

# DISTRIBUTED BAYESIAN NETWORKS IN HIGHLY DYNAMIC AGENT ORGANIZATIONS

Patrick de Oude      Jan Nunnink      Gregor Pavlin

*Informatics Institute, Faculty of Science  
University of Amsterdam, The Netherlands*

## Abstract

In this paper we focus on the problems associated with distributed approaches to exact belief propagation in multi agent systems. In particular, we discuss the Multiply Sectioned Bayesian networks (MSBN) and Distributed Perception Networks (DPNs). While MSBNs support modeling of more complex domains than DPNs, we argue that MSBN approach is not suitable for large and changing agent societies. DPNs, on the other hand, are a special class of MSBNs and have limited modeling capabilities. However, DPN approach facilitates self-organization of distributed systems and, consequently, can cope with variable agent societies.

## 1 Introduction

Recent advances in communication and sensor technology resulted in large quantities of heterogeneous information, which originates from spatially dispersed sensory systems, mobile phone networks, databases, etc. Such a large body of information can be very valuable in the context of complex controlling, decision making and prediction processes, such as crisis management applications, meteorological observation systems, etc. In such applications we reason about some hidden events through interpretation of heterogeneous and noisy evidence. Thus, we need a mapping between evidence and some hypothesis about a hidden event, which requires adequate world models. In particular, we are interested in interpretation with the help of Bayesian Networks (BNs), which supports reasoning with uncertain information. However, interpretation of large quantities of heterogeneous and uncertain information in real world applications can be very challenging: **1.** Required world models can be complex. **2.** We deal with large amounts of information, and consequently, a central model might not be appropriate due to the processing and communication overload at the central processing unit.

In this context, distributed BNs seem to be promising in different ways. Namely, given such a system different experts or machine learning processes could supply partial world models, basic building blocks, that could be assembled into complex causal models. In addition, the belief propagation can be distributed over several processing nodes, which can reduce the communication as well as the processing load at a single node (i.e. partial parallelization).

However, distribution of the belief propagation processes introduces substantial problems w.r.t. the assembly of meaningful distributed BNs as well as distributed belief propagation. Namely, belief propagation in BNs imposes constraints on the assembly of partial world models. Also, consistent belief updating in distributed systems must consider the network topology as well as node instantiations throughout a set of distributed BNs, which can be very challenging if the nodes contributing partial world models can frequently join or leave organizations of fusion nodes at runtime.

We illustrate these problems through the discussion of two approaches to distributed modeling and exact belief propagation in BNs. It turns out that the configuration of the fusion systems as well as distributed belief propagation require complex verification, communication and control as well as concurrent processing. In order to be able to deal with such complexity in a systematic manner, we consider Multi Agent System (MAS) paradigms, where agents represent processing nodes. In particular, we analyze two approaches to distributed belief propagation in MAS, the Multiply Sectioned Bayesian Networks (MSBNs) [5] and Distributed Perception Networks (DPN)

[4], a special class of MSBNs. In this analysis we consider the model complexity as well as the variability of the agent societies.

The general MSBNs support modeling of very complex processes, while DPNs allow reasoning based on models featuring relatively simple topologies. We show that each approach has certain limitations and is suitable for a certain class of applications only. In particular, we argue that it is very difficult or even impossible to achieve robust exact belief propagation if the distributed models are complex (i.e. multiply connected) and the modeling components are distributed throughout societies of agents that can change frequently. Namely, dealing with arbitrarily complex models requires partial synchronization and massive messaging as well as processing in order to establish an adequate belief propagation infrastructure, such as Junction Trees. Typically, compilation of junction trees is computationally not trivial as it requires certain time and partial synchronization. If the agent constellation is changing faster than the compilation process is completed, then we might not be able to generate a suitable inference system at all.

On the other hand, there exist problems that can be solved without the compilation of JTs, which allows very robust self-organization of distributed BNs as well as inference. However, such systems are inherently simple and often do not support modeling of processes as complex as the full MSBNs.

## 1.1 Distributed Perception Networks (DPNs)

Distributed Perception Networks (DPN) [4, 1] are multi agent systems which fuse information through Cooperative Distributed Problem Solving [2]. DPNs are used in applications that introduce very challenging problems: **1.** Relevant information sources, which are not known prior to the operation, must be found and integrated at runtime. **2.** Each hypothesis requires a specific causal model that captures the current constellations of information sources, which are not known prior to the fusion process and change frequently at runtime. Consequently, it is usually impossible to generate adequate models prior to the operation. **3.** The available information is often very uncertain/subjective. **4.** We deal with large amounts of information, and consequently, a central model might not be appropriate due to the processing and communication overload at the central processing unit.

DPN is a logical layer which allows quick discovery of the information sources relevant for the reasoning about a given hypothesis. DPN agents wrap heterogeneous information sources and integrate them into arbitrarily complex inference systems. In other words, DPN agents implement infrastructure which allows efficient configuration of information channels between the information sources and users.

Moreover, DPN agents implement distributed probabilistic inference which supports reliable fusion of very uncertain/subjective information; i.e. automatic information filtering. In particular, DPN can support reliable inference also in the applications where exact modeling parameters are not known. DPN agents provide the means to keep partial processing close to the spatially dispersed information sources, which reduces the danger of communication and processing bottlenecks.

In general, agent-based paradigm supports a systematic organization of complex functionality, required for the self organization as well as distributed inference.

Agents in the DPN framework can roughly be classified into two types. At the lowest level there are different Sensor agents with a direct access to sensors and data interpretation capabilities while at the higher levels Fusion agents use local causal models for partial information fusion. The local world models in Fusion agents are encoded through Bayesian networks and represent basic world modeling building blocks. See Figure 1 for an example DPN.

Each fusion task, i.e. reasoning about a particular hypothesis, requires a specific world model that maps the currently available evidence from the relevant information sources to some hypothesis. Since we assume that the information sources are not known in advance, we must be able to generate such models at runtime. DPN agents achieve this by assembling local probabilistic models into a distributed BN.

Each DPN agent has a service concept corresponding to an event that the agent is reasoning about; i.e. each agent updates its belief in hypothesis about a single hidden event. The belief updating is based on inputs corresponding to the so called input concepts, which in turn are identical to service concepts of other DPN agent types. Agents A and B can integrate their local models if the service concept of one agent intersects an input concept of another agent. In other

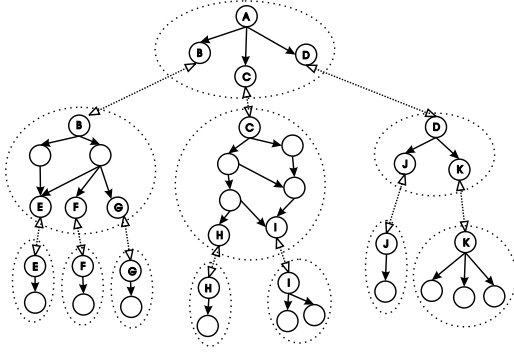


Figure 1: A DPN network.

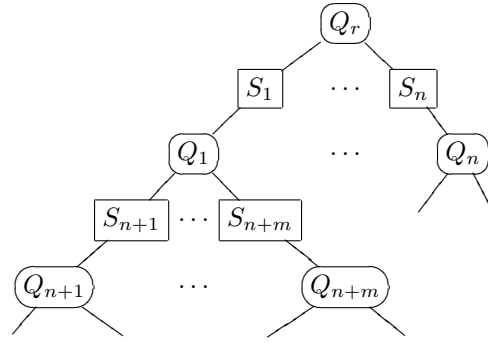


Figure 2: A DPN domain graph.

words, through the local agent ontologies we specify beforehand *what* types of agents can exchange information. But, we never know in advance *which* agent instances are actually going to participate in the distributed BN (see [4]).

Moreover, we focus on applications where agent societies change frequently, which makes centralized configuration control or central knowledge of existing agents impractical. However, given the local world models we can derive a rule which supports local self-configuration based on cooperation between individual agents without any centralized control mechanism:

*After some agent A has become a member of a particular DPN, it starts looking for other agents whose output concepts correspond to its input concept.*

This rule introduces a concept driven configuration process, which exploits the fact that causal links, which should be established between the partial world models correspond to intersections of local ontologies. In addition, this rule introduces a partial order of possible link creations. In other words, by using this rule we can build hierarchical systems in a top down manner through hierarchical service discovery.

Thus, we can generate a query about the state of a variable of interest and send it to a community of agents, which will organize a meaningful DPN fusion system that integrates all relevant agents and information sources for a given fusion task. In other words, DPN features a Concept driven self configuration process, which allows a great flexibility through adaptive combination of standard building blocks.

## 2 MSBNs

The DPN framework is a special class of the more general Multiply Sectioned Bayesian Networks (MSBNs) presented in [5]. In this section we will summarize the MSBN representation and the analysis of the design restrictions.

A representation imposes a set of restrictions on the design and operation of a system, and at the same time guarantees a set of properties. An efficient representation causes properties which are all strictly required by the problem class, while being as unrestrictive as possible.

We will now give the definition of an MSBN as presented in [5].

**Definition 1** Given two graphs  $G_0 = \langle N_0, E_0 \rangle$  and  $G_1 = \langle N_1, E_1 \rangle$ , the union is defined by  $G = \langle N_0 \cup N_1, E_0 \cup E_1 \rangle$  and denoted by  $G = G_0 \sqcup G_1$ .

**Definition 2 (hypertree)** Let  $G = \langle N, E \rangle = \bigsqcup_i G_i$  be a graph sectioned into subgraphs  $G_i = \langle N_i, E_i \rangle$ . If the subgraphs can be organised as a connected tree  $\Psi$ , where each node is labeled by a  $G_i$  and each edge between  $G_a$  and  $G_b$  is labeled by  $N_a \cap N_b$ , such that for each  $i$  and  $j$ ,  $N_i \cap N_j$  is contained in each subgraph on the path between  $G_i$  and  $G_j$  in  $\Psi$  (i.e. the tree has the running intersection property), then  $\Psi$  is a hypertree over  $G$ .

**Definition 3 (d-sepset)** Let  $G$  be a directed graph such that a hypertree over  $G$  exists. A node  $x$  contained in more than one subgraph is a d-sepnode if there exists a subgraph that contains the parents  $\pi(x)$  of  $x$  in  $G$ . A d-sepset is a set of d-sepnodes.

**Definition 4 (hypertree MSDAG)**  $\mathcal{D} = \bigsqcup_i D_i$ , where each  $D_i = \langle N_i, E_i \rangle$  is a directed acyclic graph (DAG), is a hypertree MSDAG iff (1) there exists a hypertree  $\Psi$  over  $\mathcal{D}$ , and (2) each link in  $\Psi$  is a  $d$ -sepset.

**Definition 5 (MSBN)** An MSBN is defined over a set of variables  $\mathcal{N}$  called the domain, and consists of a finite set of agents  $\mathcal{A}$ . Each agent  $A_i$  can reason about a subdomain  $N_i$  such that  $\mathcal{N} = \bigcup_i N_i$ , and is structured as a DAG  $D_i$ . The complete MSBN is structured as a hypertree MSDAG  $\mathcal{D} = \bigsqcup_i D_i$ , where each node is labeled by one  $D_i$ . Furthermore, each agent defines a joint probability table (JPD)  $P_i$  over  $N_i$ , such that  $P_i = \prod_{x \in N_i} P(x|\pi(x))$ , where  $\pi(x)$  are the parents of  $x$  in  $A_i$ . For each shared variable  $x$ , only one agent which contains  $\pi(x)$  specifies the correct conditional probability distribution (CPD)  $P(x|\pi(x))$ , and all other agents specify a uniform CPD. The JPD over  $\mathcal{N}$  is defined as  $\mathcal{P} = \prod_i P_i$ .

It is clear that this representation is quite restrictive, especially w.r.t. the allowed agent network structure. The following is a list of the major restrictions [6]: **1.** Beliefs are represented using probabilities. **2.** The complete domain is sectioned into subdomains, and through shared variables the agents form a connected network. **3.** This agent network is structured as a tree. **4.** The agent tree satisfies the running intersection property, i.e. is a hypertree. **5.** Each subdomain is structured as a DAG. **6.** The union of the DAGs of all subdomains is a connected DAG. **7.** Each edge in the hypertree is a  $d$ -sepset. **8.** The JPD over the domain is specified as in Def. 5.

## 2.1 Analysis

In this analysis it is shown that the MSBN restrictions follow directly from a set of very basic assumptions and required properties (for more details, see [5]).

The first assumption is rather trivial:

**BA 1** An agent uses probabilities to represent its belief.

This implies restriction 1, and indirectly requires the system to perform exact propagation.

**BA 2** Two agents  $A_i$  and  $A_j$  can directly communicate only about their shared nodes,  $N_i \cap N_j$ .

This assumption is needed to keep message passing between agents efficient and concise. This message domain,  $I = N_i \cap N_j$ , is called the *separator* or *interface* between agents  $A_i$  and  $A_j$ . The agents and interfaces define a connected *communication graph* (CG) (restriction 2).

**BA 3** A tree structured agent network is preferred.

This assumption is required for globally consistent belief propagation. The CG described above is also a *junction graph*. In junction graphs, consistent belief updating is in general only possible if the graph is equivalent to a tree. Trees are therefore uniformly preferred (restriction 3). The tree also needs to have the running intersection property (see Definition 2) making it a hypertree (restriction 4).

**BA 4** Each agent structures its internal model as a DAG.

DAGs are a representational choice as they are very concise models for domains with causal relationships. This implies both restriction 5 and 6. Belief propagation also requires each separator to be a  $d$ -sepset (restriction 7).

**BA 5** Agents combine their knowledge on shared nodes.

Since the JPD over the complete DAG has to be equal to the product of agents' JPDs, the JPDs have to be specified conform restriction 8.

## 2.2 Compilation

Before inference in a hypertree MSDAG is possible the structure has to be compiled into a *Linked Junction Forest* (LJF) with *Linkage Trees* (LT) as communication channels before belief updating through concise message passing is possible (see [5]). The compilation process contains tree steps, namely **1.** moralization of hypertree MSDAG, **2.** triangulation of moral graph, and **3.** compiling to LJF.

The compilation procedure is somewhat similar to the traditional way of organizing a DAG into a JT. First, all the agents has to be moralized by *cooperative distributive moralization* after doing local moralization within the agent itself. Cooperative distributive moralization between agents is necessary to do correct moralization over the whole hypertree MSDAG and means that newly

created undirected edges are communicated between adjacent agents if both connected variables are in the agent’s interface. The main algorithm is **CoMoralize** that calls first **CollectMlink** and after that **DistributeMlink**. **CollectMlink** is a recursive algorithm that collect links from adjacent agents. If all the links are collected then the recursive **DistributeMlink** is called. In this recursive algorithm all the adjacent agents get the collected links communicated. After **DistributeMlink** is finished all the agents are correctly moralized.

Second, the moralized hypertree MSDAG now needs to be triangulated through *cooperative distributive triangulation*. Triangulation is performed by elimination using a constraint on the elimination sequence of variables. It will eliminate all the variables  $V_i \setminus I_i$  where  $V_i$  are variables in agent  $A_i$  and  $I_i$  is the agent  $A_i$ ’s interface to another adjacent agent and then it will eliminate variables in  $I_i$ . The reason for this constraint is to be able to compute a LT when compiling to a LJF. The triangulation procedure uses an algorithm called **CoTriangulate** that calls the algorithms **DepthFirstEliminate** and **DistributeDlink** (see [5]). **DepthFirstEliminate** goes through all the agents in a depth first manner. If **DepthFirstEliminate** finishes the recursive algorithm **DistributeDlink** will be called and communicates the added edges to the adjacent agents. Sometimes it is required to rerun **CoTriangulate** because not every local moral graph is fully triangulated, which mean that new *fill-ins* can be introduced when using the elimination sequence  $(V_i \setminus I_i, I_i)$ .

Last, the triangulated graphs have to be organized into a LJF to do effective inference with concise message passing. This is done by first organizing all the local triangulated graphs into their JT representation. This transformation can be done locally without any knowledge from other adjacent agents. Because local JTs in a LJF are connected through LT they must be computed from an agent’s JT. This can be done for all agents and the final structure will be a LJF.

## 2.3 Inference

The multiagent belief communication in a LJF is organized as follows: **CommunicateBelief** is the main algorithm for belief propagation and calls first the algorithm **CollectBelief** and after that the algorithm **DistributeBelief**. **CollectBelief** recursively propagates belief inwards from terminal agents towards an initiating agent, where **CollectBelief** is analogous to **CollectEvidence** for the single agent paradigm proposed by [3]. **DistributeBelief** recursively propagates belief outwards from the initiating agent to the terminal agents and is analogous to **DistributeEvidence** also proposed by [3]. Running **CommunicateBelief** will bring the LJF into global consistence. So in case of observations made by the agents the evidence is injected into the network with **EnterEvidence** and will make the LJF globally inconsistent. Now by running **CommunicateBelief** it will return in a globally consistent state.

## 2.4 Organizational Issues

Recall from the introduction that we are targeting applications which are characterized by highly dynamic agent networks and relatively high probability of agent breakdown. Therefore, flexible and fast self-organization is required. From the MSBN description above it is clear that several aspects of MSBN construction are not well suited for these kinds of situations. In particular, constructing the LJF and checking for a valid network structure (hypertree MSDAG) can be problematic, since they require massive message passing and partial synchronisation. This causes construction time to depend significantly on the size of the agent network, and during this time the network is not allowed to change, or the process has to be restarted. Clearly, this is very likely to happen in our kind of applications. In the next section we will present a framework that counteracts these problems, however at the expense of not being able to handle models of arbitrary complexity.

# 3 Distributed Bayesian Networks in DPNs

DPNs do not require any compilation of LJFs and, therefore, are suitable for applications involving dynamic agent constellations. BNs in DPNs are assembled out of smaller DAGs at runtime. DPNs implement a special, more restrictive, case of MSBNs. The local DAGs are stored in different agents and satisfy all conditions/design restrictions of MSBNs described in the previous

section. However, in order to facilitate the self organization of DPN fusion structures, we introduce additional constraints.

### 3.1 Self Organisation Principles

We are dealing with problems where the models must be assembled at runtime through the self-organisation of DPN agents. In order to facilitate automated generation of distributed BNs that support exact inference, we introduce a few additional structural constraints that have to be considered when adding or substituting a new agent with its own problem domain. Let an agent's subdomain be denoted by a *cluster* of variables. An agent can only be connected to one parent agent:

**Definition 6 (DPN Constraints)** *Given a DPN with a set of clusters  $\mathcal{Q} = \{Q_1, \dots, Q_n\}$  that are directly or indirectly connected, a set of separators  $\mathcal{S} = \{S_1, \dots, S_n\}$  and a new cluster  $Q_i$ . Cluster  $Q_i$  can be connected to  $Q_j \in \mathcal{Q}$  with separator  $S\langle Q_i, Q_j \rangle$  if the following rules are satisfied:*

- i)  $Q_i \cap Q_j \neq \emptyset \wedge \forall k \neq j, i, Q_i \cap Q_k = \emptyset$
- ii)  $\forall S_k : S_k \cap S\langle Q_i, Q_j \rangle = \emptyset$  where  $S_k \in \mathcal{S}$
- iii)  $|Q_i \cap Q_j| = 1$

The first rule i) is the *intersection constraint*. This means when an agent is connected to a DPN its cluster may only have overlap with one other cluster of an agent. The second rule ii) is the *uniqueness constraint* and states that the newly added separator  $S\langle Q_i, Q_j \rangle$  is uniquely defined given the separator set  $\mathcal{S}$  of the DPN. The last rule iii) is the *separator constraint* and simply states that all separators should have size 1. Any cluster that is a candidate for assembling should satisfy all constraints otherwise it is rejected and not connected to the DPN.

These design constraints imply the following proposition:

**Proposition 1 (Preserving Junction Tree)** *If the assembly rules are such that a newly added cluster satisfies constraints in Definition 6, the resulting distributed model will automatically correspond to a junction tree.*

Each agent represents exactly one cluster. That is, the variables contained in the local BN of an agent form the domain of one cluster in the domain graph. The consequence of DPN assembly rules reflecting the DPN design constraints are special DPN separators, DPN local DAGs as well as DPN domain graphs.

**Definition 7 (DPN Separator)** *Given clusters  $Q_i$  and  $Q_j$  the separator  $S\langle Q_i, Q_j \rangle$  is defined by*

$$S\langle Q_i, Q_j \rangle = Q_i \cap Q_j, \quad \text{where } |S\langle Q_i, Q_j \rangle| = 1, \quad (1)$$

*i.e. each separator has size one and is uniquely defined. In addition, it is obvious that a DPN separator is a d-sepset (see Definition 3).*

**Definition 8 (DPN Local DAG)** *Each agent can contain an arbitrary BN. However, only one root node can be considered a service concept and included in a DPN separator.*

Given this definition and the DPN assembly rules we know that every pair of separators has an empty intersection.

**Definition 9 (DPN Domain Graph)** *A DPN domain graph has a tree structure with a root cluster  $Q_r$ , which contains at least the domain of the hypothesis variable, and a set of other clusters  $\mathcal{Q} = \{Q_1, \dots, Q_N\}$ . Any pair of clusters  $Q_i$  and  $Q_j$  is connected if a separator exists according to Definition 7. (see Figure 2)*

Definition 7 to 9 defines the DPN structure the following definition will tell us about the potentials of a DPN.

**Definition 10 (DPN Potentials)** *Given a set of variables  $\mathcal{V}$  which defines the universe of a DPN problem domain and a set of variables  $\mathcal{R} \subset \mathcal{V}$  which defines all the root variables in the local BNs. The potentials of  $\mathcal{V} \setminus \mathcal{R}$  are defined according to the CPTs and the potentials of  $\mathcal{R}$  are defined as uniform distributions.*

## 3.2 Distributed Inference in DPNs

Distributed inference in DPNs are only focusing on diagnostic reasoning, because we are only interested in the root hypothesis of the full connected DAG. This means that we will never calculate the marginal probabilities of non local root variables. Moreover, consistent reasoning is possible by collecting the locally computed marginal probabilities of the local root variables of the participating DPN agents in a bottom-up manner [1]. The reason that we can do inference in this way is because of Proposition 1.

The used algorithm to do inference in a DPN uses a hybrid approach, a combination of algorithms for local inference whose partial results are combined via an inter-agent messaging process. While the details about the DPN belief propagation algorithm can be found in [1], we would like to emphasize that this approach exploits the fact that DPN agents integrate their partial world models in such a way that inference through local processing and message passing is possible. This algorithm supports *exact* inference.

## 4 DPN justification

In this section we will justify the DPN structure and show why the more general MSBNs are not suitable for our type of problems. We would like to emphasize that DPNs are meant to handle domains where the agent network structure can change rapidly, and fast self (re)organization is a critical feature.

**Configuration:** Dynamic configuration requires that different agents discover other agents with matching interfaces. However, MSBNs can have arbitrarily long interfaces. Therefore, it is difficult to develop a community of agents with matching interfaces without significant coordination between the developers of different agents. In other words, the development of complex MAS either becomes impractical or the chance that agents can cooperate becomes very low.

In a DPN, however, the interfaces are much simpler (see Definition 6, constraint 3) which means that designers merely have to coordinate the names and semantics of the shared concepts. This is significantly simpler than the coordination required for the development of full MSBNs.

**Compilation:** MSBNs require compilation of the hypertree MSDAG which mean that the hypertree has to be moralized, triangulated, the LTs and local JTs has to be computed in order to build a LJF before belief propagation is possible. Because of the dynamic topology of a DPN recompiling the network would not be feasible. The reason for this is that compiling the network requires some time to finish. During compilation time the topology of the DPN can again be changed which would require recompilation in order to do belief propagation. It turns out that compiling a DPN on the agent level is not required, because given the DPN constraints in Definition 6 we know by Proposition 1 that our network already corresponds to a junction tree. Because of the fact that DPNs do not need to be compiled in order to do inference it also prevents the communication and computation costs of compiling.

**Propagation:** In DPNs we are only interested in the root hypothesis which means that we are not calculating the marginal probabilities of non root variables. Because of this we are only interested in doing diagnostic reasoning. This means that the information flow is only bottom-up and avoids communicating evidence through the whole network. In addition, propagating belief throughout the whole network would not be trivial in a rapidly changing network.

**Interfaces:** Like for MSBNs, the interfaces between DPN agents must be d-sepsets. However, for MSBNs the task of finding and selecting a valid network topology is not trivial, and infeasible for highly dynamic networks. In DPNs, Definition 6 states that a node can only be shared by two agents. Definition 8 states that in one of the two agents the shared node is a root of the local DAG. This implies that all parents of the shared node are in the other agent, which makes every DPN-separator automatically a d-sepset (Definition 3).

**Hypertree:** Similar to checking for d-sepset interfaces, one must verify that the agent network is a hypertree; a procedure which is problematic in dynamic agent networks. However, Definition 6 states that the domain of each agent can only have a non-empty intersection with the domains of adjacent agents, and thus a DPN is guaranteed to be a hypertree.

**Shared Belief:** The MSBN definition forces agents to combine their local (conditional) probabilities on shared nodes in order to obtain a consistent JPD (see restriction 8). This requires extra

coordination between agents each time the agent network is (re)configured, which causes problems if the network changes its topology often. As explained above, shared nodes in DPNs are always a root in one of the two agents' DAGs. Definition 10 states that root nodes in a DPNs are initialized with uniform prior distributions, causing the JPD to be automatically consistent and therefore no extra coordination is needed.

**Types of DPN Applications:** DPNs are used in a class of applications where we reason about a single hypothesis through a fusion of large amounts of heterogeneous information. In other words, we do not compute the marginal distribution over all variables in a BN. In addition, the targeted domains can be described through models with a significant portion of conditionally independent branches, which can easily be modeled by DPNs and the complexity of the general MSBNs would not be justified. For example, consider a system for estimating the presence of toxic gases in environments with a significant hazard of escaping toxic gases. We could use special chemical sensors as well as reports obtained from humans via mobile phone networks or databases. The estimation system would use a BN with a simple tree topology featuring several abstraction levels. The hypothesis about the presence of the gas would be captured in a single root node and several conditionally independent branches would model the relations between the presence of gas and the symptoms. Typically, such symptoms could be chemical sensor reports, particular smell, coughing, etc. The decision makers specify appropriate decision thresholds for different states of the root concepts. Thus, distributions over the states trigger different decisions, dependent on the exceeded threshold. In such an application we expect large amounts of sensor reports or human observations, which results in a BN topology with large branching factors.

## 5 Conclusion

We have presented DPNs, a framework for probabilistic reasoning in distributed systems, and shown that it is a special class of MSBNs. We have argued that while MSBNs can handle problem domains of high complexity, in the case of highly dynamic agent societies they are impractical to use, primarily because of the costs of compiling a MSBN into an LJP to enable inference. Rapidly changing agent societies do not allow the structure to undergo compilation before allowing inference.

Dynamic agent societies require a model where compilation and organization is fast and immediate inference is supported. DPNs, in contrast to MSBNs, allow immediate inference by avoiding compilation of the model. This is possible because additional constraints are defined on the structure of a DPN. The trade-off of these additional constraints is that DPNs are limited to problems that can be described by models with a significant portion of conditionally independent network fragments and only diagnostic reasoning about a top level hypothesis is allowed. Given these limitations DPNs are still usable in a significant class of problems.

## References

- [1] P. de Oude, B. Ottens, and G. Pavlin. Information fusion in distributed probabilistic networks. In *Artificial Intelligence and Applications*, pages 195–201, Innsbruck, Austria, 2005.
- [2] E. Durfee, V. Lesser, and D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, 1989.
- [3] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.
- [4] G. Pavlin, M. Maris, and J. Nunnink. An agent-based approach to distributed data and information fusion. In *IEEE/WIC/ACM Joint Conference on Intelligent Agent Technology*, pages 466–470, 2004.
- [5] Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.
- [6] Y. Xiang and V. Lesser. Justifying multiply sectioned bayesian networks. In *Proc. of the 6th Int. Conf. on Multi-agent Systems*, pages 349–356, Boston, 2000.