Approximate Inference in Dynamic Possibilistic Networks

Abdelkader Heni, Member, IEEE & Adel M. Alimi, Senior Member, IEEE

Abstract—This paper describes an approximate algorithm for inference in dynamic possibilistic networks (DPNs). DPNs provide a succinct and expressive graphical language for representing sequential data and factoring joint possibility distributions and they are powerful models using only the concepts of random variables and conditional possibilities. The proposed algorithm, to perform inference in such networks, is an approximate one and it is based mainly on the standard Boyen-Koller (BK) algorithm well defined for dynamic probabilistic networks. The new possibilistic framework, proposed in this paper, is notable because it gives a counterpart of traditional probability framework, generally used to represent uncertainty in sequential data. The possibilistic BK algorithm is based on the junction tree technique where inference is done via an interface clusters that decrease the size of the dynamic network structured and amenable to a very simple form of inference. We present this algorithm in terms of two possibilistic conditioning; the product based and the min-based one.

Index Terms—Approximate inference, Dynamic possibilistic networks, BK algorithm, interface algorithm.

I. INTRODUCTION

DURING the last few years, Bayesian networks, also known by belief networks, causal networks, have become a very popular approach in computer science. One of the reasons is their higher performance in knowledge and causality representation, where exact and approximate inference algorithms are used to cope with the difficulties of calculation. Dynamic Bayesian networks are an extension of Bayesian networks and they give a powerful framework for modeling dynamic systems.

We explore in this paper a better understanding of an alternative version of these models in a possibilistic way. We called these models dynamic possibilistic networks (DPNs). We show an approximate algorithm for inference in such networks, named PBK (Possibilistic Boyen-Koller) as an approximate possibilistic reasoning mechanism that provides filtering and prediction tasks, given a set of available uncertain information. Our research on this algorithm has addressed two main challenges. On one front, we have explored the use of a product-based and min-based reasoning architecture. On another front, we have investigated many concepts and notions used in DBNs (dynamic Bayesian networks). We have shown how the selective use of clusters and associated computation reduces the overall computational task when working with an exact algorithm as described in our work [3],[4]. This paper is organized as follows: we first provide background on possibility theory, possibilistic networks and dynamic possibilistic networks in section 2. Section 3 describes our work on approximate inference in dynamic graphical models and gives details for the PBK algorithm. Finally, we summarize our work in section 4.

II. BACKGROUND ON DBNs AND DPNs

Dynamic Bayesian networks present several advantages to the problem of knowledge representation from sequential information: they can handle incomplete data as well as uncertainty; they are simple and provide means for decision making; they encode causality in a natural way [1],[2]; algorithms exist for filtering and doing predictive inference; they offer a framework for combining prior knowledge and data; finally, they offer a natural way for sequential data representation. However, they pose, in the general case, difficult inference problems, especially with unrolled graphs when we use a standard algorithm as those defined to make inference in standard causal networks [6],[7]. Several algorithms have been defined during the last few years to perform inference in dynamic probabilistic networks [13],[14],[18].

With different available representations, there is still the open question of how suitable a possibilistic representation might be for these. In this section we give a brief description of this possibilistic framework, but we will first, show a brief outline of possibility theory and possibilistic networks.

A. Possibility distributions and possibility measures:

A possibility distribution $\Pi$ is a mapping from $\Omega$ to the interval $[0,1]$. The degree $\pi(\omega)$ represents the compatibility of $\omega$ with the available information about the real world. By convention, $\pi(\omega) = 0$ means that the interpretation $\omega$ is impossible, and $\pi(\omega) = 1$ means that $\omega$ is totally possible. Two different ways of rank ordering formulae of the language are defined from this possibility distribution [11]. This is obtained using two mappings grading, respectively, the possibility and the certainty of a formula $p$:

- The possibility (or consistency) degree:
  $$\Pi(p) = \max \{ \pi(\omega) : \omega \in [p] \}$$

Manuscript received in revised form June 25, 2007. This work was supported in part by the Research Group on Intelligent Machines REGIM-ENIS, Tunisia.

A. Heni is with the Preparatory Institute for Engineers Studies, Monastir, 5019, Tunisia (phone: +216-98-915-119; fax: +216-73-500-512; e-mail: abdelkader.heni@ieee.org).

Adel M. Alimi is the header of the Research Group on Intelligent Machines (REGIM), University of Sfax, ENIS, BP. W-3038 – Sfax – Tunisia (phone: +216-74-274-090; e-mail: adel.alimi@ieee.org).
Which evaluates the extent to which \( p \) is consistent with the available beliefs expressed by \( p \) [11]. It satisfies:

\[
\forall p, \forall q \quad \Pi(p \wedge q) = \max(\Pi(p), \Pi(q)) \quad (2)
\]

• The necessity (certainty) degree

\[
N(p) = 1 - \Pi(\neg p) \quad (3)
\]

Which evaluates the extent to which \( p \) is entailed by the available beliefs. We have:

\[
\forall p, \forall q \quad N(p \wedge q) = \max(N(p), N(q)) \quad (4)
\]

A. Possibilistic networks:

Possibilistic networks are a counterpart of probabilistic networks. They are graphical models that are capable of modeling domains comprising uncertainty. They were introduced to the field of expert systems by [15]-[17]. Since then possibilistic networks were successfully applied in several areas. Strength of these models is not only that they enable efficient uncertainty reasoning with random variables but also they help humans to understand better the modeled domain. This is mainly due to their comprehensible representation by use of directed acyclic graphs representing dependencies between domain variables.

**Definition 1:** a possibilistic network is a directed acyclic graph (DAG) \( G = (V, E) \). Where \( V \) is the set of nodes and \( E \) is the set of links (edges) between them. To each node \( X_i \in V \) corresponds one random variable, and a conditional possibility distribution (CPD): \( \pi(X_i|X_j) \in Pa(X_i) \), where \( Pa(X_i) \) denotes the set of parents of node \( X_i \) in the possibilistic graph (\( PG \)).

\[
\pi(P) \quad \text{Point} \quad \text{Stroke} \quad \pi(S)
\]

\[
\pi(L|P,S) \quad \text{Letter} \quad \text{Number} \quad \pi(N|S)
\]

Fig.1. Example of possibilistic network used for handwriting recognition.

A possibilistic network encodes a decomposition of possibility distribution according to the following chain rule:

\[
\pi(A_1,\ldots,A_n) = \pi(A_i | Pa(A)) \quad (5)
\]

Where \( \pi(A_1,\ldots,A_n) \) is either the product or the minimum operator.

B. Dynamic Possibilistic networks:

Analogously to dynamic Bayesian networks (DBNs), we define dynamic possibilistic networks (DPNs) as an extension of possibilistic networks for modeling dynamic systems. In a DPN, the state at time \( t \) is represented by a set of random variables \( Z_t = (Z^1_t,\ldots,Z^n_t) \). The state at time \( t \) is dependent on the states at previous time steps. We assume that each state only depends on the immediately preceding state (i.e., the system is first-order Markov), and thus we need to represent the transition distribution \( \pi(Z_t^1 | Z_t) \). This can be done using a two-time-slice possibilistic network fragment (2TPN) \( P^{t,t+1} \), which contains variables from \( Z^n_t \) whose parents are variables from \( Z \) and/or from \( Z^{n_2} \), and variables from \( Z \) without their parents in time-slice \( t-1 \). We also assume that the process is stationary, i.e., the transition models for all time-slices are identical: \( P^{t} = P^{0} = P^{T} \). Thus a DPN is defined to be a pair of Bayesian networks (\( P^{t} \), \( B^{t} \)), where \( P^{t} \) represents the initial distribution \( \pi(Z^1_0) \), and \( P^{t} \rightarrow i \) is a two-time-slice possibilistic network, which defines the transition distribution \( \pi(Z^1_{t+1} | Z_t) \).

Fig. 2. An example of DPN with three time-slices.

The set \( Z \) is commonly divided into two sets: the unobserved state variables \( X \) and the observed (the set of evidences) variables \( Y \). The observed variables \( Y \) are assumed to depend only on the current state variables \( X \). The joint possibility distribution represented by a DPN can then be obtained by unrolling the 2TPN. Using these models, a DPN provides two compact factorizations for the posterior distributions \( \Pi(X_t^1) \) for any \( T \):

**Product-based factorization:**

\[
\Pi_t(X_t^1) = \bigotimes_{t=1}^T (\Pi(X_t | Y_t), \Pi(X_{t-1} | X_t^1, Y_t)) \quad (6)
\]

**Min-based factorisation:**

\[
\Pi_t(X_t^1) = \min_{t=1}^T (\Pi(X_t | Y_t), \Pi(X_{t-1} | X_t^1, Y_t)) \quad (7)
\]

III. APPROXIMATE INFERENCE IN DPNS

In this section we describe our approximate algorithm for inference in DPNs. This algorithm is based on the standard Boyen-Koller (BK) algorithm [18] which performs approximate probabilistic inference in the context of dynamic processes. We will describe our possibilistic version of this algorithm by only considering processes that can be modeled by dynamic possibilistic networks. Approximate inference is performed through junction trees. Details of construction of these trees are the same as described for static possibilistic networks (moralization, triangulation etc.). In order to understand the details of this algorithm, we will show first, some concepts:

A. Boyen-Koller Approximation

Based on assumptions of independence between chosen sets of variables in outgoing interface, the BK approximation is a general scheme for stochastic processes defined in [18]. Boyen and Koller show in their work how to use the BK approximation in the junction tree algorithm for Dynamic Bayesian Networks [5]. This approximation requires less computation and storage than interface algorithm defined by Murphy in [13], [14].
**Definition 2:** The outgoing interface $I_t$ of a 2-TPN Graph $\Pi G$, is the set of nodes in the time slice $t$ with children in time-slice $t+1$. Formally an Interface is defined as a set $I$, such that:

$$I_t = \{ u \in V_t \mid \exists (u,v) \in E^{\text{out}}(t+1), v \in V_{t+1} \}$$

Where $V_t$ is the set of nodes in time-slice $t$, $V_{t+1}$ is the set of nodes in time-slice $t+1$ and $E^{\text{out}}(t+1)$ is the set of temporal edges linking some nodes in time-slice $t$ with some nodes in time-slice $t+1$. (Figure 3)

**Definition 3:** A BK cluster is a set of nodes which form a partition of outgoing interface. The BK approximation scheme assumes independence among these sets.

![Fig. 3. A 2TPN. (a) The Murphy’s clusters. (b) The BK interfaces [12]](image)

In Figure 3(a), $\{A_1, B_1, C_1\}$ is the outgoing interface of time-slice 1 of a 2TPN, and the set $\{\{A_1, C_1\}, \{B_1\}\}$ in Figure 3(b) is a valid set of BK clusters.

**B. Inference in DPNs**

The PBK algorithm (possibilistic Boyen-Koller) is an approximate version of the possibilistic interface algorithm defined in our work [3] for exact inference in DPNs. It mainly performs inference through two initial steps:

1) Initialization step: in this step we get two junction trees $J_1$ and $J_2$. $J_1$ is the junction tree obtained from the first time-slice of a 2TPN, and $J_2$ is the junction tree obtained from a ½-TPN, where the ½-TPN is the network created from a 2TPN after removing non interface nodes.

2) Global step: in this step, inference is performed either in $J_1$ or in $J_2$. If time = 0 propagation is performed in $J_2$ and we get a value $\alpha$ for each BK cluster which will be used if time is incremented to perform inference in $J_1$. When propagation is performed in $J_1$. When we increment time again a new Alpha values are obtained. These new values will be used in the next time and etc. Next subsections give details for the propagation process.

**C. Initialization steps**

1) Initialization of junction trees: constructing a junction tree from a graph is a process based on two steps; on the first hand we have to moralize the original graph (we add links to make nodes sharing a child in the same clique) and on the other hand the triangulation step. Before giving details about the initialization step, we show a brief outline of the two versions of possibilistic conditioning (Product-Based and Min-based). A product-based possibilistic network denoted $\Pi G$ is a graph over a set of variables where conditional distributions are defined by the product-based conditioning:

$$\pi(\omega | \phi) = \frac{\pi(\omega)}{\Pi(\phi)} \text{ if } \omega \in [\phi]$$

$$0 \text{ otherwise}$$

And the joint possibilistic distribution encoded by $\Pi G$ can be performed with the product-based chain rule

$$\Pi(A_1, \ldots, A_n) = \bigotimes_{i=1}^{n} \Pi(A_i | Pa(A_i))$$

On the other hand, a min-based possibilistic network denoted $\Pi G_p$ is a graph where conditional distributions are defined by the min-based conditioning:

$$\pi(\omega | \phi) = \begin{cases} 1 & \text{if } \pi(\omega) = \Pi(\phi) \text{ and } \omega \in [\phi] \\ \pi(\omega) & \text{if } \pi(\omega) < \Pi(\phi) \text{ and } \omega \in [\phi] \\ 0 & \text{otherwise} \end{cases}$$

And oint possibilistic distribution encoded by $\Pi G_p$ can be performed with the min-based chain rule

$$\Pi(A_1, \ldots, A_n) = \min \{ \Pi(A_i | Pa(A_i)) \}$$

In the initialization of $J_1$, which is valid for the two versions of DPNs, we identify the set of BK clusters and the set of cliques containing these clusters in a first time. In the next time, the initial graph ($P'$) should be moralized as a static PN by adding links between each pair of nodes sharing a child.

In addition we have to add links to make each BK cluster in the same clique [12]. Once, the moral graph is obtained and the cliques are identified, we can construct the junction tree.

**Algorithm 1: Initialization of the first junction tree $J_1$**

Input: A 2TPN  Output: a junction tree $J_1$

Begin

1- Get $P'$ by removing all nodes of the second time-slice of a 2TPN.

2- $BK \leftarrow \emptyset$ (initially empty)

Repeat

C $\leftarrow$ Boyen-Koller cluster C in $P'$

BK $\leftarrow$ BK + C

Until no new cluster is identified

3- Moralize: add links to make all nodes of each BK cluster in a clique (the set of BK out-cliques).

4- Construct $J_1$, using the same process as in a static PN.

Let m be the number of cliques.

5- Associate each of the N variables in $P_o$ to one and only one clique in $J_1$

6- Initialize clique’s and separator’s potentials

End.
Algorithm 2: Initialization of clique’s potentials in J1

Input: m cliques. Output: clique’s and separators potentials

1. for each clique Ci in the junction tree J1 (i = 1... m)
   \( \pi_{Ci} \leftarrow 1 \) (initialization clique’s potentials);
   Let h be the number of variables associated to Ci
   for each associated variable Ak to Ci (k = 1...h)
   if Ak is a non evidential node
     Case of PB-DPNs: \( \pi_{Ci} \leftarrow \odot(\pi_{Ci} , \pi(A_k|pa(A_k))) \);
     Case of M-B DPNs: \( \pi_{Ci} \leftarrow \min(\pi_{Ci} , \pi(A_k|pa(A_k))) \)
   else
     \( \pi_{Ci} \leftarrow \pi_{Ci} \); if Ak is instantiated to the value a_i in Ci
     \( \pi_{Ci} \leftarrow 0 \); otherwise
   end if
end for
2. For each separator Sij between two cliques Ci and Cj
   \( S_{ij} \leftarrow 1 \) (initialization of the set of separators)
end for

Algorithm 3: Initialization of the second junction tree J2

Input: a ½-TPN Output: a junction tree J2, a set of BK clusters and cliques.

1. Get the ½-TPN from the 2TPN by removing non interface nodes.
2. Identify the set of In-BK and the set of Out-BK clusters.
3. Moralize (same as in static PN)
   Key change: Add links as needed to make each BK cluster in the same clique.
4. Construct a junction tree J2 using the same process as in static PN.
   Let m be the number of cliques
5. Associate each variable V to one and only one clique C containing its parents.
6. Initialize clique’s and separator’s potentials

Note that in step 5 of algorithm 3, we only associate variables from time-slice 2 of a 2TPN.
Figure 5 shows the execution details of algorithm 3. The BK in-cliques are [(A1,A,C1), (B1,B2)] and
the BK out-cliques are [(A2,C1,C2), (C1,B2,D2)].

In step 3 of algorithm 1, the set of BK clusters is the only required information when we perform inference
from time-step to time-step. Since the standard BK algorithm approximates the outgoing interface potential
at each time-step by breaking up the outgoing interface into sets of nodes
and assuming that these sets of variables are independent.
The breakdown of these sets, which are called BK clusters,
is specified as an input-output to the algorithm. An example of applying steps 1-5 to the network presented
in figure 3.b is shown in figure 5.
The sixth step of algorithm 1 is the initialization of potentials. The details of this step are shown in algorithm 2:
The second junction tree is obtained by converting the 2TPN to ½ -TPN (figure 5(b)) in a first time.
In a second time the ½-TPN should be converted to a moral graph by adding links as in a static PN.
We note here that we have also to add links in order to make each BK cluster in clique.
The details of the initialization are shown in algorithm 3.

Example 1: let the prior and conditional distributions for the DPN presented in figure 5,
be as described in table 1.
In the initialization step of clique’s and separator’s potentials for the first junction tree J1, we get these results in the case of PB-DPN:
\[ \pi_c = \pi(A, C) = \pi(A) \pi(C | A) = (1, 0.5, 0.3, 0.12) \]
since the associated nodes are A and C,
\[ \pi_c = \pi(B, C, D) = \pi(B) \pi(D | B, C) = (1, 0.6, 1, 0.2, 0.1, 0.5, 0.05, 0.5) \]
since the associated nodes are B and D,

And the separator’s potential \( \pi_s = \pi(C) = (1, 1) \)

For the second junction tree J2, we get the following results in case of PB-DPN:
\[ \pi_c = \pi(A, A, C, B) = \pi(A) = (1, 0.5, 0.3, 0.3, 0.3) \]
since the associated node is A,
\[ \pi_c = \pi(A, C, C) = \pi(C | A) = (1, 0.5, 1, 0.5, 1, 0.4, 1, 0.4) \]
since the associated node is C,
\[ \pi_c = \pi(B, C, D) = \pi(D | B, C) = (1, 0.6, 1, 0.2, 0.2, 1, 0.1, 1) \]
since the associated node is D,
\[ \pi_c = \pi(B, B) = \pi(B) = (1, 0.5, 1, 0.5) \]
since the associated node is B.

Separator’s potentials are all initialized to the value 1:
\[ \pi_{s1} = (1, 1, 1, 1), \pi_{s2} = (1, 1), \pi_{s3} = (1, 1) \]

Note here that in case of a MB-DPN, we get the same results shown above since only one variable is associated to each clique. Next subsection gives details about the propagation process.

D. Propagation process

With the BK independence assumptions, the joint distribution on the outgoing interface can be decomposed into the product of the joint distributions on the BK clusters in case of PB-DPN (respectively the minimum in case of MB-DPNs). The clusters must be disjoint sets and their union must form the original outgoing interface, i.e., together they must form a partition of the outgoing interface defined in [14] for exact inference. Also, for best results, no cluster should affect any other cluster within the same time-slice [18]. That is, within a time-slice, no node in any one cluster should have as parent a node in a different cluster. For example, the outgoing interface in the 2-TBN in figure 3(a) can be factored using two BK clusters: \{A, C\} and \{B\} as shown in figure 3(b) (section III of this paper).

Once the junction trees have been constructed and initialized, inference is performed through two stages of message-passing, as described by the standard Interface algorithm (Forwards pass and backwards pass). Each clique containing an outgoing BK interface in time-slice \( t \) is called in-clique ((A, A, C) and (B, B) in figure 5(d)), while the clique containing an outgoing BK interface in slice \( t \) is called the out-clique ((A, C, C) and (C, B, D) in figure 5(d)).

Algorithms 5: Approximate inference

Input: J, J. Output: possibilities distribution

1. for each BK out-clique \( i \) \[ \alpha \leftarrow 1 \]
2. \[ \{\alpha_i\} \leftarrow \text{Forwards}(J, \alpha_i) \] running forwards pass on J, using the initial \( \alpha \) values and computing the new values
3. if \( t = 0 \) Backwards(J, \( \alpha_i \) : i = 1 .. number(BK out-cliques))
else for \( i = 1 \) to \( t \)
   \[ \{\alpha_i\} \leftarrow \text{Forwards}(J, \alpha_i) \] induction (since time is incremented)
   end for
Backwards(J, \( \alpha_i \) : termination (computing the final possibilistic distribution)

end if

Once inference has been completed on the junction tree for time-slices \((t-1, t)\) and the algorithm is ready to advance, each out-clique potential is marginalized down to the corresponding outgoing BK interface potential, the 2-TBN is advanced to time-slices \((t, t+1)\), and is multiplied onto the in-clique potential in the new 2-TBN (respectively minimized in case of MB-DPN). This procedure is shown in figure 6.

Since we would be repeating the same junction tree construction steps for each time-step, we can simply build the junction tree once and use it for all time-steps for which inference is performed. Thus, when time is incremented in the advance step above, the junction tree is re-initialized to its initial clique potentials.

Details of forwards pass and backwards are shown in algorithms 5 and 6. Let \( C \) be the clique in J, containing the ou-I of time slice \( t \).

Algorithm 5: Forwards pass

Input: junction tree J, initial CPD, \( V \) :set of ordered cliques, a vector of values \( \alpha_i \) related to BK out-cliques in time \( t-1 \)
Output: interface potential \( \alpha_t \)
1. Root $\leftrightarrow C$: (the root is generally the last clique in $V$
2. for each BK in-clique $C_i$ in $J_t$
   Case of PB-DPN: $\pi_{C_i} \leftarrow \oplus (\alpha_{C_i}, \pi_{C_i})$
   Case of MB-DPN: $\pi_{C_i} \leftarrow \min(\alpha_{C_i}, \pi_{C_i})$
3. for each clique $C_i$ from $V$, for each neighbour clique $C_j$
   updating the separator($C_i, C_j$) potential: $\pi_{C_i} \leftarrow \max(\pi_{C_i}, \pi_{C_j})$
   updating the $C_i$ clique’s potential : $\pi_{C_i} \leftarrow \min(\pi_{C_i}, \pi_{C_j})$
   end for
4. for each BK out-clique $C_t$
   updating clique’s potentials : $\pi_{C_t} \leftarrow \max(\pi_{C_t}, \pi_{C_j})$
   updating separator’s potentials:
   end for
4. for each BK out-clique $C_t$
   Marginalize $C_t$ potential down to the outgoing interface $I_t$
   $\alpha_{C_t} \leftarrow \max \pi_{C_t}$

Once the forwards pass is run many times as needed, we get the final potential’s cliques and since we can update the possibility degrees for each node in that time-slice by the mean of the backwards pass. The backwards pass is same to the one used in a static PN; distribution of evidence from the root of propagation to other cliques.

This process is not needed for each time-slice and can be performed only for the time-slice $t$ (current time-slice) in case of filtering and for time-slice $T (T>t)$ in case of prediction. Details of this process are shown in algorithm 6. This algorithm uses many concepts of junction tree technique [8], [9] used for propagation in probabilistic Networks.

Algorithm 6: Backwards Pass

Input: $J_t$, initial CPD, V clique’s potentials, Output: possibilities distribution
1. for $i = \text{length (} V) \text{ to 2}$
    repeat
    $C_i \leftarrow$ adjacent clique ($C_i$)
    $k \leftarrow n_i$
    updating clique’s potentials : $\pi_{n_i} \leftarrow \max(\pi_{n_i})$
    updating separator’s potentials: $\pi_{C_i} \leftarrow \oplus(\pi_{C_i}, \pi_{C_j})$
    until (all adjacent clique’s potentials are updated)
    end for
2. for $i = 1$ to $N$ (N : number of variables in time-slice $t$)
    Computing the joint possibility for each variable $A_i$ given evidence $Y_t$
    $\Pi(A_i, Y_t) \leftarrow \max \pi_{C_i}$
    Case of PB-DPN
    Computing conditionals: $\Pi(A_i | Y_t) = \frac{\Pi(A_i, Y_t)}{\Pi(Y_t)}$
    Case of MB-DPN
    $\Pi(A_i | Y_t) \leftarrow \begin{cases} \Pi(A_i, Y_t) & \text{if } \Pi(A_i, Y_t) \geq \Pi(Y_t) \\ 1 & \text{otherwise} \end{cases}$

E. Example of propagation

Starting with the DPN shown in figure 2, we want to make filtering just after two times

Time = 0, inference is performed on the first junction tree $J_1$. We get the following results:

1) Case of PB-DPN

$\pi_{C_1} = \pi(A_i, C_1) = (1, 0.5, 0.3, 0.12)$
$\Rightarrow \alpha_{C_1} = \alpha_{C_1}^{\text{min}} = (1, 0.5, 0.3, 0.12)$
$\pi_{C_2} = \pi(B, C_2) = \max(\pi_{C_2}) = (1, 0, 0.2, 0.1, 0.5, 0.015, 0.15)$
$\Rightarrow \alpha_{C_2} = \alpha_{C_2}^{\text{min}} = (1, 0, 0.2, 0.1, 0.5, 0.015, 0.15)$
$\Rightarrow \alpha_{C_2} = \alpha_{C_2}^{\text{min}} = (1, 0, 0.2, 0.1, 0.5, 0.015, 0.15)$
$\Rightarrow \alpha_{C_2} = \alpha_{C_2}^{\text{min}} = (1, 0, 0.2, 0.1, 0.5, 0.015, 0.15)$

Time = 1, time is incremented inference is performed on the second junction tree $J_2$. We get the following results:

1) Case of PB-DPN

Updating potentials of cliques containing the BK clusters:

$\pi_{C_1} = \pi(A_i, C_1) = (1, 0.5, 0.3, 0.12)$
$\Rightarrow \alpha_{C_1} = \alpha_{C_1}^{\text{min}} = (1, 0.5, 0.3, 0.12)$

$\Rightarrow \alpha_{C_1} = \alpha_{C_1}^{\text{min}} = (1, 0.5, 0.3, 0.12)$

$\Rightarrow \alpha_{C_1} = \alpha_{C_1}^{\text{min}} = (1, 0.5, 0.3, 0.12)$

$\Rightarrow \alpha_{C_1} = \alpha_{C_1}^{\text{min}} = (1, 0.5, 0.3, 0.12)$

Hence, the new $\alpha_{C_1}$ which may be used in a next iteration are

$\alpha_{C_1} = \alpha_{C_1}^{\text{min}} = (1, 0.5, 0.3, 0.12)$

Since we want to do filtering at the second time we should next run the backwards pass on $J_1$ using the above results and etc. We note here that the process of prediction is similar to filtering. We have to extend the forwards pass to a time $T>t$ where $t$ is the current time. The same process described above can be used also in the case of a min-based dynamic possibilistic networks with minor changes as described in the above algorithms.

IV. CONCLUSION

In this paper we have presented a generative possibilistic framework for modeling with dynamic networks. We
presented dynamic possibilistic networks and evidence propagation with the PBK algorithm and indicated similarities and differences to probabilistic graphical models. Possibilistic approaches serve for the approximate modeling of uncertain and imprecise data. Therefore, probabilistic and possibilistic graphical models are useful in quite different domains of knowledge representation, which makes them cooperative rather than competitive. Approximate inference with the PBK algorithm, described in this paper for the two versions of possibilistic networks, can be performed for any kind of DPNs (possibilistic HMM, SSM, etc). The accuracy of the PBK algorithm depends on the clusters that we use to approximate the belief state. Exact inference corresponds to using a single cluster, containing all the interface nodes. The most aggressive approximation corresponds to using D clusters, one per variable; this is the fully factorized approximation.

Although the proposed approach requires results for a real application, it provides an intuitive graphical framework for expressing uncertain knowledge. Our aim is to validate our algorithm in a pattern recognition task. We also aim to improve this algorithm by introducing the thin junction tree technique. Another topic of future work is to study in which way possibilistic data, obtained from expert knowledge or databases of sample cases, can be combined and then be represented as the quantitative part of a unified type of graphical model.

ACKNOWLEDGMENT

The authors are grateful to Salem Benferhat and Nahla Benamor for useful discussions relating to this work. The authors wish also to thank the anonymous reviewers for their helpful comments and suggestions.

REFERENCES