_{1..n})$):

$$\mathsf{P}(q_{1..m}|e_{1..n}) = \alpha\mathsf{P}(q_{1..m}, e_{1..n}) = \alpha \sum_{r_{1..p}} \mathsf{P}(q_{1..m}, e_{1..n}, r_{1..p}) \tag{INFERENCE}$$

As this probability is calculated for all values of $q_{1..m}$, and it is known that these will sum to 1, it is not necessary to calculate the constant factor $\alpha$. Thus, the inference task consists of repeatingly evaluating a summation over the $R_{1..p}$ dimensions.

In general, this summation is exponential in $p$. However, in the case of a *factored* joint distribution, it can be rewritten into a more efficient form. For a small example, assume we have a probability distribution over variables $A, B, C$ and $D$ which is factored as follows: $\mathsf{P}(a, b, c, d) = f_1(a, b)f_2(c, d)$. Now, let us take $A$ as the query variable and $D$ as the evidence variable. We can rewrite $\mathsf{P}(a|d)$ to:

$$\alpha \sum_{b,c} f_1(a, b)f_2(c, d) = \alpha \left[\sum_b f_1(a, b)\right] \left[\sum_c f_2(c, d)\right]$$

The two-dimensional summation is split into a product of one-dimensional summations. The same principle can be applied for a greater number of $R_{1..p}$ variables. The complexity of the inference task will be linear in $p$ if the distribution is always sufficiently factorized (i.e. the number of variables in a factor is limited independently of $p$).

However, the above example is simplified by the fact that there is no overlap between the factors. Usually this is not the case. For example, the joint probability distribution can be factored like:

$$\mathsf{P}(a, b, c, d) = f_1(a)f_2(a, b)f_3(b, c)f_4(c, d)$$

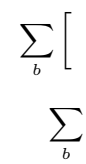
In this case, the summation expressions cannot be factored into a product, but can take different nested forms. If we do a summation over all the variables in the above distribution (for clarity of exposition; it does not make sense as an

inference task), several of these forms are:

$$\sum_{a,b,c,d} f_1(a) f_2(a,b) f_3(b,c) f_4(c,d) \tag{1a}$$

$$\sum_a f_1(a) \sum_b f_2(a,b) \sum_c f_3(b,c) \sum_d f_4(c,d) \tag{1b}$$

$$\sum_{c,d} f_4(c,d) \sum_b f_3(b,c) \sum_a f_2(a,b) f_1(a) \tag{1c}$$



$$\sum_b \left[\sum_a f_2(a,b) f_1(a)\right] \left[\sum_c f_3(b,c) \sum_d f_4(c,d)\right] \tag{1d}$$

Shown on the left are the corresponding sum-factor diagrams, which are explained in the next section. Exprs. (1b)–(1d) all represent more efficient ways to calculate the summation than Expr. (1a).

## 3.2 Sum-factor diagrams

For the models described in section 2, the summation expressions can already get quite complex. As a tool for visualizing their structure and efficiency, and for rewriting them into a more efficient form, we introduce the *sum-factor diagram*. This diagram can represent an expression of the form

$$\sum\cdot\sum f\cdot f \sum\cdot\sum f\cdot f \;\cdots\; \sum\cdot\sum f\cdot f,$$

i.e. an arbitrary permutation of factors and summations over the free variables of the factors, with the scope of each summation extending to the end of the expression. It is not allowed to introduce the same variable name twice, so $\sum_a f_1(a,b) \sum_a f_2(c,a)$ and $f_1(a,b) \sum_a f_2(c,a)$ are illegal. This is a purely syntactic restriction, and can be avoided by using $\sum_a f_1(a,b) \sum_d f_2(c,d)$ and $f_1(a,b) \sum_d f_2(c,d)$ instead. (To avoid all confusion: using the same variable in two different factors, e.g. $\sum_a f_1(a,b) f_2(c,a)$ is legal.) We call this form a *sum-factor expression*. The expressions (1a)–(1c) are sum-factor expressions; expression (1d) is not, but both factors on its top level are. The corresponding diagrams are shown next to the expressions.

On the diagram's horizontal axis (not labeled) are the factors, in the order in which they appear from left to right in the expression. On the vertical axis are the free variables of all the factors, in an arbitrary order. A dot in the diagram indicates that a variable appears in a factor. For example, for Expr. 1c, the leftmost column of its diagram shows that the leftmost factor in the expression, $f_4(c,d)$, contains the free variables $c$ and $d$.

In a sum-factor expression, a summation occurs between factors, and so it does in the diagram; it is indicated by a vertical stroke, in the row corresponding to the variable over which it sums. If there are multiple summations between two factors, their order is considered irrelevant, and all the strokes are at the same horizontal position (see Expr. 1a).

The grey color is auxiliary, and is derived from the dots and strokes. Reading a row from left to right, the grey bar starts at the position of the summation over that row's variable, or at the left margin if there is no summation. The bar ends at the rightmost position where the variable occurs in a factor. When read from right to left, the grey bar corresponds to the lifetime of the variable in the evaluation of the expression. The larger the grey area, the more inefficient the evaluation (see Sect. 3.3).

The expressions (1a)–(1d) calculate the same value, and can be transformed into each other due to distributivity $x(y+z) = xy + xz$. To ease these transformations, we define four derived rewrite rules which map more directly to sum-factor expressions. Using these rules, we can rewrite an expression *visually*, by manipulating its diagram.

In these rules, we use the notation $\sum_R f$, which is a multi-dimensional summation over the variables in the (possibly empty) set $R$. For example, if $R = \{a,b\}$, then $\sum_R f = \sum_{a,b} f = \sum_{b,a} f$ (the order of the summations is irrelevant). In the notation $\sum_{R,v}$, the set of variables which are summed over is partitioned into two disjoint subsets $R$ and $\{v\}$.

*Moving a summation to the left.*

$$\cdots \sum_R f \sum_{S,v} \cdots \Rightarrow \cdots \sum_{R,v} f \sum_S \cdots \tag{Move-Left}$$

*Swapping two adjacent factors.*

$$\cdots f_1 f_2 \cdots \Rightarrow \cdots f_2 f_1 \cdots \tag{Swap-Factors}$$

*Moving a summation to the right. This rule is only applicable if factor $f$ does not contain free variable $v$.*

$$\cdots \sum_{R,v} f \sum_S \cdots \Rightarrow \cdots \sum_R f \sum_{S,v} \cdots \tag{Move-Right}$$

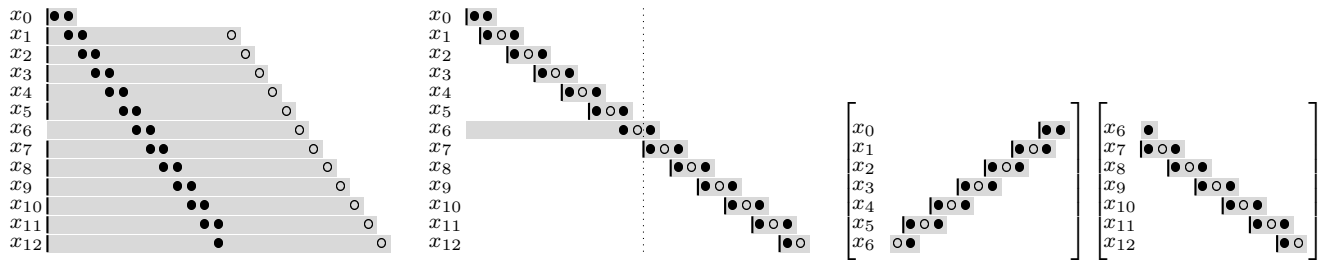
*Splitting an expression; $R$ is the set of variables which are both summed over in $\phi$ and free in $\psi$.*

$$\underbrace{\sum \cdots}_{\phi} \underbrace{\sum \cdots}_{\psi} \Rightarrow \sum_R \underbrace{\left[\sum \cdots\right]}_{\phi} \underbrace{\left[\sum \cdots\right]}_{\psi} \tag{Split-Expr}$$

## 3.3 Efficiency of sum-factor expressions

An important purpose of a sum-factor diagram is to visualise the efficiency of the expression. Talking about the efficiency of an expression implies an *operational semantics*: a function that maps an expression to an execution plan. When run, this execution plan results in a sequence of basic instructions (multiplication, addition). In this article, we are only concerned with the *number* of these basic instructions. However, there is nothing that prevents the definition of more complex cost models (taking e.g. running time and memory consumption into account) for our execution plans.

A naive operational semantics for a sum-factor expression would map it to a nested loop over all variables (i.e. one level of nesting per variable). The number of instructions in such an execution plan is always exponential in the number of variables: no matter how a sum-factor expression is rewritten, the number of summations stays the same. However, such a plan often performs a lot of redundant work, because many calculations in the inner levels do not depend at all on the value of the variable in the outer level. As an example, take the expression $\sum_a f_1(a) \sum_b f_2(a,b) \sum_c f_3(b,c)$ and assume the variables all have a domain of 50 values. Calculation of the subexpression $\sum_c f_3(b,c)$ then requires 49 additions, and has to be repeated (because it has a different outcome) for every value of $b$, yielding $50 \cdot 49$ additions. However, this is again repeated for every value of $a$; $50 \cdot 50 \cdot 49$ additions, while no new values are calculated.



(a) Initial diagram. (b) Factors are interleaved, and summations moved to the right. (c) The diagram has been split, and the order of factors on the left side has been reversed.

Figure 2: Deriving an efficient inference algorithm for $\mathsf{P}(x_6|y_{1..12})$ in the regular HMM, using sum-factor diagrams. For readability, the $\mathsf{P}(y_t|x_t)$ factors are shown using open dots (○).

Hence, we introduce our own operational semantics for sum-factor expressions, which avoids recalculating subexpressions. In these semantics, the result of evaluating an expression $\phi$ is always a $n$-dimensional array, containing the expression's value for all possible combinations of values for its $n$ free variables $\mathrm{FV}(\phi)$.

Given a sum-factor expression $\phi$, we will write $[\![\phi]\!]$ for its execution plan. We inductively define $[\![\sum_R f \cdot \phi]\!]$:

$$\begin{array}{l}
\mathsf{v'}{:=}[\![\phi]\!]; \\
\overbrace{\mathsf{for}\ i_1{:=}..{:}\ \mathsf{for}\ i_2{:=}..{:}\cdots}^{\{i_1,i_2,\ldots\}=\mathrm{FV}(\sum_R f\cdot\phi)} \\
\quad \mathsf{v}(i_1,i_2,..){:=}0; \\
\quad \overbrace{\mathsf{for}\ j_1{:=}..{:}\ \mathsf{for}\ j_2{:=}..{:}\cdots}^{\{j_1,j_2,\ldots\}=R} \\
\qquad \mathsf{v}(i_1,i_2,..){:=}\,\mathsf{v}(i_1,i_2,..){+}\ \overbrace{f(i_p,..,j_q,..)}^{\{i_p,j_q,\ldots\}=\mathrm{FV}(f)}\ \cdot\ \overbrace{\mathsf{v'}(i_r,..,j_s,..)}^{\{i_r,j_s,\ldots\}=\mathrm{FV}(\phi)}; \\
\mathsf{return\ v}
\end{array}$$

First, the plan for the inner sum-factor expression $\phi$ is executed, which results in array v'. Next, the outer loops range over the elements of the result array v. For every element, the inner loops sum the $f \cdot \phi$ product over the $R$ dimensions.

As a result of this recursive structure, the execution plan for a sum-factor expression starts at the rightmost factor. In this base case, there is no subexpression $\phi$: we then define array v' to have dimensionality 0 and value $\mathsf{v'}() := 1$.

Now, in a sum-factor diagram, each column corresponds to a set of loops like in the above plan. They loop over the grey variables: variables that have occurred in a subexpression and have not been summed out yet. These columns are each 'executed' once, from right to left. Thus, the number of basic instructions produced by a sum-factor expression's execution plan relates exponentially to the height of its diagram's grey area, and linearly to its width.

# 4. INFERENCE FOR OUR MODELS

In this section, we show how to derive efficient inference algorithms for our models using sum-factor diagrams. We start with the (Fact-BN) expression that is directly derivable from the model structure, and then use the rewrite rules from Sect. 3.2 to minimize the diagram's grey area.

## 4.1 Regular HMM

We first illustrate the inference of $\mathsf{P}(x_6|y_{1..12})$ for a regular HMM with length 12. Using (Inference) and (Fact-BN), we rewrite this probability into

$$\alpha \sum_{\substack{x_{0..5} \\ x_{7..12}}} \mathsf{P}(x_0) \left[\prod_{t=1..12} \mathsf{P}(x_t|x_{t-1})\right] \left[\prod_{t=1..12} \mathsf{P}(y_t|x_t)\right]$$

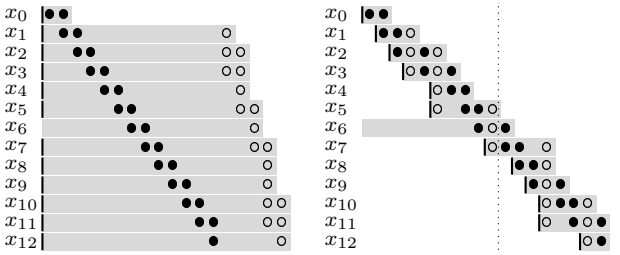
The $\prod_{t=1..12}$ expressions are not part of the sum-factor form: we use them to abbreviate a sequence of multiplications (starting with the $t$=1 factor and ending with the $t$=12 factor). Taking this into account, the above expression is in sum-factor form, and its diagram is shown in Fig. 2a. Note that the $y_t$ variables do not occur in the diagram because they are not free variables; at the time that we are doing the inference, we can replace them by their actual values (0 or 1). The $\mathsf{P}(y_t|x_t)$ factors only have one free variable left.

The large grey area shows that the expression is highly inefficient to evaluate in this form. To remedy this, we interleave the $\mathsf{P}(x_t|x_{t-1})$ factors with the $\mathsf{P}(y_t|x_t)$ factors by repeated application of (Swap-Factors), which permutes the columns of the diagram; then we move the summations as far to the right as possible using (Move-Right). The effect of these operations on the diagram is shown in Fig. 2b. Because we do not sum over query variable $X_6$, it still causes a grey line in the left side of the diagram; therefore, all evaluation steps on this side take $n$ times as much time. To avoid this, we use (Split-Expr) to split the expression at the dotted line. As the diagram shows, none of the free variables on the right side ($x_6$–$x_{12}$) are summed on the left side, so it is not necessary to move summations to the outer level. After the split, we reverse the order of the factors on the left side, which finally results in the split diagram in Fig. 2c.

Evaluating the left and right sides of this diagram according to the execution plan in Sect. 3.3 corresponds to a 'forward pass' from $X_0$ to $X_6$ and a 'backward pass' from $X_{12}$ to $X_6$, respectively; recall that evaluation proceeds from right to left in the diagram. As the maximal number of grey $x_t$ variables is 2, a column costs at most $n^2$ multiplications and additions (where $n$ is the size of the $X_t$ domain). Hence, the total complexity of HMM inference is $O(n^2T)$, with $T$ the number of time units.

## 4.2 HMM-AO

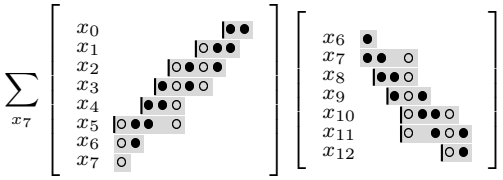
Derivation of the inference algorithm for the HMM-AO model goes along the same lines. As an example, we show the inference of $\mathsf{P}(x_6|y_{\mathsf{I}..\mathsf{v}})$, where the $Y_{\mathsf{i}}$ observations stretch over the intervals $X_1$–$X_3$, $X_2$–$X_5$, $X_5$–$X_7$, $X_7$–$X_{11}$ and $X_{10}$–



(a) Initial diagram.



(b) Factors are interleaved, summations moved right.



(c) The diagram has been split, and the order of factors on the left side has been reversed.

Figure 3: Inference of $\mathsf{P}(x_6|y_{\mathsf{I}..\mathsf{V}})$ in the HMM-AO model. The $\mathsf{P}(y_\mathsf{i}|x_{r..s})$ factors are shown using open dots (∘).

$X_{12}$, respectively. Filling in the equations (Inference) and (Fact-BN) yields the following sum-factor expression:

$$\alpha \sum_{\substack{x_{0..5}\\x_{7..12}}} \mathsf{P}(x_0) \left[\prod_{t=1..12} \mathsf{P}(x_t|x_{t-1})\right] \left[\prod_{\mathsf{i}=\mathsf{I}..\mathsf{V}} \mathsf{P}(y_\mathsf{i}|\mathit{parents}(Y_\mathsf{i}))\right]$$

Its diagram is shown in Fig. 3a; reordering the factors and moving the summations to the right yields Fig. 3b. Still following the example of the regular HMM, we split the expression to avoid the long lifetime of the $x_6$ variable. However, this time it is not possible to separate the free variables on the right side from the variables summed over on the left side: the factor $\mathsf{P}(y_{\mathsf{III}}|x_{5..7})$ will always cause trouble. We decide to put this factor on the left-hand side, which causes variable $x_7$ to occur on both sides. Hence, the summation over $x_7$ is lifted out of the sub-expressions. Next, we reverse the order on the left side again, and end up with the expression corresponding to Fig. 3c.

Note that the considerable amount of grey area in the figure is caused by the $\mathsf{P}(y_\mathsf{i}|\mathit{parents}(Y_\mathsf{i}))$ factors; as we do not make any assumptions about these cpds in the HMM-AO model, we cannot split them up. Hence, the number of grey $x_t$ variables in a column depends on the interval length $i$: the complexity of HMM-AO inference is $O(n^i T)$ (for $i \geq 2$).

## 4.3 HMM-NOR

For HMM-NOR, we *can* rewrite the $\mathsf{P}(y_\mathsf{i}|\mathit{parents}(Y_\mathsf{i}))$ factors, because we have restricted them to the (Cpd-NOR) form. As this form depends on the concrete value (0 or 1) that $Y_\mathsf{i}$ takes, we make a distinction in our rewriting procedure according to this value. In our example, we use the same observations $Y_\mathsf{I}$ through $Y_\mathsf{V}$ from the previous section, and additionally define that $Y_\mathsf{i} = 1$ for $\mathsf{i} = \mathsf{I}..\mathsf{IV}$, and $Y_\mathsf{V} = 0$.

The case for $Y_\mathsf{i} = 0$ is simple: in the sum-factor expression we use for inference, we can just replace the $\mathsf{P}(0_\mathsf{i}|\mathit{parents}(Y_\mathsf{i}))$ factor by the product of its sub-factors; the result is still a sum-factor expression. So, instead of one factor with three free variables $\mathsf{P}(0_\mathsf{V}|x_{10}, x_{11}, x_{12})$, we have three factors with one free variable each: $f_{\mathsf{V}10}(x_{10})$, $f_{\mathsf{V}11}(x_{11})$ and $f_{\mathsf{V}12}(x_{12})$. Like in the regular HMM, these can be interleaved between the $\mathsf{P}(x_t|x_{t-1})$ factors.

For $Y_\mathsf{i} = 1$, the case is more complicated. We cannot plug the form $1 - \prod_{t=r..s} f_{\mathsf{i}t}(x_t)$ directly into a sum-factor expression, so we have to rewrite it first. For each of these positive observations, we introduce a variable $b_\mathsf{i}$ in the sum-factor expression, which can take the values 0 and 1. We demonstrate how to rewrite the factor for $Y_{\mathsf{II}}$:

$$\begin{aligned}\mathsf{P}(1_{\mathsf{II}}|x_{2..5}) &= 1 - \prod_{t=2..5} f_{\mathsf{II}t}(x_t) = \sum_{b_{\mathsf{II}}} 1^{1-b_{\mathsf{II}}} \left(-\prod_{t=2..5} f_{\mathsf{II}t}(x_t)\right)^{b_{\mathsf{II}}} \\ &= \sum_{b_{\mathsf{II}}} (-1)^{b_{\mathsf{II}}} \prod_{t=2..5} (f_{\mathsf{II}t}(x_t))^{b_{\mathsf{II}}} \qquad \text{(NOR-as-SFE)}\end{aligned}$$

As usual, summation variable $b_{\mathsf{II}}$ implicitly ranges over all its possible values: 0 and 1. The last expression above can be plugged into the sum-factor expression in place of factor $\mathsf{P}(1_{\mathsf{II}}|x_{2..5})$ without problems. The range of the new $b_{\mathsf{II}}$ summation will expand until the end of the sum-factor expression, but this is no problem because this variable is not used in the rest of the expression.

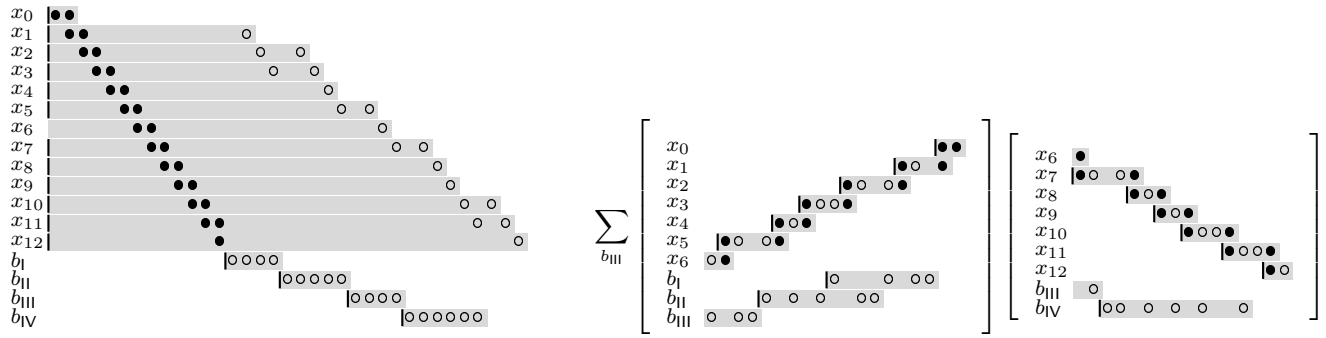
Now, for the inference of $\mathsf{P}(x_6|y_{\mathsf{I}..\mathsf{V}})$, we apply (Inference) and (Fact-BN), followed by the rewrites above:

$$\begin{aligned}&\alpha \sum_{\substack{x_{0..5}\\x_{7..12}}} \mathsf{P}(x_0) \left[\prod_{t=1..12} \mathsf{P}(x_t|x_{t-1})\right] \left[\prod_{\mathsf{i}=\mathsf{I}..\mathsf{V}} \mathsf{P}(y_\mathsf{i}|\mathit{parents}(Y_\mathsf{i}))\right] \\ =&\alpha \sum_{\substack{x_{0..5}\\x_{7..12}}} \mathsf{P}(x_0) \left[\prod_{t=1..12} \mathsf{P}(x_t|x_{t-1})\right] \\ &\sum_{b_\mathsf{I}} (-1)^{b_\mathsf{I}} \left[\prod_{t=1..3} (f_{\mathsf{I}t}(x_t))^{b_\mathsf{I}}\right] \cdots \sum_{b_{\mathsf{IV}}} (-1)^{b_{\mathsf{IV}}} \left[\prod_{t=7..11} (f_{\mathsf{IV}t}(x_t))^{b_{\mathsf{IV}}}\right] \\ &f_{\mathsf{V}10}(x_{10}) f_{\mathsf{V}11}(x_{11}) f_{\mathsf{V}12}(x_{12})\end{aligned}$$

The corresponding diagram is shown in Fig. 4a. Again, we reorder the factors for better efficiency. This time, we first move the $b_\mathsf{i}$ summations to the left, so we can freely reorder the $(f_{\mathsf{i}t}(x_t))^{b_\mathsf{i}}$ factors in ascending $x_t$ order. The purpose of this reordering is to keep the lifetime of the $x_t$ variables as short as possible, at the expense of the $b_\mathsf{i}$ lifetimes: a living $x_t$ variable makes the evaluation step $n$ times as expensive, while a living $b_\mathsf{i}$ variable makes it only twice as expensive. So, the factors are sorted first by $t$, then by i. Next, we continue along the usual lines, and end up with Fig. 4b (notice that $b_{\mathsf{III}}$ appeared on both sides of the split, so its summation had to be lifted out). The number of grey $x_t$ variables in a column is always 1 or 2; the number of grey $b_\mathsf{i}$ variables depends on the number $s$ of simultaneous $Y_\mathsf{i} = 1$ observations. So, the complexity of HMM-NOR inference is $O(n^2 2^s T)$.

# 5. RELATED WORK

Sensor data implies probabilistic dependencies. The fact that Bayesian networks represent these dependencies in a clear and efficient way is well recognized within the AI community; the same goes for the fact that their (conditional) *in*dependencies can be exploited for efficient inference. This has resulted in a lot of techniques to derive, from the struc-




(a) Initial diagram. (b) Final diagram.

Figure 4: Inference of $\mathsf{P}(x_6|y_{1..12})$ in the HMM-NOR model. The $(-1)^{b_{\mathsf{i}}}$ and $(f_{\mathsf{i}t}(x_t))^{b_{\mathsf{i}}}$ factors are shown using open dots (∘).

ture of the network, an efficient 'execution plan' for an inference query; the best known are *variable elimination*[12] and *junction tree propagation*[7] (which is also known as join tree or clique tree propagation, and generalizes *belief propagation*[8]). Finding an optimal plan is NP-complete[1], but there are several heuristics.

The advantages of Bayesian networks (or, more generally, factored distributions) for scalable probabilistic processing has also been recognized by database research: they are used in systems that probabilistically model the existence of relational tuples[10] as well as in systems for processing streaming sensor data[5]. From a database perspective, the AI inference techniques we mentioned can be cast as query optimization[11, 2].

However, these techniques depend on heuristics and/or graphical transformations. They can deal with any Bayesian network, but are reasonably complicated to apply. For our simple networks (whose graphical structure is essentially a chain), they are 'overkill'; moreover, they do not lead to the efficient decomposition (NOR-AS-SFE) of the Noisy-OR cpd, unless an additional dedicated technique[4] is used.

## 6. CONCLUSIONS & FUTURE WORK

We have presented two simple and general probabilistic models for interval-valued sensor data processing, and focused on a new method for deriving efficient inference algorithms for these models. The core of this method is the sum-factor expression, a representation of a probabilistic inference query that lends itself well to database research: it both allows easy algebraic manipulation (cf. query plan rewriting) and, via its operational semantics, a cost analysis. Using the sum-factor diagram, manipulation and analysis can also be done visually. In contrast to the AI methods, the method does not require graphical manipulations or understanding of probabilistic semantics.

A restriction of the sum-factor expression is that it only represents linear (right-deep) query plans. However, the graphical structure of our models is also linear; it is essentially a Markov chain. Presumably, this holds for many dynamic models used in sensor networks.

We hope that the sum-factor expression can provide a fertile common ground for integrating optimization techniques from the AI and database fields.

*Acknowledgements.* The authors would like to thank the anonymous reviewers for their constructive comments.

## 7. REFERENCES

[1] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[2] Héctor Corrada Bravo and Raghu Ramakrishnan. Optimizing MPF queries: decision support and probabilistic inference. In *SIGMOD Conference*, pages 701–712, 2007.

[3] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[4] Francisco Javier Díez and Severino F. Galán. Efficient computation for the Noisy MAX. *Int. J. Intell. Syst.*, 18(2):165–177, 2003.

[5] Bhargav Kanagal and Amol Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE2008)*, pages 1160–1169, April 2008.

[6] John Krumm and Eric Horvitz. LOCADIO: Inferring Motion and Location from Wi-Fi Signal Strengths. In *MobiQuitous*, pages 4–13. IEEE Computer Society, 2004.

[7] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B*, 50(2):157–224, 1988.

[8] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, USA, 1988.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2004.

[10] Prithviraj Sen and Amol Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proc. of the 23rd Int. Conf. on Data Eng. (ICDE)*, pages 596–605, April 2007.

[11] S. K. Michael Wong, Cory J. Butz, and Yang Xiang. A method for implementing a probabilistic model as a relational database. In *Proc. 11th Conf. on Uncertainty in AI*, pages 556–564, 1995.

[12] Nevin Lianwen Zhang and David Poole. Exploiting causal independence in Bayesian network inference. *J. Artif. Intell. Res. (JAIR)*, 5:301–328, 1996.