HAL
open science

# From Factorial and Hierarchical HMM to Bayesian Network : A Representation Change Algorithm

Sylvain Gelly, Nicolas Bredeche, Michèle Sebag

## To cite this version:

Sylvain Gelly, Nicolas Bredeche, Michèle Sebag. From Factorial and Hierarchical HMM to Bayesian Network : A Representation Change Algorithm. Symposium on Abstraction, Reformulation and Approximation, Jul 2005, Edinburgh, Scotland, UK. inria-00000548

**HAL Id: inria-00000548**

**https://inria.hal.science/inria-00000548v1**

Submitted on 9 Nov 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# From Factorial and Hierarchical HMM to Bayesian Network : A Representation Change Algorithm

Sylvain Gelly, Nicolas Bredeche, and Michèle Sebag

Equipe Inference&Apprentissage - Projet TAO (INRIA futurs),
Laboratoire de Recherche en Informatique,
Université Paris-Sud, 91405 Orsay Cedex
FRANCE
(gelly,bredeche,sebag)@lri.fr
http://tao.lri.fr

**Abstract.** Factorial Hierarchical Hidden Markov Models (FHHMM) provides a powerful way to endow an autonomous mobile robot with efficient map-building and map-navigation behaviors. However, the inference mechanism in FHHMM has seldom been studied. In this paper, we suggest an algorithm that transforms a FHHMM into a Bayesian Network in order to be able to perform inference. As a matter of fact, inference in Bayesian Network is a well-known mechanism and this representation formalism provides a well grounded theoretical background that may help us to achieve our goal. The algorithm we present can handle two problems arising in such a representation change : (1) the cost due to taking into account multiple dependencies between variables (e.g. compute $P(Y|X_1, X_2, ..., X_n)$), and (2) the removal of the directed cycles that may be present in the source graph. Finally, we show that our model is able to learn faster than a classical Bayesian network based representation when few (or unreliable) data is available, which is a key feature when it comes to mobile robotics.

## 1 Introduction

Many works in mobile robotics rely on probabilistic models such as POMDP or HMM[1], etc.) to build a map of an environment [2, 1, 7, 4, 5]. Indeed, the properties of these models are particularly relevant in the context of robotics, as well as extensions of these models. Firstly, the problem of knowledge generalization can partly be solved if we consider a hierarchical model (encode a given place at sereral granularities) [6]. Secondly, taking into account the invariants can also be achieved if we consider a model that implements a factorization operator (e.g. a given place location should be perceived with no considerations for the actual

[1] in the following of the article, we deal with HMM rather than with POMDP. The particularity of the latter being that they explicitly take into account action, which is not a key issue for the inference problem at hand.

orientation of the robot) [4]. However these two extensions have been well studied separately, it is quite difficult to endow a HMM-based model with these two simultaneously. As far as we know, there exists no efficient inference algorithm that can deal with such a model.

In this paper, we present an approach to perform inference within a Factorial and Hierarchical HMM (i.e. FHHMM[2]). Our approach relies on an algorithm that performs a representation change from FHHMM to the Bayesian Network representation formalism. The choice of the Bayesian Network formalism is motivated by the strong theoritical fundations and the efficient algorithms that exists in it.

However, several difficulties arise with such a representation change because of the structural differences between the two formalisms and their intrinsic properties. In particular, we identify two main problems that must be taken into account during this process :

- There exists multiple dependencies in the FHHMM. These implies an exponential growth of the number of parameters to learn, which is a challenging problem when dealing with a small set of example (this is an intrinsic property in mobile robotics) ;
- There exists directed cycles in the conditional dependencies between the variables of a FHHMM. It is well known that directed cycles are not allowed within a Bayesian network (we should note however that these dependencies are a problem only between variables at a same time step (see section 2)).

In the following section, we present the HMM formalism and the factorial and hierarchical extensions. Then, we describe the inference problem in the case of FHHMM. Section 3 and 4 presents our approach along with the representation change algorithm. Lastly, section 5 presents two experiments which confront the resulting model and classical Bayesian networks for a learning task. We conclude this paper with a discussion about the interesting properties shown by our model as well as the compromise we made so as to be able to learn from few data, which is often the case of a mobile robot building a map of its environment.

## 2 Problem Setting

### 2.1 Hierarchical and Factorial HMM

Known limitations with HMM, and more generaly with markov models, are concerned with scaling, taking into account independent phenomena and the difficulty to generalize. However, there exists several extensions to solve this problem. In the following, we focus our attention on hierarchical HMM [7, 5] and factorial HMM [3][3].

[2] We use this abrevation in the following of the article.

[3] These extensions have been used separately (with POMDPs) for map-building by a robot [5, 4].

On the one hand, the hierarchical extension allows to reduce the number of links between the states of an HMM, and then reduce the algorithmic complexity of learning as well as improving the acurracy. On the other hand, the factorial extension makes it possible to explain observations with several causes rather than only one. In this case, the goal is to turn the $P(Y|X)$ of HMM into $P(Y|X^1, X^2, ..., X^n)$. The $X^i$ are hidden variables and can be dealt with separatly. Thus, the $P(X^i_{t+1}|X^i_t)$ are different for each $i$.

### 2.2 Conditional dependencies and sparse data

Let's begin by introducing the following definitions :

- A static dependency denotes the conditional dependency between two variables at the same time step. It is important to notice that the problem of directed cycles arise only from this kind of dependencies.
- A dynamic dependency is defined as a conditional dependency for two (e.g. classical HMM) or several variables between two time steps (e.g. factorial HMM).

Classic and hierarchical HMM contain only dynamic dependencies. However, static dependencies can be found in the case of factorial HMM when conditional dependencies are to be created between some variables.

In the scope of this paper, we consider a special kind of HMM, where the dependencies type may be *a priori* undefined. As a matter of fact, dynamic and static dependencies are both expressed as conditional dependencies within the Bayesian network formalism.

### 2.3 Problem Issues

Since we consider an HMM that implements both the factorial and hierarchical extensions along with undefined dependencies, we face the problem of finding a fitted inference algorithm. As a matter of fact, there do not exists any such algorithms for this kind of model. This is the first issue : how to perform inference in such a model.

Another important issue is that due to the original motivation (i.e. mobile robotics), we have to consider the case where there is few data to learn from. Indeed, the sample process is supposed to be controlled by the robot's behavior and the environment, which usually gives few and biaised examples. Hence, we state that a good property of our model would be to favor the learning speed even at the cost of a (reasonable) loss in accuracy.

## 3 Representation change : from FHHMM to Bayesian networks

### 3.1 Constrained representation change

**Taking into account multiple dependencies :** we suggest to reformulate a directed (and potentially cyclic) graph into a Bayesian network. Indeed, the

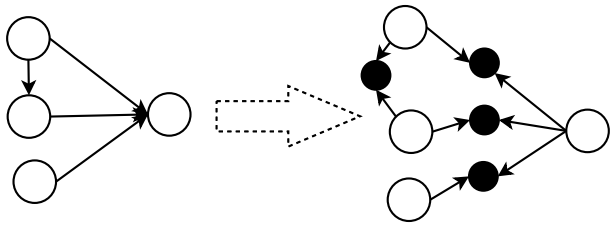

**Fig. 1.** Example of representation change (BN => FBN).

Bayesian network formalism is a well known and grounded theoritical and practical framework.

However, two problems arise with such a representation change : (1) the cost of taking into account the multiple dependencies which exist for a variable (i.e. computing $P(Y|X_1, X_2, ..., X_n)$, resulting in $2^n$ parameters when dealing with binary variables) and (2) reformulating a directed cycle within a Bayesian network.

Our solution rely on simplifying the constrains due to multiple dependencies. Indeed, multiple dependencies are decomposed by dealing with them two by two (i.e. taking separately $P(Y|X_1), P(Y|X_2), ..., P(Y|X_n)$ (resulting in $2n$ parameters for binary variables) as well as introducing constraints during the transformation process).

### 3.2 Taking into account multiple dependencies two by two

Let $V_1$, $V_2$, ..., $V_n$, with $n$ discrete random variables, of modality $m_1$, ..., $m_n$.

We assume that $p_i = P(V_i)$ are known (vector of size $m_i$), for all $i$, and some $p_{i,j} = P(V_j|V_i)$, $j \in I_i \subset \{1, ..., n\}$ ($p_{i,j}$ is a matrix of size $(m_i, m_j)$).

This model can be represented by a graph where nodes are random variables $V_i$ and edges $a_{i,j}$ that represents the $p_{i,j}$. The conditional probabilities induce a structure that is not constrained (for instance, there may exist directed cycles). In order to simplify the notation, we introduce the notion of **Flattened Bayesian Network** (or FBN) to designate the networks that are described in the following of the paper. Figure 1 shows an example of representation change from a graph into a Flattened Bayesian Network.

**Reformulating into Bayesian network formalism : additional variables and axioms :** For each pair of dependent variables $(V_i, V_j)$, we add an additional variable which parents are $V_i$ and $V_j$. This provides two advantages : (1) limiting the complexity of multiple dependencies (at the cost of approximation), (2) avoiding directed cycles (in the new formalism, all edges target additional variables). Once this reformulation is completed, inference is made possible thanks to one of the several inference algorithm of Bayesian network.

Each variable $V_i$ from the original graph is mapped into a variable of the Bayesian network, with the same modality, noted $V_i$ (as before).

Each edge $a_{i,j}$ is mapped into an additional boolean variable in the Bayesian network, noted $A_{i,j}$. The $A_{i,j}$ have exactly two parents in the Bayesian network, namely $V_i$ and $V_j$ (i.e. a V-structure). These variables are *artificially* observed in order to induce a dependency between the variables $V_i$ and $V_j$ (observation values are assigned to "true").

Once the additional variables are added, conditional probabilities must be computed as a last step to the transformation process, that is to compute the $P(A_{i,j}|V_i, V_j)$. Let's introduce the following notations :

- Let $K_j = \cup_i \{A_{i,j}\}$;
- Let $K = \cup_j K_j$. Let $L \subset K$. We note $L = true$ the event $\forall A \in L, A = true$.

Now, we shall define an axiomatic system to satisfy. The goal is to make the probabilities $P(A_{i,j}|V_i, V_j)$ reach a fixed point (i.e. stable). This fixed point is reached thanks to an EM-inspired iterative algorithm which is described in the following. Satisfying this axiomatic system garantees a coherent network behavior with respect to the dependencies taken two by two (compared to the behavior of a classic network).

The first axiom named "behavior axiom" determines the influence of a variable onto another. This axiom specifies a property defined from $K = true$, i.e. $\forall i, j\ A_{i,j} = true$. Then, this implies a coupled equation system. The **behavior axiom** is defined as follow :

$$\forall i, j\ P(V_j|V_i, K = true) = p_{i,j} \tag{1}$$

Secondly, the information contained in a probability distribution is linked to the difference between this distribution and the *a priori* distribution. We then introduce a second axiom named "not adding information" which states that adding additional variables do not bring information to the network. Then, this axiom implies local constrains on the $P(A_{i,j}|V_i, V_j)$, i.e. independently taking into account the $A_{i,j}$. The **not adding information axiom** is defined as follow :

$$\forall j, P(V_j|K = true) = p_j \tag{2}$$

Let's now describe the iterative process that satisfies the axioms. For more details on the equation system induced by the axioms, the reader can refer to the appendix at the end of this paper.

**Satisfaction mechanism of the axiomatic system :** for each iteration, there is an inter-dependency problem when computing the probabilities $P(A_{i,j}|V_i, V_j)$[4]. Indeed, if an element of the matrix $P(A_{i,j}|V_i, V_j)$ is modified, then the axioms may be invalidated for another dependency. In pratical, we check that the system

[4] This is even more true with directed cycles

satisfy the axioms once all the matrices are calculated. We iterate the process (updating the matrix) until it converges. This is achieved thanks to an EM-inspired iterative algorithm which is concerned with the axioms and is defined as follow :

- step E : $\forall i, j \; q_{i,j} = P(V_j | V_i, K \setminus \{A_{i,j}\} = true)$;
- step M : compute $P(A_{i,j} | V_i, V_j)$ wrt. $q_{i,j}$.

At this point, this algorithm is not sufficient to make $P(A_{i,j} | V_i, V_j)$ converge. Thus, we have to limit the influence between variables through "limited update" constraints. In the following, we present the mechanisms which are necessary to the algorithm that will be described in the next section.

*Convergence parameter : link "strength"* For each arc between two variables, we introduce a new term, namely "strength", which determines the influence of one variable upon another. A zero strength means that the variable has no direct influence (i.e. same as removing the additional variable). The strength is expressed by $f$, function defined on the set of additional variables $A_{i,j}$. $f(A_{i,j}) = (f_1(A_{i,j}), ..., f_{m_i}(A_{i,j}))$ is a vector of size $m_i$ (number of modality for the variable $V_i$), and $f_k(A_{i,j}) = 1 - H_k(P(A_{i,j} | V_i, V_j))$ where $H_k(P(A_{i,j} | V_i, V_j))$ is the entropy of line $k$ ($P(A_{i,j} | V_i, V_j)$ is a matrix).

*Updating criterion used to converge : limiting the direct influence of variables thanks to the strength term.* In order to compute the influence of a variable $i$ on another variable $j$, we have to take into account both the direct influence (i.e. through an additional variable $A_{ij}$) and indirect influence (i.e. through the other variables of which $i$ and $j$ both depend).

For some configurations however, influences will compensate each other so that they will both tend to a limit state (probability will tend to 0 or 1), making it difficult to take them into account any further. As a matter of fact, we shall then face (1) possibly infinite convergence towards 0 or 1 and (2) computational problem related the computer accuracy (the latter being the most important in practical).

In order to solve this problem, we compute a maximum threshold for the strength which is defined for every pairs of variables and for every modality of the source variable such as :

Let $f_k^0(i, j) = f_k(A_{i,j})$ when $\forall i, j \; q_{i,j} = p_j$.

This threshold is meant to be used as the link strength if there is no indirect influence. Hence, the iterative algorithm we present in the next section must satisfy for each step : $\forall i, j \; f_k(A_{i,j}) \leq f_k^0(i, j)$ (refer to algorithm 2 in the next section).

## 4 Representation change algorithm

In this section, we present two complementary algorithms that perform the desired representation change. The first algorithm makes the system converge (i.e.

N iterations until convergence) while the second algorithm makes sure that the representation change is performed with respect to the axioms for any pair of variables (i.e. a single iteration which may or may not lead to convergence).

### 4.1 Algorithm 1 : do $N$ iterations until convergence

**while** $P(A_{i,j}|V_i, V_j)$ haven't converged (distance from the term before is more than a given threshold) or while the number of iterations have not reached a maximum **do**
  call algorithm 2
  compute the distance between new and old probabilities
**end while**

### 4.2 Algorithm 2 : do an iteration for all the variables pairs

**for all** pairs of variables $V_i$, $V_j$ such that there exists a dependency $V_i -> V_j$ **do**
  **if** first iteration **then**
    Set all the additional variables as unobserved.
    Affect the $q_{i,j} = P(V_j)$.
  **else**
    Set the variable $A_{i,j}$ unobserved and the other additional variables observed to true
    Calculate the $q_{i,j} = P(V_j|V_i, K \setminus \{A_{i,j}\} = true)$ using an inference in the Bayesian network. These conditional probabilities represents the direct influence(without the link through variable $A_{i,j}$) of $V_i$ on $V_j$.
  **end if**
  Apply the equations of the first axiom in order to determine the $P(A_{i,j}|V_i, V_j)$ with a multiply constant for each line $i$
  **for all** The lines $k$ of the matrix $P(A_{i,j}|V_i, V_j)$, caculate the "strength" $f_k = f_k(A_{i,j}) = 1 - H_k(P(A_{i,j}|V_i, V_j))$ of the link $i -> j$. **do**
    **if** First iteration **then**
      $f_k^0(i, j) = f_k(A_{i,j})$
    **else**
      **if** $f_k > f_k^0$ **then**
        Calculate by dichotomy the $0 \leq y \leq 1$ such as $f_k(A_{i,j}^y))f_k^0$,(i.e. all the coefficients of the matrix are powered by $y$). This is done in order to "smooth" the parameters to increase the entropy and then decrease the "strength".
      **end if**
    **end if**
  **end for**
  Apply the equations of the second axiom to determine the multiply constants
  Compute the matrix $P(A_{i,j}|V_i, V_j)$

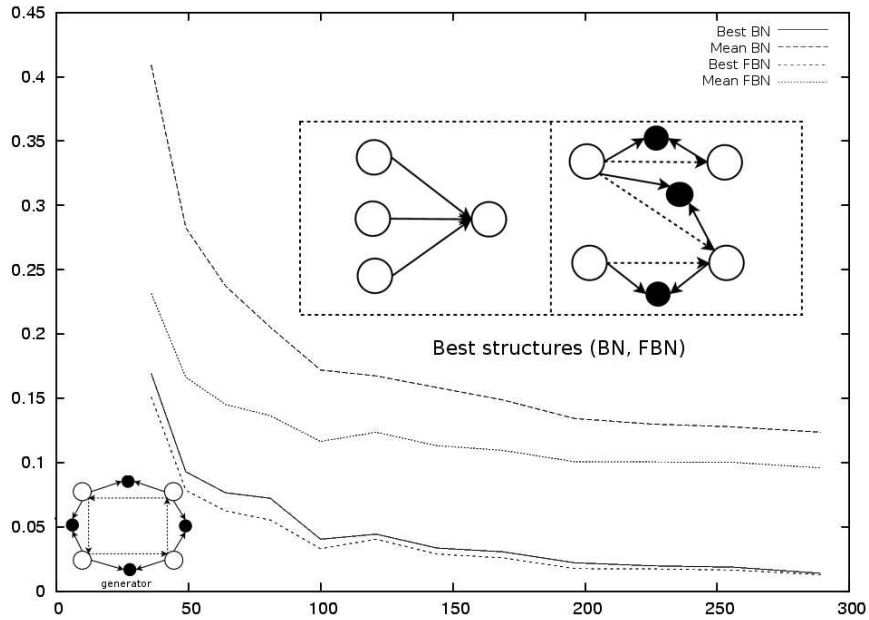


**Fig. 2.** Results using a Flattened Bayesian Network generator. The X-axis shows the number of examples used for learning. The Y-axis shows the Kullback-Leibler distance between the learned joint distribution and the one that was used to generate the learning data. The generator network is shown on the figure (lower-left). The best performing Bayesian and flattened Bayesian networks for 50 examples are also shown on the figure (up).

21: **end for**

In the next section, we show some experiments that rely on this algorithms.

# 5 Experiments

## 5.1 Experimental setup

In order to experimentally validate our approach, we conducted some experiments on the learnability of the networks after a representation change (i.e. flattened Bayesian networks). Our experimental setup is defined as follow :

- a generator network which can either be a flattened Bayesian network (exp. 1) or a classic Bayesian network (exp. 2). In both experiments, the number of nodes in the generator and learnable networks is fixed (in the case of flattened Bayesian network, we do not count the additional nodes built by our representation change algorithm).

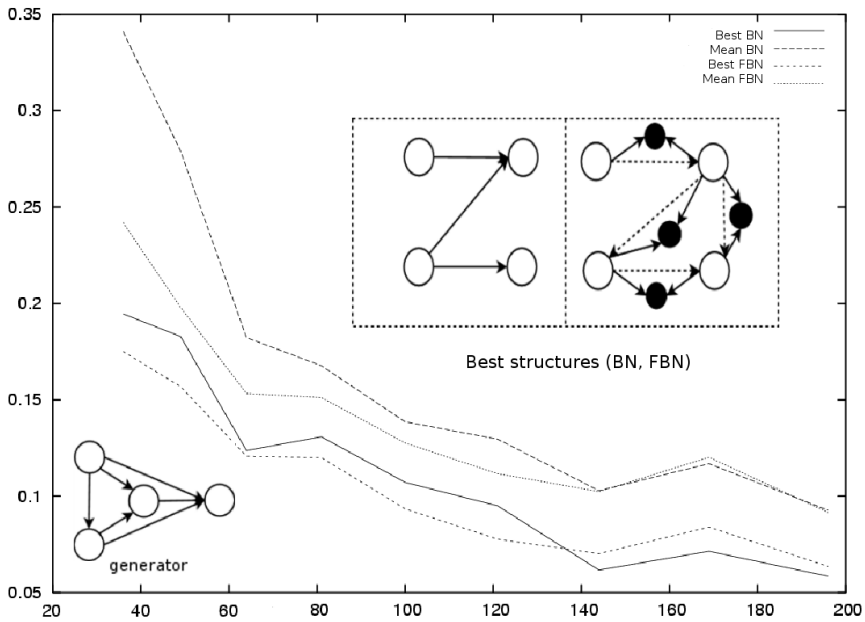


**Fig. 3.** Results using a Bayesian Network generator. The X-axis shows the number of examples used for learning. The Y-axis shows the Kullback-Leibler distance between the learned joint distribution and the one that was used to generate the learning data. The generator network is shown on the figure (lower-left). The best performing Bayesian and flattened Bayesian networks for 50 examples are also shown on the figure (up).

– a set of learning networks that covers both *all* the possible classic Bayesian networks and flattened Bayesian networks structures with the same number of nodes than the generator (i.e. learning is exhaustive for all structures with a given size).

So as to get a good approximation of the results, we compute $N$ data sequence from $M$ random initializations for the generator network. As a consequence, we perform $N * M$ learning sessions for each target network ($20 \leq N * M \leq 50$).

The error is defined as the Kullback-Leibler distance between the joint distribution of a given target network and the distribution of the generator network. In the scope of this paper, the network size for all experiments is limited to 4 so that it is possible to evaluate the performance for all possible structures. As a matter of fact, the number of possible structures grows more than exponentially in function of the network size, which makes computation quickly prohibitive.

### 5.2 Experiment 1 : Learning from data generated by a flattened Bayesian network

Firstly, we study the behavior of flattened Bayesian networks in the most favorable setup, i.e. when learning on data generated by a flattened Bayesian network. In this experiment, the generator is a 4-node cyclic flattened Bayesian network. Figure 2 shows this generator as well as the results obtained with both all the flattened Bayesian networks and classic Bayesian network that contains 4 nodes.

This figure shows that the flattened Bayesian networks always perform better for average *and* best performances. However, learning performance tends to be the same as the number of examples increases ($\geq$ 250). Flattened Bayesian networks are thus relevant when learning from such data. Moreover it should be noted that the best performing flattened Bayesian network is structuraly different from the generator, meaning that the more reliable structure when few examples are available is not the very structure of the generator.

### 5.3 Experiment 2 : Learning from few examples

Secondly, we choose a 4-node classic Bayesian network as data generator (cf. fig. 3). As a consequence, learning with flattened Bayesian networks faces the worst case since the generator's joint probability can be anything. As a matter of fact, flattened Bayesian network are supposed to be better for *some* distributions (unknown at this stage of our research).

Figure 3 shows the results with respect to the experimental setup described earlier. The important result is that the flattened Bayesian networks show the best results both in average and for the best when there are few examples to learn from. However, classic Bayesian networks become better as the number of examples grow. These results show clearly that flattened Bayesian network pay for the advantage of learning speed with a loss in accuracy in the long term (i.e. compromise between a fast learning curve againt non-accurate learning in the long term).

### 5.4 Discussion

According to the results obtained earlier, it appears that the best networks are also the simplest ones. Thus, it seems more relevant to learn with a simple yet inadequate structure rather than with a more complex structure that is closer to the generator : this can be seen as an explanation for the good learning capabilities of flattened Bayesian networks. Figure 4 tends to confirm this assertion by showing the distribution of classic and flattened Bayesian networks according the learning performance for a given number of examples (here arbitrarily fixed to 50) in experiment 2. Indeed this figure shows that flattened Bayesian network are much less sensitive to structural variations than classic Bayesian networks.

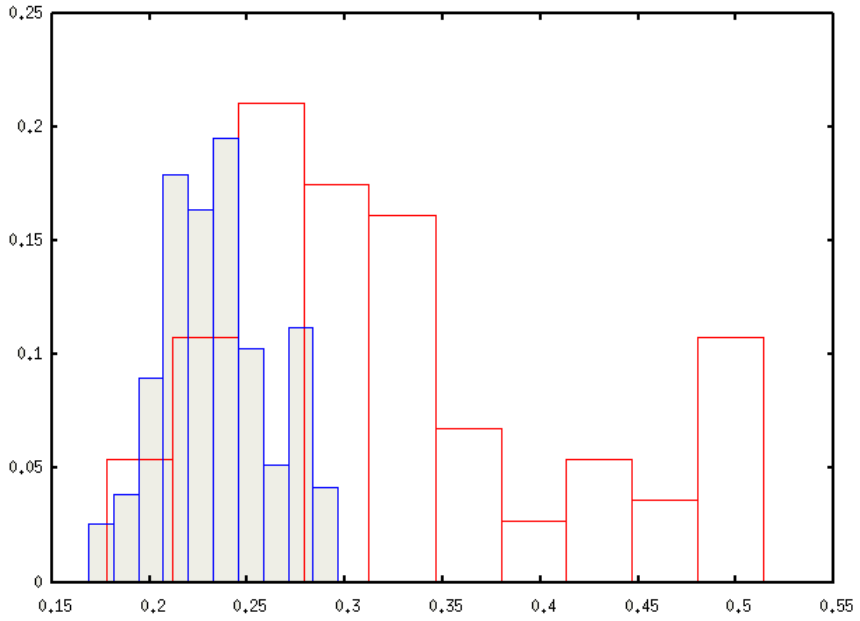


**Fig. 4.** Distribution (y-axis) wrt. the learning error (x-axis) for both classical (white bars) and flattened (grey bars) Bayesian networks. Learning performance for the flattened Bayesian networks are much more structuraly-independent than for classic Bayesian networks.

## 6 Conclusions

In the scope of this paper, we were interested in the transformation of a graph (in practical, a hierarchical and factorial HMM) into a Bayesian network according to some given constraints in order to reformulate the multiple dependencies and cycles inherent to such a representation. We presented an algorithm that performs a representation change in order to build a flattened Bayesian network. We also presented the axioms that are used to provide a relevant model of reformulated multiple dependencies. This model is based on a compromise between accurateness and learning speed which is achieved by taking into account multiple dependencies by modeling variables only by pairs.

In order to study the behavior of flattened Bayesian networks, we performed two experiments that successively showed (1) the learning behavior with a cyclic flattened Bayesian network generator and (2) the learning behavior with few examples. Thanks to these experiments, we have shown that flattened Bayesian networks are especially good when learning from few examples, compared to classic Bayesian networks.

Given our original motivation, i.e. map representation in mobile robotics, the results we obtained are very promising since it has been observed that flattened Bayesian network have the following properties :

- it is possible to modelize cycles one may encountered when dealing with factorial and hierarchical HMM;
- learning is performed more quickly with fewer examples. Of course, this results from a compromise that implies a loss of accuracy in the long term. However, in the scope of mobile robotics, this compromise is worthwhile since a robot often deals with few or biased examples to build a representation of the environment.

In the scope of this paper, we presented some experiments that compare classic and flattened Bayesian networks. However, the representation formalism remains the one of the Bayesian network, eventhough the representation change algorithm adds additional variables. Thus, it is possible to build some hybrid representations that combine both flattened and classical Bayesian sub-networks depending on the availability of data to learn from. As a consequence, this would make it possible to get the best of both worlds : learning from few examples with flattened Bayesian networks and precision learning with classic Bayesian networks when many examples are available.

Some issues remain to be explored. From the model viewpoint, the convergence mechanism describe in section 3.2 is based on experimental validation and may require some further theoretical investigations. From the robotic viewpoint, it is crucial to evaluate the learning behavior of flattened Bayesian networks using real-world data, i.e. such as those a robot could gather in its environment. Then, we should investigate the learning mechanism that may be used to find a relevant structure for a flattened Bayesian network, eventhough those netwoks have been shown to be less sensitive to an ill-chosen structure than classical Bayesian networks actually are.

## References


1. D. Fox and J. Ko and K. Konolige and B. Stewart: A hierarchical Bayesian Approach to the Revisiting Problem in Mobile Robot Map Building Proc. of the International Symposium of Robotics Research (ISRR-03) (2003)
2. David Filliat and Jean-Arcady Meyer: Global localization and topological map-learning for robot navigation Proceedings of the Seventh International Conference on simulation of adaptive behavior : From Animals to Animats (SAB-2002), pages 131-140. The MIT Press (2002)
3. Zoubin Ghahramani and Michael I. Jordan: Factorial Hidden Markov Models Machine Learning, vol. 29. 1996, pages 245-273
4. Michael Montemerlo and Sebastian Thrun and Daphne Koller and Ben Wegbreit: FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem Proc. 10 th National Conference on Artificial Intelligence p593-598 (2002)
5. G. Theocharous and K. Murphy and L. Kaelbling: Representing hierarchical POMDPs as DBNs for multi-scale robot localization Proc. of the IEEE international Conference on Robotics and Automation (ICRA'04) (2004)

6. G. Theocharous and Sridhar Mahadevan: Approximate Planning with Hierarchical Partially Observable Markov Decision Process Models for Robot Navigation Proc. of the IEEE International Conference on Robotics and Automation (ICRA'02) (2002)
7. G. Theocharous and K. Rohanimanesh and S. Mahadevan: Learning Hierarchical Partially Observable Markov Decision Processes for robot navigation Proceedings of the IEEE Conference on Robotics and Automation (ICRA-2001). IEEE Press (2001)

# 7 Appendix : Property of the model and conditional probabilities

This section details the equations obtains from the axioms.

We add to the notations above, for $i, j$ fixed : $K' = K \setminus \{A_{i,j}\}$.

## 7.1 First axiom

The first axiom, named "behavior" determines the influence of a variable on another. This axiom specify a property defined from $K = true$, that is to say from $\forall i, j\ A_{i,j} = true$. It can be expressed as :

Behavior axiom :

$$\forall i, j\ P(V_j | V_i, K = true) = p_{i,j}$$

(axiom 1)

From Bayes formulae,

$$\begin{aligned} & P(A_{i,j} = true | V_i = k, V_j = l, K' = true) \\ = & \frac{P(V_j = l | V_i = k, K = true) P(A_{i,j} = true | V_i = k, K' = true)}{P(V_j = l | V_i = k, K' = true)} \end{aligned} \tag{3}$$

Finally :

$$\forall k, l, P(A_{i,j} | V_i = k, V_j = l) = \gamma_k \frac{p_{i,j}(k, l)}{P(V_j = l | V_i = k, K' = true)}$$

The proportionality coefficients $\gamma_k$, which do not have influence on the satisfaction of this property allow us to obtain the following property.

## 7.2 Second axiom

The information contained in a probability distribution is linked to the difference between this distribution and the *a priori* distribution. We then introduce a second axiom named "not adding information" which states that adding additional variables will not bring information in the network. Then, this axiom implies local constraints on the $P(A_{i,j} | V_i, V_j)$, that is to say taking into account the $A_{i,j}$ independently. More precisely :

not adding information axiom :

$$\forall i, \forall k, P(V_i = k|K = true) = p_i(k)$$

Again from the Bayes formulae :

$$\begin{aligned} & P(V_i = k|K = true) \\ = & \frac{P(A_{i,j} = true|V_i = k, K' = true)P(V_i = k|K' = true)}{P(A_{i,j} = true|K' = true)} \\ = & P(V_i = k|K' = true) \\ & \frac{\sum_l P(A_{i,j} = true|V_j = l, V_i = k)p(V_j = l|V_i = k, K' = true)}{P(A_{i,j} = true|K' = true)} \end{aligned} \tag{4}$$

Futhermore, we have from above :

$$P(A_{i,j} = true|V_j = l, V_i = k) = \gamma_k \frac{p_{i,j}(k,l)}{P(V_j = l|V_i = k, K' = true)}$$

Hence :

$$\begin{aligned} & P(V_i = k|K = true) \\ = & \frac{P(V_i = k|K' = true) \sum_l \gamma_k \frac{p_{i,j}(k,l)}{P(V_j = l|V_i = k, K' = true)} p(V_j = l|V_i = k, K' = true)}{P(A_{i,j} = true|K' = true)} \\ = & \frac{P(V_i = k|K' = true) \sum_l \gamma_k p_{i,j}(k,l)}{P(A_{i,j} = true|K' = true)} \\ = & \frac{P(V_i = k|K' = true)\gamma_k}{P(A_{i,j} = true|K' = true)} \end{aligned}$$

Finally :

$$\forall k, \gamma_k = P(A_{i,j} = true|K' = true) \tag{5}$$

All the $\gamma_k$ are equals. This constant does not have influence on the wanted properties and is then chosen in order to have all the probabilities between 0 and 1, and secondly for numerical considerations. More precisely it is chosen such as

$$max_{j,l} P(A_{i,j} = true|V_i = k, V_j = l) = 1$$

■