A Multi Agent Systems Approach to Distributed Bayesian Information Fusion

Gregor Pavlin[b,a], Patrick de Oude[a], Marinus Maris[a], Jan Nunnink[a], Thomas Hood[b]

[a] *Intelligent Autonomous Systems Group, Informatics Institute*
University of Amsterdam, The Netherlands
[b] *Thales Research & Technology Netherlands*
Delftechpark 24, 2628 XH, Delft, The Netherlands

Abstract

This paper introduces design principles for modular Bayesian fusion systems which can (i) cope with large quantities of heterogeneous information and (ii) can adapt to changing constellations of information sources on the fly. The presented approach exploits the locality of relations in causal probabilistic processes, which facilitates decentralized modeling and information fusion. Observed events resulting from stochastic causal processes can be modeled with the help of Bayesian networks, compact and mathematically rigorous probabilistic models. With the help of the theory of Bayesian networks and factor graphs we derive design and organization rules for modular fusion systems which implement exact belief propagation without centralized configuration and fusion control. These rules are applied in distributed perception networks (DPN), a multi agent systems approach to distributed Bayesian information fusion. While each DPN agent has limited fusion capabilities, multiple DPN agents can autonomously collaborate to form complex modular fusion systems. Such self-organizing systems of agents can adapt to the available information sources at runtime and can infer critical hidden events through interpretation of complex patterns consisting of many heterogeneous observations.

1. Introduction

Decision making in real world domains often critically depends on timely and reliable detection of significant events, such as the presence of toxic gases, disease outbreaks, fires, etc. In such settings, hidden events must be inferred through interpretation (i.e. fusion) of large quantities of uncertain and heterogeneous information. The information can be obtained from static sensors or ad-hoc sensor networks formed at runtime, as sensors are delivered to the location of interest via mobile platforms. In addition, valuable information can be obtained from humans by using conventional communication infrastructure, such as mobile phone networks, Internet, etc. Interpretation of observed patterns requires domain models which provide mapping between observations and hypotheses of interest. In order to be able to draw correct conclusions about hidden events based on observations, the domain models must adequately capture the relations between the relevant variables; i.e., the models must be grounded in the real world. However, obtaining such models can be very challenging because:

- Information sources are heterogeneous and noisy. Dealing with the heterogeneity of observations requires complex domain models. On the other hand, the models are inevitably abstractions associated with significant uncertainties. In addition, information fusion based on complex models requires significant communication and processing resources.

Email addresses: gregor.pavlin@icis.decis.nl (Gregor Pavlin)

Preprint submitted to Information Fusion

- Constellations of information sources are often not known prior to the operation and they change at runtime. On the other hand the models must capture all observations; i.e., each particular fusion process requires a specific domain model which maps observations from the current constellation of information sources to the hypotheses of interest. Consequently, a domain model should be adapted at runtime, as the information sources become available.
- Reliable detection in such settings requires processing of large quantities of noisy information.

Given these challenges, a modular approach to modeling and inference seems to be a good choice; adequate domain models can be assembled from basic modeling blocks at runtime and the processing load can be distributed throughout a system of networked devices.

It turns out that probabilistic causal models facilitate the design of robust and flexible modular fusion systems. Observations can be often viewed as outcomes of causal stochastic processes. Such processes can be modeled with the help of causal Bayesian networks (BN) [25] which provide a theoretically rigorous and compact mapping between hidden events of interest and observable events. By considering the locality of causal relations in BNs and their factorization properties [13], we derive design and organization rules which support creation of multi-agent systems implementing exact belief propagation in distributed fusion systems. The resulting fusion systems do not require any centralized configuration and fusion control. In addition, no secondary fusions structures spanning multiple agents have to be compiled, which allows flexible configuration at runtime; i.e. the resulting systems support hot swapping and plugging of fusion modules. This is achieved through targeted instantiations of variables in combination with local models constructed according to simple design rules. In particular, we exploit Markov boundaries[1] [20] to systematically find efficient instantiation patterns.

The modularization and combination principles are used in Distributed Perception Networks (DPN), a multi-agent fusion architecture. DPN agents are basic building blocks which autonomously form distributed domain models and support decentralized Bayesian fusion through cooperation. Each agent performs a local fusion task and shares its fusion results with other agents. A DPN fusion agent can receive information from various information sources (e.g., sensors, humans) or other fusion agents and computes probability distributions over relevant hypotheses (i.e., belief) by using its local causal models. Outcomes of such local inference processes can in turn be supplied to other fusion agents as input. Depending on the available information sources, DPN agents form a multi-agent system which corresponds to the required causal model. DPN agents thus form a task-specific DPN, which is basically a self-configurable distributed classifier. In terms of [15], DPN agents implement primarily deductive inferencing based on local fusion algorithms which can be applied at different levels of the JDL model.

The presented approach to distributed fusion is complementary to well known approaches to inference with distributed graphical models [29, 17]. In contrast to our method, these approaches require compilation of secondary inference structures, often recursive processes which can be computationally expensive and time consuming. In addition, approach from [17] requires prior knowledge of all information sources. Consequently, these approaches do not support quick adaptation of fusion systems and cannot efficiently cope with domains where information source constellations can change at runtime.

The paper is organized as follows: in section 2 we explain why causal probabilistic models are suitable for certain real-world information fusion problems. We also review important properties of Bayesian networks, such as Markov boundaries, which are central to our approach. In section 3 we introduce basic design principles for distributed fusion systems, which are based on instantiation of variables in Markov boundaries. By using the theory of *factor graphs* we show that agents designed according to these principles can form fusion organizations which support globally correct collaborative information fusion, without centralized configuration and fusion control. In section 4 we introduce Distributed Perception Networks, a multi-agent fusion architecture. Section 5 discusses real world implementation issues, such as robustness with respect to the modeling parameters and assumptions about conditional independences. We also address challenges regarding the generation of complex domain models.

[1] If all variables in a Markov boundary are instantiated (i.e. their states are observed), then the inference based on variables enclosed by this boundary is independent of any inference in the rest of the model.

2. State Estimation with Causal Bayesian Networks

An important capability of intelligent systems is estimation of relevant states in the domain. For the purpose of this paper we assume that such states can be described through finite sets of discrete random variables. For example, values of binary variables could represent the presence and absence of a toxic gas, etc. Moreover, we assume that hidden and observed states of the domain, such as the presence or absence of gas or the content of a report from a chemical detector, materialize through a stochastic causal process, which can be viewed as 'sampling' from some true distribution over the combinations of possible states. In such settings the estimation corresponds to the determination of the probability distribution $P(H|\mathcal{E})$ or equivalently $P(H, \mathcal{E})$ over a set of hypotheses represented by the states h_i of a discrete hypothesis variable H and a set of observations $\mathcal{E}$. Thus fusion corresponds to the computation of the joint probability distribution $P(H, \mathcal{E})$.

Since we can view observable states as outcomes of stochastic causal processes, distributions $P(h_i, \mathcal{E})$ can be computed with the help of causal Bayesian networks (BN) [25]. A Bayesian network is defined as a tuple $\langle \mathcal{G}, P \rangle$, where $\mathcal{G} = \langle \mathcal{V}, \mathbb{E} \rangle$ is a directed a-cyclic graph (DAG) defining a domain $\mathcal{V} = \{V_1, \ldots, V_n\}$ and a set of directed edges $\langle V_i, V_j \rangle \in \mathbb{E}$ over the domain. The joint probability distribution $P(\mathcal{V})$ over the domain $\mathcal{V}$ is defined as

$$P(\mathcal{V}) = \prod_{V_i \in \mathcal{V}} P(V_i|\pi(V_i)),$$

where $P(V_i|\pi(V_i))$ is the conditional probability distribution for node V_i given its parents $\pi(V_i)$ in the graph, which can be represented by a conditional probability table (CPT). When $\pi(V_i) = \emptyset$ then $P(V_i|\pi(V_i))$ reduces to $P(V_i)$. In general, the probability distribution over an arbitrary set of discrete variables can be computed through an appropriate marginalization of $P(\mathcal{V})$.

2.1. Causal Domain Models

By using probabilistic causal models we can often efficiently describe the mechanisms producing observations in a theoretically rigorous and compact way. We illustrate this with the help of an example from the monitoring domain. Let's assume that the presence of a certain toxic gas[2] causes conditions under which a certain type of semiconductors features distinctive conductivity. Similarly, a particular conductivity could be observed in an ionized gas mixture if the toxic gas is present. In this paper we assume two types of sensors, evaluating conductivity in semiconductors and in an ionized gas mixture, respectively. Introduction of a sensor measuring a particular type of conductivity will spawn various processes in the sensor's electronic circuitry which in turn will result in a certain state of the sensor. Dependent on the sensor state we will obtain a sequence of reports, either confirming or refuting the presence of the gas. Similarly, humans are likely to report about typical symptoms (e.g. smell).

Such a causal process can be described through the graph shown in Figure 1, where each node represents a binary variable; e.g. $GasX = true$ if the gas is present, otherwise $GasX = false$. The situation under which a semiconductor element and ionized gas mixture feature typical conductivity is represented by variables $CondC$ and Ion, respectively. States of the binary variable $CondC$ correspond to the situations in which th e electrical current in an ideal semiconductor would either exceed some detection threshold (i.e. $CondC = true$) or remain below that threshold (i.e. $CondC = false$). The states of the i-th sensor of type x are represented by S_i^x while a sequence of sensory reports is represented by binary variables $E_1^x, \ldots, E_n^x$; $E_k^x = true$ if a report confirms the presence of the Gas, otherwise $E_k^x = false$.

In this example, subgraphs containing nodes $S_1^C, C_1^C, E_1^C, \ldots, E_M^C$ and S_2^C, C_2^C, E_{M+1}^C, $\ldots$, E_N^C describe processes in two sensors measuring the conductivity of local semiconductor elements. Subgraph consisting of nodes $S_3^I, C_3^{I1}, C_3^{I2}, C_3^{I3}, E_1^I, \ldots, E_O^I$, on the other hand, corresponds to the third sensor measuring conductivity of the ionized gas mixture. Variables S_1^C and S_2^C denote the measured conductivity on the semiconductor elements in the first two sensors while S_3^I denotes the measured conductivity of the ionized

[2]For the sake of simplicity we say that a toxic gas is present if its concentration exceeds some critical value. Otherwise we say that the gas is absent.

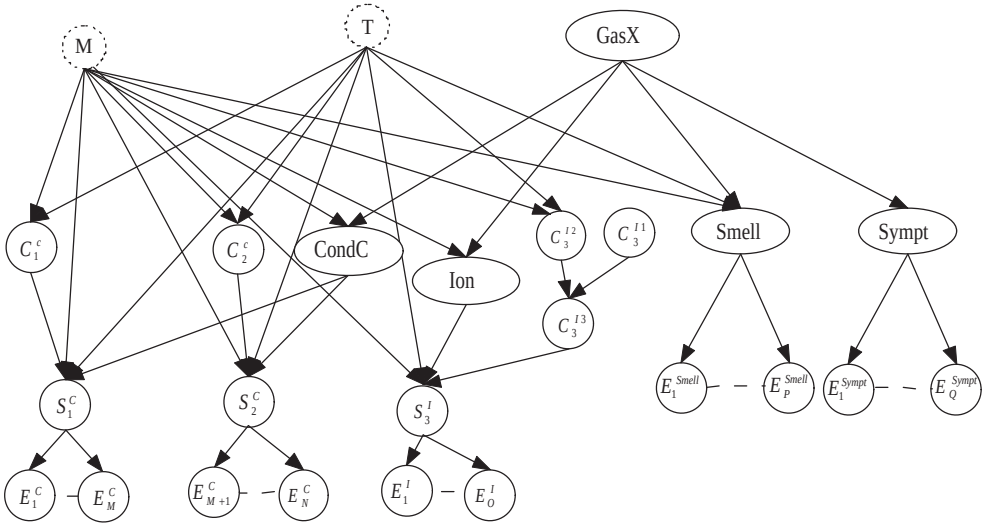


Figure 1: A causal model capturing relations between the states of interest, such as presence/absence of GasX and different types of observations represented by leaf nodes.

gas mixture in the third sensor. We also assume that the causal process is influenced by the air humidity and temperature represented by variables M and T, respectively. Note that with each sensor we introduce an independent local causal process which is initiated through some hidden phenomenon. For example, by introducing a new gas sensor the presence of gas will initiate different processes in the sensor's circuitry which will eventually produce sensor reports represented by leaf nodes in Figure 1. Besides the sensors we assume that there are humans in the area, who have olfactory reactions to $GasX$ and who submit reports of what they smell via a call service or a web-interface. The states of binary variable $Smell$ represent situations in which people familiar with a typical smell of $GasX$ either do or do not recognize the smell. Moreover, each individual report is represented by a node E_i^{Smell}. Similarly, first aid workers might be able to report about health symptoms which are typical results of exposure to $GasX$. A situation in which observable symptoms take place is denoted by variable $Sympt$ and the reports are denoted by variables E_i^{Sympt}.

Stochastic aspects in such a causal process can be described through conditional probabilities. For example, the current will exceed the detection threshold with the probability $P(CondC = true|GasX = true, M, T)$ if the gas concentration exceeds a critical value, i.e., $GasX = true$. Also, with probability $P(CondC = false|GasX = true, M, T)$ the current might not exceed the detection threshold despite $GasX = true$. This might be due to the presence of another gaseous component which would inhibit the desired chemical reaction between $GasX$ and a perfect semiconductor. The probability distribution over states of the variable $Smell$ given $GasX$, T and M is captured by $P(Smell|GasX, M, T)$. Also, since the smelling capabilities are not perfect, we use a probabilistic "sensor" model $P(E_i^{Smell}|Smell)$. Similarly, there exist a probability distribution $P(E_i^{Sympt}|Sympt)$ over correct and erroneous reports on certain symptoms given their presence.

If we know conditional probabilities for all direct influences between pairs of variables depicted in Figure 1, we can express the joint probability $P(\mathcal{V})$ over all variables with the help of a BN which corresponds to the following factorization:

$$
\begin{aligned}
P(\mathcal{V}) = {} & P(GasX)P(M)P(T) \\
& \cdot P(CondC|GasX, M)P(Ion|GasX, M)P(Smell|GasX, T, M)P(Sympt|GasX) \\
& \cdot P(S_1^C|CondC, T, M, C_1^c)P(C_1^c|M, T) \\
& \cdot P(S_2^C|CondC, T, M, C_2^c)P(C_2^c|M, T) \\
& \cdot P(S_3^I|Ion, T, M, C_3^{I3})P(C_3^{I2}|M, T)P(C_3^{I3}|C_3^{I1}, C_3^{I2})P(C_3^{I1}) \\
& \cdot \prod_{m=1}^{M} P(E_m^C|S_1^C) \prod_{n=M+1}^{N} P(E_n^C|S_2^C) \prod_{o=1}^{O} P(E_o^I|S_3^I) \\
& \cdot \prod_{p=1}^{P} P(E_p^{Smell}|Smell) \prod_{q=1}^{Q} P(E_q^{Sympt}|Sympt)
\end{aligned} \tag{1}
$$

Also, the third and the fourth line correspond to two sensors of the same type. Consequently, $P(S_1^C|CondC, T, M, C_1^c) = P(S_2^C|CondC, T, M, C_2^c)$ and $P(C_1^c|M, T) = P(C_2^C|M, T)$. Similarly, $P(E_j^C|S_i^C)$ are identical for all j.

In addition, we distinguish between two types of variables. Variables which are influenced by a causal process are called *Process variables*; i.e., states of *process variables* are outcomes of stochastic causal processes. The fusion can be viewed as estimation of the distributions over the states of *process variables* which are not directly observable. *Context variables*, on the other hand, represent phenomena that influence a causal process of interest while they can be considered independent of the same process. Consequently, in a causal model such variables are always represented through root nodes. For example, Figure 1 suggests that different sensors are influenced by the air temperature and humidity represented by variables T and M, respectively. However, we can safely assume that the sensing processes do not significantly influence the temperature and humidity, especially if we use passive information sources. In other words, *context variables* represent boundary conditions of a casual process.

2.2. Temporal Aspects

The model depicted in Figure 1 corresponds to a single time slice in which a sequence of observations was obtained while the states of non-leaf variables including $GasX$ remained constant; within such a time slice the electronic circuitry of a sensor evaluates the current and generates a stream of sensor reports. We assume that the domain's hidden phenomena are *quasi-static* in the sense that they do not change during a single time slice. For example, in Figure 1 nodes $GasX$ and $CondC$ represent the presence/absence of gas and increased conductivity of the semiconductor, respectively. If gas is present then we assume that it is present throughout the time slice; i.e. $Gas = true$ for the duration of the time slice. On the other hand, within a time slice we obtain several sensor reports at different points in time. These reports may vary throughout the time slice. If we assume that the sequence of reports results from a first order Markov process, we can equivalently describe such a process through a set of branches rooted in a single node, provided that the hidden phenomenon is *quasi-static* (see [6]). It follows that for each observation from a sequence obtained within a single time slice we can append a new leaf node.

In addition, we assume that context variables remain constant during a time slice, i.e., they are quasi-static.

2.3. Fusion Based on Inference in Probabilistic Graphical Models

Bayesian networks support efficient fusion of very heterogeneous information based on rigorous and efficient algorithms for the computation of probability distributions $P(H|\mathcal{E})$ (or equivalently $P(H, \mathcal{E})$) over the hypothesis variables given a set of observations $\mathcal{E}$. For example, we might be interested in distribution $P(CondC|\mathcal{E})$, where $\mathcal{E}$ denotes a set of observed states of the variables corresponding to the leaf nodes in Figure 1; i.e. $\mathcal{E}$ is a configuration of the variables representing reports from humans and sensors, such as

for example $\mathcal{E} = \{E_1^{C_1} = true, E_2^{C_1} = false, \ldots, E_1^{Sympt} = true\}$. BN shown in Figure 1 supports efficient inference based on the evidence $\mathcal{E}$, i.e. computation $P(CondC|\mathcal{E})$.

In principle, the computation of $P(H, \mathcal{E})$ requires marginalization of all variables in the BN except H and the evidence variables that were instantiated according to $\mathcal{E}$. If we can express a joint probability distribution over $\mathcal{V}$ as a product of factors corresponding to conditional and prior probabilities, then the computation can be made efficient by using sequences of factor multiplications and summations over the resulting products. Most types of graphical models that represent a factorization of some global function can be translated to a factor graph (FG) [13]. If $\mathcal{V}$ is the complete set of variables and ϕ is the global function over $\mathcal{V}$, then we have a factorization

$$\phi(\mathcal{V}) = \prod_{i}^{n} \phi_i(\mathcal{V}_i),$$

where $\mathcal{V}_i$ are subsets of variables, and ϕ_i are the factors. In most meaningful cases, the subsets overlap.

In the case of discrete BNs, the global function is the joint probability distribution (JPD) and the factors are the CPTs. A factor graph represents a factorization by using an undirected graph, containing variable nodes and factor nodes. Each factor node ϕ_i is connected to exactly those variable nodes that correspond to variables in $\mathcal{V}_i$. Each variable node x is connected to exactly those factor nodes ϕ_i for which $x \in \mathcal{V}_i$.

It is easy to see that any Bayesian network can be translated to a factor graph. In fact, we could have built many different factor graphs representing the same JPD, since multiple factors can always be combined into 'larger' factors through multiplication. Figure 2 shows such a factor graph corresponding to Figure 1. In this graph function f_1 is defined as a product of the factors from the first two lines in (1), function f_2 is defined as a product of the factors from the third and the fourth lines in (1), function f_3 is defined as a product of the factors from the fifth line in (1). Functions f_4 and f_5 correspond to factors $\prod_{m=1}^{M} P(E_m^C|S_1^C)$, $\prod_{n=M+1}^{N} P(E_n^C|S_2^C)$, respectively. f_6 corresponds to factor $\prod_{o=1}^{O} P(E_o^I|S_3^I)$, while factors f_7 and f_8 correspond to products $\prod_{p=1}^{P} P(E_p^{Smell}|Smell)$ and $\prod_{q=1}^{Q} P(E_q^{Sympt}|Sympt)$. Furthermore, it is well known that if a factorization of a JPD $P(\mathcal{V})$ corresponds to a factor graph with tree topology (i.e. factor tree), the sum-product algorithm [13] can be used for exact computation of marginal probabilities $P(H, \mathcal{E})$. In fact, all approaches to exact belief propagation in BNs in principle implement the sum-product algorithm on tree like factor graphs [13, 3]. If a DAG of a BN is a poly-tree, then the BN already corresponds to a factor tree.

Often, however, domain models are described through multiply connected BNs, which correspond to loopy factor graphs. For example, in Figure 1 the context variables T and M introduce loops in the DAG. This means that exact sum-product propagation on the original BN topology is not possible. Consequently, the original BNs have to be mapped to representations which correspond to tree factor graphs. There basically exist two ways of transforming multiply connected BNs into models corresponding to factor trees:

- Multiply connected BNs can be transformed into secondary representations such as Junction trees, which in turn directly correspond to tree factor graphs. Basically, Junction trees collect nodes from an original BN into clusters which represent hyper nodes in a tree.
- Alternative approach is to instantiate certain variables in a multiply connected BN such that it corresponds to a factor graph with a tree topology. By instantiation of variables the corresponding nodes and the corresponding links in factor graphs are eliminated; i.e. we obtain cross-sections [13]. For example, if the variables M and T from the model in Figure 1 are instantiated, the corresponding highlighted nodes in Figure 2 and the dashed lines disappear and we obtain a factor tree. This approach is viable if we can access the relevant information sources, which is often the case in monitoring applications (see rationale in section 5.3).

In this paper we choose the instantiation-based approach. However, one of the main challenges is to determine which variables have to be instantiated. By taking into account the locality of causal relations in BNs, we derive rules for efficient instantiation supporting creation of distributed models corresponding to factor trees (see section 3.2).

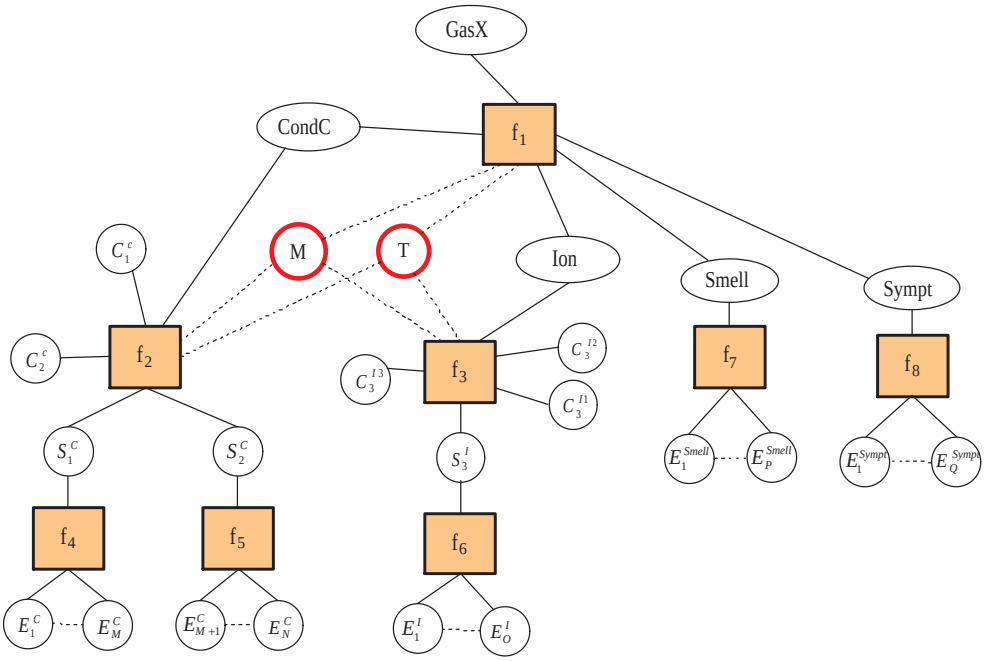


Figure 2: Factor graph where boxes represent products of original factors (CPTs). By instantiating variables M and T the corresponding nodes and the relations (dashed links) are eliminated, which transforms the graph into a factor tree.

2.4. Locality of Causal Relations

Locality of causal relations in BNs can facilitate distributed modeling and inference processes. An important concept is d-separation [25, 20], which describes how different parts of a BN can be rendered conditionally independent. Let's assume DAG $\mathcal{G} = \langle \mathcal{V}, \mathbb{E} \rangle$ and mutually disjoint sets of variables $\mathcal{V}_i \subset \mathcal{V}$, $\mathcal{V}_j \subset \mathcal{V}$ and $\mathcal{Z} \subset \mathcal{V}$. Sets $\mathcal{V}_i$ and $\mathcal{V}_j$ are d-separated by $\mathcal{Z}$, if instantiation of all variables in set $\mathcal{Z}$ makes arbitrary variable $X \in V_i$ independent of any variable $Y \in \mathcal{V}_j$. D-separation of $\mathcal{V}_i$ and $\mathcal{V}_j$ given $\mathcal{Z}$ in DAG $\mathcal{G}$ is denoted by $\langle \mathcal{V}_i | \mathcal{Z} | \mathcal{V}_j \rangle_G$. In other words, if all variables in set $\mathcal{Z}$ are instantiated and $\langle \mathcal{V}_i | \mathcal{Z} | \mathcal{V}_j \rangle_G$, then computation of probabilities over variables $\mathcal{V}_i \subseteq \mathcal{V}$ corresponding to subgraph $\mathcal{G}_i$ is independent of the instantiation and belief propagation over variables $\mathcal{V}_j \subseteq \mathcal{V}$ in another subgraph $\mathcal{G}_j$ of $\mathcal{G}$.

Moreover, d-separation is related to the central concept of our analysis, the Markov boundary:

Definition 1 (Markov boundary of a single variable). *Given a probabilistic model $P(\mathcal{V})$ over a set of variables $\mathcal{V}$ and a variable $X \in \mathcal{V}$, then a Markov boundary $B(X)$ of X is a minimal set of variables $\mathcal{Z}$ for which the following conditional independence is valid*

$$I(\{X\}, \mathcal{Z}, \mathcal{V} \backslash (\{X\} \cup \mathcal{Z}))_P. \tag{2}$$

where $I(\{X\}, \mathcal{Z}, \mathcal{V} \backslash (\{X\} \cup \mathcal{Z}))_P$ denotes that X is conditionally independent of any variable in the set $\mathcal{V} \backslash (\{X\} \cup \mathcal{Z})$ given the set $\mathcal{Z}$.

It can be shown that Markov boundary $B(X)$ of any variable X in a *faithful*[3] BN is the union of all parents of X, children of X and parents of children of X (see [20] for the proof). The concept of Markov boundaries can be extended to sets of variables. The definition of such boundaries is based on the concepts of parents and children of a set of variables [1]:

[3]Given a probabilistic model $P(\mathcal{V})$ and a DAG $\mathcal{G}$, we say that P and $\mathcal{G}$ are faithful if $\mathcal{G}$ entails all conditional independencies in P as well as all dependencies; $\mathcal{G}$ is a perfect map of P (see [20, 18]).

Definition 2 (Children and Parents of Sets of Variables). *Let $\mathcal{G}$ be a DAG over a set $\mathcal{V}$ of variables. Let $\mathcal{V}_i$ be a set of variables such that $\mathcal{V}_i \subset \mathcal{V}$ and $\mathcal{V}_i \neq \emptyset$. Any node $X \in \mathcal{V} \backslash \mathcal{V}_i$ that has a child in $\mathcal{V}_i$ is a parent of set $\mathcal{V}_i$; any node in $X \in \mathcal{V} \backslash \mathcal{V}_i$ that has a parent node in $\mathcal{V}_i$ is a child of set $\mathcal{V}_i$.*

By using parents and children of sets of variables we can define a Markov boundary of a set of variables [1]:

Definition 3 (Markov Boundary of a Set of Variables). *Markov boundary $B(\mathcal{V}_i)$ of a set of variables $\mathcal{V}_i \subset \mathcal{V}$ in a* faithful *BN is the union of all parents of set $\mathcal{V}_i$, children of $\mathcal{V}_i$ and parents of children of $\mathcal{V}_i$.*

Markov boundary of a set of variables $B(\mathcal{V}_i)$ implies that computation of beliefs in a sub model corresponding to some $\mathcal{G}_i$ with variables $\mathcal{V}_i \subset \mathcal{V}$ can be carried out in isolation, independently of inference processes in the rest of the network with variables $\mathcal{V}$, if all variables in $B(\mathcal{V}_i)$ are instantiated. This is a very important property which can be exploited for efficient distribution of models and inference processes (see section 3.2).

3. Distributed Fusion

By using basic fusion building blocks, each specialized for a limited fusion task, adequate domain models (i.e. BNs) can be assembled at runtime as the information sources are discovered. In addition, fusion in such an assembled network can easily be distributed throughout several machines thus avoiding processing and communication bottlenecks. Beside sound and efficient fusion algorithms, basic fusion modules must support also efficient communication and cooperation protocols. In addition, a distributed fusion system should be able to adapt to the current situation without intervention of humans, which requires autonomous behavior; modules should form fusion systems consisting of relevant modules autonomously and they should be able to reason about resource allocation with respect to sensing and processing capacity. In order to be able to cope with such complex functionality in a systematic way, we make use of the multi-agent systems paradigm [9, 27].

Definition 4. Fusion agent *A_i is a processing unit, a module, which can compute probability distributions over variables in its local BN.*

- *A local BN is defined through a local* **DAG** *$\mathcal{G}_i = \langle \mathcal{V}_i, \mathbb{E}_i \rangle$ and a set of conditional probabilities which encode factorization of a joint probability distribution $P(\mathcal{V}_i)$ over local variables.*
- *Each agent A_i maintains a set of* ***service variables*** *$\mathcal{R}_i \subset \mathcal{V}_i$ and a set of* ***input variables*** *$\mathcal{L}_i \subset \mathcal{V}_i$. We assume that input and output variables are either leaf or root nodes of a local DAG.*
- *Sets $\mathcal{R}_i$ and $\mathcal{L}_i$ can, in general, intersect.*
- *A_i computes marginal probability distributions over the states of any variable $X \in \mathcal{V}_i$ by using it's local probabilistic model $P(\mathcal{V}_i)$ and probability distributions over input variables $L_q \in \mathcal{L}_i$.*
- *Computation of marginal distributions (i.e. local inference) is triggered as the probability distribution over any input variable $L_q \in \mathcal{L}_i$ is changed or a local variable is observed.*
- *Each agent A_i communicates an updated probability distribution over a local variable $R_k \in \mathcal{R}_i$ to any other fusion agent $A_j, j \neq i$ whose local DAG also contains variable $R_k \in \mathcal{V}_j$. Such communication results in updated distributions over a certain input variable $L_p \in \mathcal{L}_j$, which in turn triggers computation of marginal probabilities over unobserved variables in the local BN of A_j.*
- *Each agent A_i supports standard communication and collaboration protocols [27]. This includes service discovery and establishing fusion contracts with other agents by using appropriate bidding mechanisms.*

Agents defined in this way support a data-driven approach to distributed probabilistic inference. Algorithms for local inference which support globally coherent fusion are described in sections 3.3 and 4.1.

3.1. Fusion Organization

Each fusion process depends on the constellation of cooperating agents, which corresponds to a particular problem/task decomposition. In this context we introduce the concept of a *Fusion Organization*:

Definition 5 (Fusion Organization). *A* Fusion Organization Ω *is defined by a function* $\mathcal{F}(H, \mathcal{A}) : \mathcal{A}, H \rightarrow \langle \mathcal{A}_H, \mathcal{C}_H \rangle$. *This function maps the set of existing agents* $\mathcal{A}$ *and a set of hidden variables* H *to a tuple where (i)* $\mathcal{A}_H \subseteq \mathcal{A}$ *denotes the set of agents that can provide domain models and information relevant for the reasoning about variables* H *and (ii)* $\mathcal{C}_H$ *represents a set of agent pairs which directly communicate their local beliefs about certain variables. Tuple* $\langle \mathcal{A}_H, \mathcal{C}_H \rangle$ *can be represented with the help of a graph, where the nodes represent agents from* $\mathcal{A}_H$ *and each link connects an agent pair from* $\mathcal{C}_H$.

A global task of a particular fusion organization Ω is collaborative computation of probability distribution $P(H|\mathcal{E})$ over hidden variables of interest H which correctly reflects the entire evidence set $\mathcal{E}$. This is the case if $P(H|\mathcal{E})$ is identical to $P'(H|\mathcal{E})$ which is obtained through propagation of the entire evidence $\mathcal{E}$ in a monolithic BN which correctly captures relations between the modeling variables. This requires well defined cooperation of fusion agents since evidence $\mathcal{E}$ corresponds to instantiations of variables in different agents in a fusion organization Ω.

In the following sections we investigate how this can be achieved in a system of cooperating fusion agents without any centralized configuration and coordination of local fusion processes.

3.2. Model Decomposition

The main challenge in modular information fusion is determination of local BNs, such that agent organizations support correct collaborative computation of $P(H|\mathcal{E})$ without any (i) global compilation of secondary fusion structures (e.g. Junction trees) and (ii) centralized fusion control[4]. This is possible if local BNs in different agents are designed in such a way that collaborative inference in agent organization Ω implements the sum-product algorithm in factor trees (see section 2.3). In the following discussion we provide design rules which support such processing.

Given an arbitrarily complex monolithic BN with a DAG $\mathcal{G} = \langle \mathcal{V}, \mathbb{E} \rangle$, we introduce a system of fusion agents, such that each agent A_i is reasoning about a subset of variables $\mathcal{V}_i \subset \mathcal{V}$ from $\mathcal{G}$. Local BNs with DAGs $\mathcal{G}_i = \langle \mathcal{V}_i, \mathbb{E}_i \rangle$, each corresponding to a local domain expertise of a fusion agent A_i, are obtained through partitioning of a monolithic model. We assume that DAG $\mathcal{G}$ is partitioned into n subgraphs $\mathcal{G}_1^*, \ldots, \mathcal{G}_n^*$ defined over sets of variables $\mathcal{V}_1^*, \ldots, \mathcal{V}_n^*$, respectively. Typically, the variables in a partition describe a set of phenomena which are logically related, such as states of components of a sensor, etc. The partitioning could be also based on arbitrary optimality criteria.

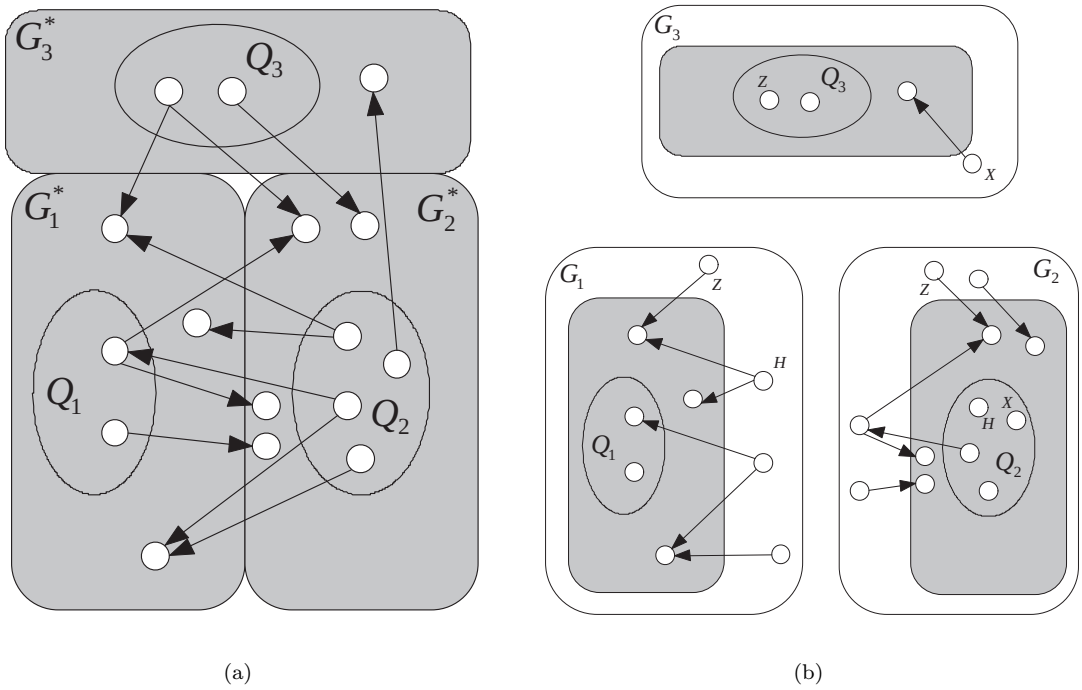
In principle, each local DAG $\mathcal{G}_i$ is derived from an original partition $\mathcal{G}_i^*$ and must support globally coherent reasoning about the local variables in $\mathcal{G}_i^*$; i.e. the estimated distributions over variables $\mathcal{V}_i^*$ correctly reflect the distributions over variables from all partitions. Thus, each local DAG $\mathcal{G}_i$ must contain all variables which directly influence variables from the original partition $\mathcal{G}_i^*$; i.e. each derived local BN must share certain variables with at least one other local BN. By taking into account the principles of d-separation, we can derive simple and effective design rules:

Definition 6 (Design Rules). *Let DAG* $\mathcal{G}$ *of a monolithic BN be defined over a set of variables* $\mathcal{V}$. *From this monolithic BN we obtain local DAGs* $\mathcal{G}_i$ *by applying the following rules:*

1. *DAG* $\mathcal{G}$ *is partitioned into* n *subgraphs* $\mathcal{G}_1^*, \ldots, \mathcal{G}_n^*$ *defined over sets of variables* $\mathcal{V}_1^*, \ldots, \mathcal{V}_n^*$, *such that* $\mathcal{V} = \bigcup_i \mathcal{V}_i^*$ *and* $\forall i \neq j : \mathcal{V}_i^* \cap \mathcal{V}_j^* = \emptyset$ *(e.g. see boxes in Figure 3.a).*
2. *A local DAG* $\mathcal{G}_i$ *is obtained by augmenting* $\mathcal{G}_i^*$ *with all parents* $\pi(\mathcal{V}_i^*)$ *of set* $\mathcal{V}_i^*$ *(e.g. see local graphs* $\mathcal{G}_1, \mathcal{G}_2$ *and* $\mathcal{G}_3$ *shown in Figure 3.b, where the nodes outside of the gray areas represent added parents). In the local DAG* $\mathcal{G}_i$ *links between the nodes* $\pi(\mathcal{V}_i^*)$ *are omitted if they exist in the original monolithic BN.*[5]

[4] i.e. central processes which determine messaging sequences between agents.

[5] The omitted links are still defined in another local DAG $\mathcal{G}_j$ where the variables of both graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ have a non-empty intersection.



(a) (b)

Figure 3: a.) Partitioning of a monolithic DAG; gray boxes represent arbitrarily complex partitions $\mathcal{G}_i^*$. The figure shows in each partition only variables which are directly related with variables from other partitions. b.) Local DAGs $\mathcal{G}_i$ obtained with the help of rules from Definition 6. In each DAG $\mathcal{G}_i$ nodes outside of the gray areas are parents added to the original partition $\mathcal{G}_i^*$.

3. *For each root node X in a local DAG $\mathcal{G}_i$, assign a uniform prior distribution if X was added to the original partition $\mathcal{G}_i^*$ (see step 2); i.e. X is one of the parents of $\mathcal{V}_i^*$: $X \in \pi(\mathcal{V}_i^*)$.*

Note that rule (*ii*) exploits one of the three possible approaches to efficient construction of adequate local DAGs (see appendix in [24]).

The introduced rules guarantee that local DAGs capture all crucial dependencies between the original partitions defined in the monolithic DAG $\mathcal{G}$. Figure 4 shows a system of local BNs which were generated from the monolithic BN from Figure 1 by using the rules from Definition 6. The original BN was partitioned into subgraphs $\mathcal{G}_1^*, \ldots, \mathcal{G}_8^*$ defined over sets:

$$\begin{aligned}
\mathcal{V}_1^* &= \{GasX, CondC, Ion, Smell, Sympt, T, M\},\\
\mathcal{V}_2^* &= \{C_1^c, C_2^c, S_1^C, S_2^C\},\\
\mathcal{V}_3^* &= \{C_3^{I1}, C_3^{I2}, C_3^{I3}, S_3^I\},\\
\mathcal{V}_4^* &= \{E_1^{smell}, \ldots, E_P^{smell}\},\\
\mathcal{V}_5^* &= \{E_1^{sympt}, \ldots, E_Q^{sympt}\},\\
\mathcal{V}_6^* &= \{E_1^C, \ldots, E_M^C\},\\
\mathcal{V}_7^* &= \{E_{M+1}^C, \ldots, E_N^C\},\\
\mathcal{V}_8^* &= \{E_1^I, \ldots, E_O^I\}.
\end{aligned} \tag{3}$$

Partitions $\mathcal{G}_1^*, \ldots, \mathcal{G}_8^*$ were used for the construction of local graphs shown in Figure 4.

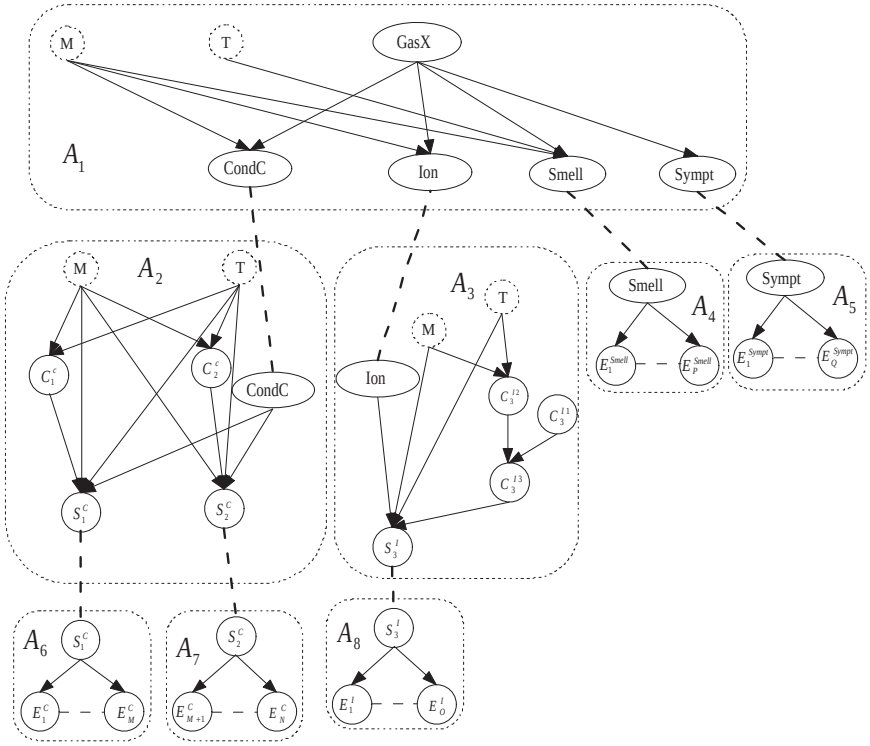


Figure 4: A distributed BN which supports inference that is equivalent to inference in the original monolithic BN shown in Figure 1. Each dashed rectangle is a local BN featuring DAG $\mathcal{G}_i$ constructed by using design and assembly rules from Definitions 6 and 8, respectively. Dashed lines show the flow of messages between the modules.

Proposition 1. *By removing from $\mathcal{G}_i$ the union of (i) all parents $\pi(\mathcal{V}_i^*)$ of the original partition $\mathcal{G}_i^*$ and (ii) nodes that have children outside of $\mathcal{G}_i^*$ we obtain a new subgraph $\mathcal{G}_i'$ with $\mathcal{V}_i'$. The set of variables in $\mathcal{G}_i^*$ which have children in other subsets of variables $\mathcal{V}_j^*, i \neq j$ is denoted by $\mathcal{Q}_i \subseteq \mathcal{V}_i^*$ (e.g. see Figure 3.a). For $\mathcal{V}_i' \neq \emptyset$, the set $\mathcal{V}_i \backslash \mathcal{V}_i'$ is $\mathcal{V}_i'$s Markov boundary $B(\mathcal{V}_i')$ if either (i) each node in $\mathcal{Q}_i$ has at least one parent in the original partition $\mathcal{G}_i^*$ or (ii) each added parent has at least one child node which is not included in $\mathcal{Q}_i$. See proof in [24].*

In other words, by using the design rules, we can construct local graphs $\mathcal{G}_i$ in which we can trivially identify a *minimal* set of variables whose instantiation renders the computation in the original partitions $\mathcal{G}_i^*$ *independent* of other variables in the original monolithic graph $\mathcal{G}$. Note that the variables included in the resulting Markov boundaries $B(\mathcal{V}_i')$ are shared between different partitions $\mathcal{G}_i$, while the variables $\mathcal{V}_i \backslash B(\mathcal{V}_i')$ are used only in $\mathcal{G}_i$. For example, for partitions $\mathcal{V}_1^*$ and $\mathcal{V}_2^*$ given in (3) we obtain $\mathcal{V}_1' = \{GasX\}$ and $\mathcal{V}_2' = \{C_1^c, C_2^c\}$. For these sets we can identify $B(\mathcal{V}_1') = \{CondC, Ion, Smell, Sympt, T, M\}$ and $B(\mathcal{V}_2') = \{CondC, M, T, S_1^C, S_2^C\}$, respectively.

If there exist uninstantiated variables in a Markov boundary $B(\mathcal{V}_i')$ the agents sharing variables in $B(\mathcal{V}_i')$ must communicate local beliefs in order to achieve globally coherent inference. In this way local inference processes become dependent. However, through instantiation of variables in the Markov boundaries we can control the dependencies and avoid inefficient inter-agent synchronization of local inference processes. In other words, a Markov boundary can be viewed as an "information valve" which supports a systematic control of the dependencies between inference processes in different local models. In addition, Markov boundaries resulting from the introduced design rules have the following important property:

Proposition 2. *If for each original partition $\mathcal{G}_i^*$ there exists $\mathcal{V}_i' \neq \emptyset$, and the set $\mathcal{V}_i \backslash \mathcal{V}_i'$ is $\mathcal{V}_i'$s Markov boundary $B(\mathcal{V}_i')$, then each node belonging to any $B(\mathcal{V}_i')$ is contained in at least one other Markov boundary $B(\mathcal{V}_j')$. In other words, a Markov boundary $B(\mathcal{V}_i')$ corresponding to any original partition $\mathcal{G}_i^*$ intersects at least one other boundary $B(\mathcal{V}_j')$ corresponding to some original partition $\mathcal{G}_j^*$. See proof in [24].*

This proposition implies that the approach based on instantiation can be efficient; given a single observation we can instantiate variables in at least two local BNs simultaneously. Under certain conditions the presented design approach results in a minimum number of instantiations needed for globally coherent distributed inference in multiply connected BNs without compiling junction trees that span multiple modules (see appendix in [24]).

Moreover, in this paper we assume that two agents can exchange their local estimates of marginal distribution over a single process variable if a valid separator exists:

Definition 7 (Separator). *Given two variable clusters $\mathcal{V}_i$ and $\mathcal{V}_j$ corresponding to agents A_i and A_j, a valid separator $S\langle \mathcal{V}_i, \mathcal{V}_j \rangle$ exists if:*

1. *Markov boundaries corresponding to the two agents intersect: $B(\mathcal{V}_i') \cap B(\mathcal{V}_j') \neq \emptyset$.*
2. *Exactly one process variable $X \in B(\mathcal{V}_i') \cap B(\mathcal{V}_j')$ is not instantiated.*
3. *The uninstantiated variable X is represented by a root node in one agent while it corresponds to a leaf node in another agent.*

Thus, the information flow between two agents is limited to a single process variable. With the help of this policy we can formulate simple assembly rules:

Definition 8 (Assembly Rules). *System $\Omega_t = \langle \mathcal{A}_H, \mathcal{C}_H \rangle$ consisting of agents $\mathcal{A}_H$ with local BNs obtained by using design rules from Definition 6 is formed as follows: a system of n cooperating fusion agents is assembled through a sequence of expansion steps. Agent $A_i \notin \mathcal{A}_H$ can join the current system Ω_t if:*

1. *in Ω_t there exists a non-empty set of agents $\mathcal{M} \subseteq \mathcal{A}_H$, such that the Markov boundary $B(\mathcal{V}_j')$ corresponding to each agent $A_j \in \mathcal{M}$ intersects the Markov boundary $B(\mathcal{V}_i')$ corresponding to agent A_i:*

$$\mathcal{M} = \{A_j | B(\mathcal{V}_i') \cap B(\mathcal{V}_j') \neq \emptyset \wedge A_j \in \mathcal{A}_H\} \neq \emptyset \tag{4}$$

2. *In the union of intersections of $B(\mathcal{V}_i')$ with all Markov boundaries $B(\mathcal{V}_j')$ corresponding to $A_j \in \mathcal{M}$:*

$$\mathcal{B}_i = \bigcup_{A_j \in \mathcal{M}} \left(B(\mathcal{V}_i') \cap B(\mathcal{V}_j') \right), \tag{5}$$

there exists exactly one variable $X \in \mathcal{B}_i$ which is not instantiated.
3. *Only one of the intersections between $B(\mathcal{V}_i')$ and other Markov boundaries corresponds to a valid separator $S\langle \mathcal{V}_i, \mathcal{V}_j \rangle$. In other words, the non-instantiated variable X is represented by a leaf node in a local BN of exactly one agent $A_j \in \mathcal{M}$.* [6]
4. *The agents A_i and A_j corresponding to the separator $S\langle \mathcal{V}_i, \mathcal{V}_j \rangle$ establish a fusion contract.*

For example, consider the agents with their local models given in Figure 4. Let's assume that agent A_2 wants to join organization Ω_t containing agents $\mathcal{A}_H = \{A_1, A_3, A_4, A_5, A_8\}$. The Markov boundary $B(V_2') = \{M, T, CondC\}$ of A_2 has a non empty intersection with the Markov boundaries $B(\mathcal{V}_1)$ and $B(\mathcal{V}_3)$ of agents $A_1 \in \mathcal{A}_H$ and $A_3 \in \mathcal{A}_H$, respectively; i.e. $B(V_2') \cap B(V_1') = \{M, T, CondC\}$ and $B(V_2') \cap B(V_3') = \{M, T\}$. Therefore, condition (i) in Definition 8 is satisfied. Conditions (ii) and (iii) are satisfied if variables M and T are instantiated. In this case the only uninstantiated variable is $CondC$, which is represented by a leaf and a root node in the local BNs of agents A_1 and A_2, respectively. A fusion contract can be established between agents A_1 and A_2 and we obtain a new fusion organization Ω_{t+1} with $\mathcal{A}_H = \{A_1, A_2, A_3, A_4, A_5, A_8\}$.

While the assembly rules might seem very restrictive at the first glance, it turns out that they are relevant for a significant class of real world problems (see the discussion in section 5.1).

Proposition 3. *If (i) design rules from Definition 6 are used for the construction of local BNs and (ii) the assembly of a fusion organization is based on the rules from Definition 8, then a system of agents implements a factor tree. See proof in [24].*

For example, by using the rules from Definitions 6 and 8 we would transform the BN in Figure 1 into a distributed system shown in Figure 4, if the original partitions were chosen according to (3). The distributed system corresponds to the factor tree depicted in Figure 2. In this figure, each box corresponds to an agent implementing a factor and nodes connecting factors represent the shared variables from the separators. Since variables M and T are instantiated, the corresponding nodes and the associated links disappear, which is represented by the dotted links and highlighted nodes in Figure 2.

Proposition 3 implies that a distributed system obtained through application of the rules from Definitions 6 and 8 supports exact belief propagation without compilation of secondary structures spanning multiple agents.

3.3. Distributed Inference

Distributed inference in a fusion organization Ω is based on sharing results of belief propagation in local BNs. Given condition (iii) from Definition 8, Ω corresponds to a tree, where agents are represented by nodes while the separators correspond to links. For example, Figure 5 depicts a simple organization, where agents are represented by ellipses and link labels denote the process variables that are pairwise shared in the separators. Moreover, we assume a data-driven approach to collaborative inference. For example, agent A_i receives from agent A_s probability distribution $P_{in}(X_j|\mathcal{E}_s^t)$, which is based on the evidence $\mathcal{E}_s^t$ in the branch connected to A_i via agent A_s[7]. $\mathcal{E}_s^t$ represents all observations of variables in the branch defined through agents A_s and A_u up to the time t. Upon receiving $P_{in}(X_j|\mathcal{E}_s^t)$ agent A_i runs algorithm 1 which computes distribution $P_{out}(X_k|\mathcal{E}^t \backslash \mathcal{E}_r^t)$ communicated to agent A_r. $\mathcal{E}^t \backslash \mathcal{E}_r^t$ denotes the evidence over variables in the entire tree except the branch connected to A_i via agent A_r. In this example, $\mathcal{E}^t \backslash \mathcal{E}_r^t$ denotes the evidence over variables from the branch consisting of agents A_i, A_s and A_u. $\mathcal{E}_r^t$, on the other hand, corresponds to the evidence over variables in the branch defined through agents A_r, A_p and A_q. Note also that evidence includes observations of process variables as well as context variables (see section 2.1).

[6] If $B(\mathcal{V}_i')$ intersects more than one Markov boundary from Ω_t, then the uninstantiated variable X must be represented by a root node in all but one local BN. With this condition we guarantee that the graph corresponding to Ω_{t+1} is a tree.

[7] By using the design rules from Definition 6 each shared root node (i.e. added parent) is assigned a uniform prior distribution. Consequently, $P_{in}(X_j|\mathcal{E}_s^t)$ is proportional to $P(\mathcal{E}_s^t|X_j)$; i.e. ratio $P_{in}(X_j|\mathcal{E}_s^t)/P(\mathcal{E}_s^t|X_j)$ is constant for all states of X_j.

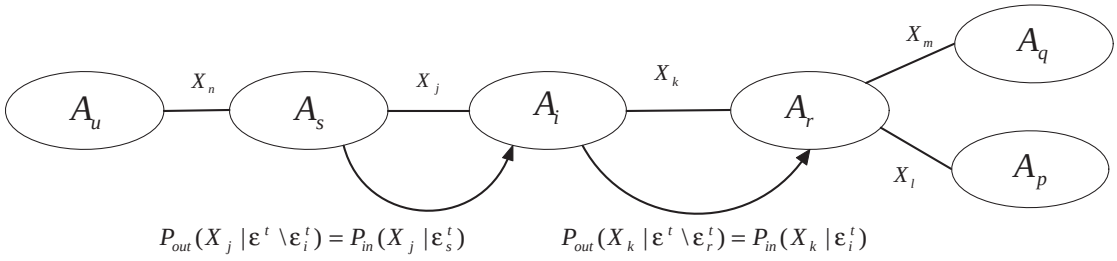


Figure 5: Types of probability distributions exchanged between cooperating fusion agents. Labels of links denote process variables from the separators shared by neighboring agents.

In general, the algorithm 1 outputs distributions $P_{out}(X_k|\mathcal{E}^t\backslash\mathcal{E}_r^t)$ for each separator except the separator shared by the agent A_s that supplied the updated distribution $P_{in}(X_k|\mathcal{E}_j^t)$; i.e. agent A_i with n separators, each corresponding to an adjacent agent in Ω, thus computes and communicates to other agents $n-1$ distributions $P_{out}(X_k|\mathcal{E}^t\backslash\mathcal{E}_r^t)$. If every agent in Ω runs algorithm 1 then every variable in any local BN is updated as a new piece of evidence is entered into a local BN of an arbitrary agent. In other words, the algorithm is executed in a sequence by all agents in Ω.

The local computation of $P_{out}(X_k|\mathcal{E}^t\backslash\mathcal{E}_r^t)$ and $P(X|\mathcal{E}^t)$ in algorithm 1 can be based on any standard approach to belief propagation in arbitrarily complex local BNs, such as Junction tree propagation [29]. Since a local BN of an agent does not change, the corresponding Junction trees can be computed prior to the operation. A system using algorithm 1 has the following important property:

Proposition 4. *A system of agents Ω implements the sum-product algorithm [13] for the computation of marginal distributions over all uninstantiated variables contained in the separators, if (i) Ω is based on the principles from the definitions 6 and 8 and (ii) each agent uses algorithm 1. See proof in [24].*

In other words, algorithm 1, combines partial results of local standard inference processes in such a way that a marginal distribution over an arbitrary subset of variables is globally coherent; the presented approach makes use of local BNs directly and, due to the induced conditional independences, the partial results from the local fusion processes can be shared without any global synchronization and compilation of secondary structures spanning several agents. From the global perspective, each evidence spawns a distribute evidence process throughout the entire Ω. At the same time, from the perspective of an individual agent A_i, the algorithm correctly implements a sequential collect evidence [10, 5]. This is achieved through the use of the input sets $\mathcal{I}_i^{X_k}$, which contain all distributions reflecting the total evidence over variables in various branches of Ω, rooted in A_i; change of any element of this set spawns recalculation of the distributions over the variables in the local model of A_i. By using algorithm 1, a process variable from a separator has both a role of an input and a service (i.e. output) variable, dependent on which nodes in Ω were instantiated.

The same algorithm can be used for the distribution of the prior knowledge; prior probability over states of root nodes in each agent should be propagated throughout the entire system of agents. Each agent A_j joining an organization Ω first receives from a new neighbor A_r distribution $P_{in}(X_k|\mathcal{E}_r)$. As next A_j runs the algorithm and communicates the resulting distribution over the separator variable X_k shared with agent A_r. This spawns local processing in agent A_r which in turn will communicate updated distributions over separators with all its neighbors except A_j. Local propagation is used to compute prior distributions over the separator variables and communicated to the neighboring agents.

4. Distributed Perception Networks

The distribution principles explained in section 3.2 are exploited in Distributed Perception Networks (DPN), multi-agent systems for efficient and robust information fusion. In this paper we introduce a simplified version of the DPN fusion architecture optimized for the estimation of hidden events based on the

Algorithm 1: Local fusion algorithm

procedure : `LocalFusion`

input : Local BN $\langle \mathcal{G}_i, P(\mathcal{V}_i)\rangle$ of agent A_i, where each root node added to the original partition $\mathcal{G}_i^*$ is assigned a uniform distribution.

input : Distribution $P_{in}(X_m|\mathcal{E}_j^t)$ over the uninstantiated variable X_j from separator $S\langle \mathcal{V}_i, \mathcal{V}_j\rangle$. $P_{in}(X_m|\mathcal{E}_j^t)$ is communicated by another agent A_j. $\mathcal{S}_i$ denotes all uninstantiated separator variables in local DAG of A_i.

input : Observations of states in the local DAG $\mathcal{G}_i$; set of observed local variables is denoted by $\mathcal{C}_i \subset \mathcal{V}_i$.

output : For each neighboring agent A_r, except the agent that supplied $P_{in}(X_m|\mathcal{E}_j^t)$, compute and communicate updated distribution $P_{out}(X_k|\mathcal{E}^t \backslash \mathcal{E}_r^t)$ for the uninstantiated separator variable X_r shared by agents A_i and A_r.

initialization: For each uninstantiated separator variable X_k create a set $\mathcal{I}_i^{X_k} = \{P_{in}(X_k|\mathcal{E}_j = \emptyset), \ldots, P_{in}(X_k|\mathcal{E}_q = \emptyset)\}$ whose entries are uniform distributions, each entry corresponding to an agent sharing variable X_k with agent A_i. Instantiate all observed variables $X_p \in \mathcal{C}_i \subseteq \mathcal{V}_i$, including all context variables; i.e. set $P(X_p)$ to the corresponding point mass distributions.

1 Replace the distribution $P_{in}(X_m|\mathcal{E}_j)$ in list $\mathcal{I}_i^{X_m}$ with $P_{in}(X_m|\mathcal{E}_j^t)$: $P_{in}(X_m|\mathcal{E}_j) \leftarrow P_{in}(X_m|\mathcal{E}_j^t)$;

2 **foreach** *neighboring agent* A_r **do**

3 (i) determine $\mathcal{I}_i^r = \bigcup_{X_m \in \mathcal{S}_i} \mathcal{I}_i^{X_m} \backslash \{P_{in}(X_k|\mathcal{E}_r)\}$, a set of all separator distributions provided by the neighbors of A_i excluding the distribution supplied by A_r and (ii) compute:

$$P_{out}(X_k|\mathcal{E}^t \backslash \mathcal{E}_r^t) = \alpha_1 \sum_{\mathcal{V}_i \backslash X_k} P(\mathcal{V}_i) \prod_{X_p \in \mathcal{C}_i} P(X_p) \prod_{P_{in}(X_n|\mathcal{E}_q) \in \mathcal{I}_i^r} P_{in}(X_n|\mathcal{E}_q), \tag{6}$$

where X_n denotes the separator variables, $\mathcal{E}_q$ denotes the evidence corresponding to the branch in Ω connected to A_i via agent A_q and α_1 is a normalizing constant;

4 For each uninstantiated separator variable X_k except X_j communicate $P_{out}(X_k|\mathcal{E}^t \backslash \mathcal{E}_r^t)$ to the agents whose local models contain X_k;

5 **end**

6 For any variable $X_k \in \mathcal{V}_i$ we can compute posterior distribution given the entire current evidence $\mathcal{E}^t$ and set $\mathcal{I}_i = \bigcup_{X_m \in \mathcal{S}_i} \mathcal{I}_i^{X_m}$[8]:

$$P(X_k|\mathcal{E}^t) = \alpha_2 \sum_{\mathcal{V}_i \backslash X_k} P(\mathcal{V}_i) \prod_{X_p \in \mathcal{C}_i} P(X_p) \prod_{P_{in}(X_m|\mathcal{E}_m) \in \mathcal{I}_i} P_{in}(X_m|\mathcal{E}_m), \tag{7}$$

where α_2 is a normalizing constant;

interpretation of large quantities of observable events. In such settings we are interested only in posterior probability over a few hypothesis variables representing hidden causes of many observations. Consequently, DPNs agents use fusion algorithms which are derived from algorithm 1, but differ from this algorithm primarily in the distribution of partial fusion results.

DPN agents implement diagnostic inference, i.e., reasoning from effects to causes by considering the retrospective support [25]. Each fusion agent computes distribution over a single output (service) variable R_i by taking into account distributions over several input variables from set $\mathcal{L}_i$, where $R_i \notin \mathcal{L}_i$. Each input variable $L_k \in \mathcal{V}_i$ is typically a leaf node while R_i is one of the roots in the local DAG $\mathcal{G}_i$, respectively. Agent A_i with a set of input variables $\mathcal{L}_i \subset \mathcal{V}_i$ and agent A_j with the service concept $R_j \in \mathcal{V}_j$ can cooperate if a valid separator exists; i.e. $R_j \in \mathcal{L}_i$. Because local Bayesian network models are interpreted causally and service node of a DPN agent A_i is always ancestor of the input nodes of A_i, it follows that also collaborative fusion in DPN systems corresponds to diagnostic reasoning: that is, reasoning from effects, or symptoms,

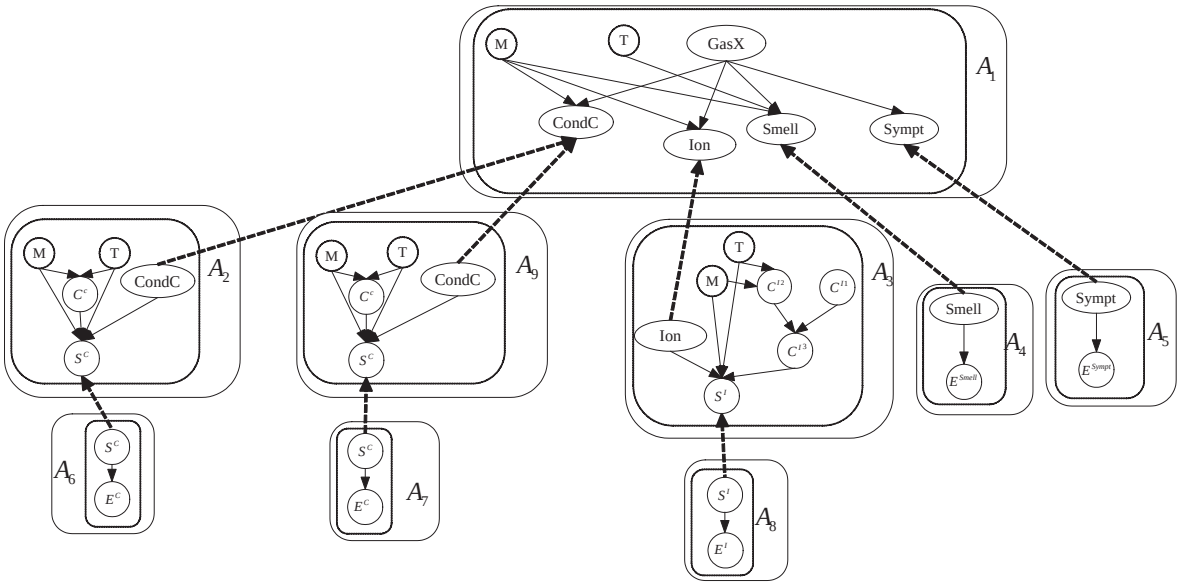


Figure 6: A DPN consisting of fusion agents implementing a distributed causal probabilistic model capturing a particular constellation of information sources. Each rectangle corresponds to an agent with a specific BN, a modeling fragment. Dashed arrows show the flow of the inter-agent messages containing partial fusion results.

to their causes.

Thus, agent A_j supplies A_i with distribution $P_{out}(R_j|\mathcal{E}^t\backslash\mathcal{E}_i^t)$ over the service variable R_j. This distribution is used as input in agent A_i; i.e. $P_{in}(L_i|\mathcal{E}_j^t) = P_{out}(R_j|\mathcal{E}^t\backslash\mathcal{E}_i^t)$. In other words, we can build hierarchical systems of collaborating agents, where each agent is supplier of a single consumer throughout the duration of the operation. Therefore we can simplify the notation in the following text by setting $P_{out}(R_j|\mathcal{E}_j^t) = P_{out}(R_j|\mathcal{E}^t\backslash\mathcal{E}_i^t)$. $\mathcal{E}_j^t$ denotes the entire evidence over variables in the branch of a fusion organization Ω which is connected to the consumer agent A_i via supplier agent A_j.

4.1. DPN Agent Types

DPN agents represent basic modeling building blocks, fusion modules which must be assembled into causal models that explicitly capture every observation obtained within a certain estimation time slice. The DPN agent design is based on the rules from Definition 6 and, at the same time, it reflects the type of targeted applications; each agent supports computation of a marginal distribution over a single service variable which is based on diagnostic reasoning. In addition, local BNs can contain nodes representing context variables. Since these variables cannot be influenced by monitoring processes, the corresponding nodes in local DAGs $\mathcal{G}_i$ are always roots. As a DPN agent with such nodes joins a DPN organization, it either instantiates such variables by using its own sensors (e.g. humidity and temperature sensors) or it obtains such information from some agent that has access to such sensors in the area of interest. We assume that observations of such variables are deterministic. Also, the agent does not start supplying any other agent in the DPN organization until all context variables are instantiated.

The DPN agent design supports adaptation to changing information source constellations. As new information sources become available within a time slice, the fusion organization is expanded. We achieve this by introducing two types of DPN agents, each suitable for different modeling aspects. Each agent type is characterized through a set of possible BN topologies and a specific belief updating algorithm.

4.1.1. Static Fusion Agents

Agents implementing *Static modeling building blocks* can reason exclusively about distributions over quasi static variables (see section 2.2), as for example the top agent in Figure 6. Expertise of a static agent A_i

can be captured by a BN featuring an arbitrarily complex DAG $\mathcal{G}_i$ that has a set of quasi-static variables $\mathcal{V}_i$. Only one root variable $R_i \in \mathcal{V}_i$ can represent the service variable of agent A_i. The leaf nodes of $\mathcal{G}_i$ correspond to the input variables $\mathcal{L}_i \subset \mathcal{V}_i$.

Static agent A_i runs algorithm 2 to compute distribution $P_{out}(R_i|\mathcal{E}_j^t)$ over its service variable R_i upon receiving distribution $P_{in}(L_m|\mathcal{E}_j^t)$ over one of its input variable $L_m \in \mathcal{L}_i$. This algorithm is identical to algorithm 1, except that the posterior distribution is computed over a single service variable, which always corresponds to a root node in a local BN. The output of a static agent is computed with the help of equation (8). Arbitrary inference algorithms, such as belief propagation in Junction Trees [29, 10], can be deployed for efficient evaluation of expression (8). Compilation of Junction trees corresponding to the local BN can be carried out prior to the operation.

Algorithm 2: Static fusion algorithm

procedure : `StaticFusion`

input : Local BN $\langle \mathcal{G}_i, P(\mathcal{V}_i)\rangle$ of agent A_i, where each root node added to the original partition $\mathcal{G}_i^*$ is assigned a uniform distribution.

input : Distribution $P_{in}(L_m|\mathcal{E}_j^t)$ over the uninstantiated input variable $L_m \in S\langle \mathcal{V}_i, \mathcal{V}_j\rangle$. $P_{in}(L_m|\mathcal{E}_j^t)$ communicated by agent A_j

input : Observations of states in the local DAG $\mathcal{G}_i$; set of observed local variables is denoted by $\mathcal{C}_i \subset \mathcal{V}_i$.

output : Compute and communicate to agent A_r updated distribution $P_{out}(R_i|\mathcal{E}_i^t)$ for the uninstantiated root variable R_i shared by A_i and A_r.

initialization: For each input variable $L_k \in \mathcal{L}_i$ create a set $\mathcal{I}_i^{L_k} = \{P_{in}(L_k|\mathcal{E}_j = \emptyset), \ldots, P_{in}(L_k|\mathcal{E}_q = \emptyset)\}$ whose entries are uniform distributions, each entry corresponding to an agent sharing variable L_k with agent A_i. Instantiate all observed variables $X_p \in \mathcal{C}_i \subseteq \mathcal{V}_i$, including all context variables; i.e. set $P(X_p)$ to the corresponding point mass distributions. Note, the observations of variables in set $\mathcal{C}_i$ are a subset of the total evidence $\mathcal{E}_i^t$ in branch connected by A_i to consumer agent A_r.

1 Replace the distribution $P_{in}(L_m|\mathcal{E}_j)$ in list $\mathcal{I}_i^{L_m}$ with $P_{in}(L_m|\mathcal{E}_j^t)$: $P_{in}(L_m|\mathcal{E}_j) \leftarrow P_{in}(L_m|\mathcal{E}_j^t)$;

2 Determine $\mathcal{I}_i^r = \bigcup_{L_k \in \mathcal{L}_i} \mathcal{I}_i^{L_k}$, a set of all separator distributions provided by neighbors of A_i excluding the receiver A_r and compute:

$$P_{out}(R_i|\mathcal{E}_i^t) = \alpha \sum_{\mathcal{V}_i \setminus R_i} P(\mathcal{V}_i) \prod_{X_p \in \mathcal{C}_i} P(X_p) \prod_{P_{in}(X_n|\mathcal{E}_q) \in \mathcal{I}_i^r} P_{in}(X_n|\mathcal{E}_q), \tag{8}$$

where α is a normalization constant;

3 Supply agent A_r with $P_{out}(R_i|\mathcal{E}_i^t)$;

4.1.2. Dynamic Fusion Agents

Agents implementing *dynamic modeling building blocks* can describe sequences of observations, originating from a single information source represented by a sensor state node (see agents at the bottom in Figure 6). We assume that reports from a single sensor are caused by a combination of some hidden quasi-static phenomena captured by the sensor's state variable. Therefore, sequences of resulting observations can be captured by a naive BN, consisting of a root node representing the sensor states and leaf nodes, each capturing a sensor report. However, reasoning directly with a naive BN is not practical, since each new observation requires a modification of the local DAG. Instead, we can obtain identical results by using algorithm 3 that makes use of a BN consisting of a single service node R_i and a single leaf node E_i related through a CPT $P(E_i|R_i)$ (see agents A_4 through A_8 in Figure 6). This approach relies on two assumptions: (i) that all observations are conditionally independent given the sensor states and (ii) the generative model is the same

for all observations[9]. In other words, dynamic agents implement *plates* [3].

Algorithm 3: Dynamic fusion algorithm

procedure : `DynamicFusion`
input : Local BN $\langle \mathcal{G}_i, P(\mathcal{V}_i)\rangle$ of agent A_i, where each root node added to the original partition $\mathcal{G}_i^*$ is assigned a uniform distribution.
input : Point-mass distribution $P(E_i)$ over observation variable E_i, which corresponds to $\mathcal{E}_i^t$, an observation of E_i states at time t.
output : Communicate distribution $P_{out}(R_i|\mathcal{E}_i^t)$ over the service variable R_i.i.e. R_i is included in the separator $S\langle \mathcal{V}_i, \mathcal{V}_r\rangle$ formed by agent A_i and its consumer A_r.
initialization: Set $P(R_i)$ to a uniform distribution.

1

$$P_{out}(R_i|\mathcal{E}_i^t) = \alpha \cdot P(R_i) \sum_{E_i} P(E_i|R_i)P(E_i), \tag{9}$$

where α is a normalization constant and $\mathcal{E}_i^t$ denotes all measurements $\mathcal{E}_i^t$ obtained until time t. ;

2 Store $P_{out}(R_i|\mathcal{E}_i^t)$ and use it as a new prior in the next iteration: $P(R) \leftarrow P_{out}(R_i|\mathcal{E}_i^t)$;

3 Return $P_{out}(R_i|\mathcal{E}_i^t)$;

4.2. DPN Organization

Generation of a particular DPN fusion organization Ω is governed by the assembly rules from Definition 8. This process is based on the discovery of separators which depend on local BNs. In other words, a DPN reflects local agent ontologies encoded through local DAGs and it determines which types of information can be processed and in what order this can be done. Because of the design constraints from Definitions 6 and 8, a DPN organization Ω can be described by a graph with a tree topology which, in turn, corresponds to a factor tree. In addition, a single DPN agent can participate in several DPN organizations simultaneously, each specialized for a different fusion task[10].

By combining static and dynamic DPN agents we can assemble DPN organizations that implement complex and flexible distributed fusion systems. Each dynamic agent filters sequences of observations from a particular sensor, while static agents support reasoning with arbitrarily complex BNs. In addition, due to the properties of Algorithm 2, static agents support expansion of causal models during a time slice as new information sources and fusion agents become available.

This can be illustrated with the help of an example. Consider agent A_2 in Figure 4 which supplies its consumer A_1 with $P(CondC|\mathcal{E}_1, \mathcal{E}_2, m, t)$ based on the fusion of observations from two sensors corresponding to nodes S_1^C and S_2^C, respectively. In this example, all sensors are captured in a single local BN by nodes S_1^C and S_2^C. If the second sensor corresponding to S_2^C were added at runtime, then A_2's local model would have to be expanded by the corresponding nodes and links. In principle, for each new sensor a new local model would be required which is not practical. Instead, equivalent fusion can be achieved by distributing the fusion over a set of static agents using simpler local BNs. This becomes obvious by investigating inference in A_2. Internal belief propagation in A_2 requires marginalization of C_1^c, S_1^C, C_2^c, and S_2^C while the context variables M and T are instantiated. Since the causal processes corresponding to the two sensors are independent given M and T and $CondC$, the local computation in A_2 is equivalent to a multiplication of two independently obtained sums, each corresponding to a sensor:

[9] i.e. all readings of a certain type are obtained through random sampling from the same model with two nodes.

[10] Since each agent A_i supports only diagnostic reasoning, multiple consumers of A_i do not exchange any information; i.e. explaining away can take place only between variables in a single BN but not between the variables from local models of different agents.

$$P(CondC|\mathcal{E}_1,\mathcal{E}_2,m,t) = \alpha \sum_{C_1^c,S_1^C} P(S_1^C|CondC,t,m,C_1^c)P(C_1^c|m,t)P(\mathcal{E}_1|S_1^C) \cdot \sum_{C_2^c,S_2^C} P(S_2^C|CondC,t,m,C_2^c)P(C_2^c|m,t)P(\mathcal{E}_2|S_2^C), \quad (10)$$

where α is a normalization constant. Note that $P(CondC)$ does not appear in this expression since according to design rules it is a uniform distribution and, consequently, it does not influence computation of $P(CondC|\mathcal{E}_1,\mathcal{E}_2,m,t)$. Moreover, the sums in (10) could be computed by separate agents, each using a simpler local BN than A_2 in Figure 4. For example agents A_2 and A_9 in Figure 6 supply A_1 with $P(CondC|\mathcal{E}_1,m,t)$ and $P(CondC|\mathcal{E}_2,m,t)$ which are equivalent to the first and the second summation terms in (10), respectively. Namely, due to uniform priors in local models of supplying agents the following relations hold:

$$P(CondC|\mathcal{E}_i,m,t) = k_i \cdot \sum_{C_i^c,S_i^C} P(S_i^C|CondC,t,m,C_i^c)P(C_i^c|m,t)P(\mathcal{E}_i|S_i^C),$$

where k_i denotes a proportionality constant. If the sums in (10) are replaced by distributions $P(CondC|\mathcal{E}_1,m,t)$ and $P(CondC|\mathcal{E}_2,m,t)$, respectively, then $P(CondC|\mathcal{E}_1,\mathcal{E}_2,m,t)$ does not change. Namely, constant k_i is canceled out through the normalization that takes place in agent A_1. $P(CondC|\mathcal{E}_1,m,t)$ and $P(CondC|\mathcal{E}_2,m,t)$ are outputs of equation (8) executed by A_2 and A_9, respectively. Agent A_1, on the other hand, maintains distinctive separators with A_2 and A_9 and multiplies the supplied distributions $P(CondC|\mathcal{E}_1,m,t)$ and $P(CondC|\mathcal{E}_2,m,t)$ by using (8). Obviously, agents A_2 and A_9 in combination with agent A_1 in Figure 6 support marginalization which is identical to the marginalization supported by agents A_1 and A_2 in Figure 4. In this way we can build complex distributed models efficiently, without maintaining a centralized repository of all sensor models. For example, as a helicopter carrying a sophisticated sensor suite enters the monitored area, its on-board static agent contains all the knowledge of the local sensor which maps probability distribution over helicopter's sensor states S_j to a distribution over some measured phenomenon.

Belief propagation in a DPN organization Ω can be viewed as a combination of local fusion processes that are implemented by the two DPN agent types and run simultaneously on different networked devices. In principle, in a DPN organization the agents collaboratively compute distribution over a single hypothesis variable H by using evidence $\mathcal{E}$. Variable H is included in the agent which can be viewed as the root of Ω. All non-leaf nodes in Ω correspond to static DPN agents, while leaves of Ω typically correspond to dynamic fusion agents. Each piece of evidence from the set $\mathcal{E}$ is used for the instantiation of either a context variable or an input variable in one of the dynamic fusion agents; i.e. different observations from $\mathcal{E}$ correspond to instantiations in agents throughout Ω.

Proposition 5. *Agents in a DPN organization collaboratively compute posterior distribution $P(H|\mathcal{E})$ over a hypothesis variable which correctly reflects the entire evidence set $\mathcal{E}$. See proof in [24].*

In other words, a combination of fusion processes in cooperating DPN agents yields $P(H|\mathcal{E})$, which is identical to the distribution obtained through exact belief propagation in the monolithic BN that correctly captures the causal process. Note that belief propagation in DPNs is analogous to *collect evidence* procedure [10, 29, 5].

4.3. Self-organization Principles

A DPN *fusion organization* describes how predefined modeling building blocks are combined into a meaningful distributed world model that correctly maps heterogeneous evidence to a distribution over some hypothesis variable H. Since a causal model must capture every piece of evidence by a separate node and the information sources are not known prior to the operation, the DPN agents must be able to autonomously

form DPN organizations which implement adequate distributed causal models and support correct inference. By considering local causal models distributed throughout different agents, we can derive algorithms which support self-organization based on the cooperation between pairs of agents without any centralized configuration control. The basic self-organization mechanism is implemented through the algorithm for a Top-Down Network Configuration (see algorithm 4). Essentially, this algorithm implements a simple configuration process derived from the well known contract net protocol [26]: *After some agent A has become a member of a particular DPN, it starts looking for other agents whose output variables correspond to its input variable.*

Algorithm 4: Top Down Network Configuration

procedure: `TopDownConfiguration(`L`)`
input : L is a input variable of a caller agent A_k
Find a set of agents $\mathcal{A}$ such that $\forall A_i \in \mathcal{A} : R_i = L$;
foreach *agent* $A_i \in \mathcal{A}$ **do**
  if *the local DAG of agent* A_i *satisfies all rules from definition 8* **then**
    Establish a fusion contract between the caller agent A_k and A_i;
    foreach *input variable* $L_j \in \mathcal{L}_i$ *from agent* A_i **do**
      A_i calls: `TopDownConfiguration(`L_j`)`;
    end
  end
end

Algorithm 4 is suitable for the situations where we can provide a query variable H to a set of agents which organize meaningful DPN fusion systems that integrate all relevant agents and information sources for a given fusion task. For example, the DPN depicted in Figure 6 could be obtained by executing algorithm 4 for each leaf node in the local DAG of agent A_1.

In addition, algorithm 4 is a basic building block of more advanced configuration mechanisms implemented by algorithm 5, which support organization of fusion systems in response to critical observations. For a given set of DPN agents, such mechanisms support discovery of all possible DPN organizations that would make use of certain types of observations. In other words, DPNs can serve as alarm systems. This protocol is captured by the algorithm for a Bottom-Up network configuration (see Algorithm 5).

Algorithm 5: Bottom-Up Network Configuration

procedure: `BottomUpConfiguration(`R_k`)`
input : R_k is the service variable of a caller agent A_k
Find a set of agents $\mathcal{A}$ such that $\forall A_i \in \mathcal{A} : R_k \in \mathcal{L}_i$;
foreach *agent* $A_i \in \mathcal{A}$ **do**
  if *the local DAG of agent* A_i *satisfies all rules from definition 8* **then**
    Establish a fusion contract between the caller agent A_k and A_i;
    foreach *input variable* $L_j \in \mathcal{L}_i \wedge L_j \neq R_k$ *from agent* A_i **do**
      A_i calls: `TopDownConfiguration(`L_j`)`;
    end
    A_i calls: `BottomUpConfiguration(`R_i`)`;
  end
end

For example, the DPN depicted in Figure 6 could be obtained by first executing algorithm 5 by agent A_4. This would result in establishing a contract with agent A_1 which in turn would spawn a top-down configuration process in agent A_1. In this way a sequence of additional contracts would be established between A_1 and agents A_2, A_9, A_3 and A_5. This in turn would spawn top down configuration processes

resulting in contracts between the agent pairs $\{A_2, A_6\}$, $\{A_9, A_7\}$ and $\{A_3, A_8\}$ eventually resulting in the DPN in Figure 6.

By using 5, we can implement robust alarming systems. A single sensor observation can activate many DPN fusion organizations simultaneously. However, each of these organizations tries to find as many other relevant information sources as possible. In this way a false alarm by one sensor can effectively be overridden through the reports from other information sources, if a sufficient portion of information sources work properly. The decision maker is then presented with different hypotheses corresponding to critical events if their probabilities exceed certain predefined thresholds.

Algorithms 4 and 5 implement model-driven configuration processes, which exploit the fact that causal links, which should be established between the local world models, correspond to the intersections of local ontologies. Self configuration processes support a great flexibility through adaptive combination of standard building blocks at runtime. The configuration processes are decentralized, since agents initiate cooperation pairwise, by using their local ontologies encoded by their local world models. In addition, these algorithms enforce a partial order of possible link creations. Thus, by using algorithm 4 we can build hierarchical fusion systems through a hierarchical service discovery. Given the constraints for local BNs in DPN agents, we can easily show that configuration algorithm 4 automatically constructs DPN fusion organizations that correspond to factor trees over all involved variables.

4.4. Model-driven Information Acquisition

Explicit causal models used by DPNs support also systematic acquisition of large amounts of information that is relevant for a particular fusion task. This is especially useful for the fusion of information obtained from humans [16]. Causal models explicitly encode the relevant concepts through random variables which, in turn, can be mapped to meaningful queries. Such mapping is also used in DPN agents which support active querying of humans. Upon integration into a DPN structure, the DPN agents can generate relevant queries from the concepts encoded in their local BNs. Concise queries can be answered by a simple yes or no, which is used as evidence about different symptoms in DPN agents that generated the queries. The communication between the DPN agents and humans can be based on SMS messages, appropriate Web-interfaces or email. In addition, human information sources have limited capacities; a person can reply to a limited number of SMS queries. Therefore, a DPN system should first generate queries that will have the greatest impact on its belief about the hidden events of interest. Fortunately, rigorous causal models support theoretically sound and efficient approach to the ranking of evidence types with respect to the relevance [21]. The same method can be used for the selection and allocation of scarce/expensive sensors, such as a UAV with a certain type of detectors, etc.

5. Real-world Modeling Issues

Causal models used in the presented fusion approach obviously have a critical impact on the fusion quality. Domain models must describe the processes of interest sufficiently well; i.e., they must be grounded in the domain. In the case of causal BNs this requires adequate (i) causal graphs and (ii) parameters captured by the CPTs. In the following text different aspects of the grounding problem are discussed and we show that by considering the locality of causal relations grounding challenges can efficiently be tackled.

5.1. Domain Complexity

Whether modular approaches support adequate solutions depends primarily on the domain. It turns out that in some real world domains the presented approach to distributed inference based on instantiation of variables can support very efficient solutions. For example, this is the case in monitoring applications, where the detection is based on interpretation of observations from different types of passive information sources [16]. Usually each type of sensors provides observations about a single phenomenon represented by a process variable. In addition, usually components of one sensor do not influence components of another sensor. Similarly, reports from one sensor do not influence reports from another sensor. In other words, we can represent a process generating observations of a certain type with a network fragment which is sparsely

connected with other fragments. *Each fragment is typically connected to the rest of the BN via a set of context variables and a single process variable representing the phenomenon the sensor of a certain type is reporting about* (see for example figure 1). By instantiating nodes M and T in all agents whose local BNs contain these two variables, we eliminate two variables from the inference process, thus removing multiple loops in the corresponding factor graph.

Another relevant question is, whether the assumption about deterministic observations of context variables is realistic. Often these variables represent phenomena which can easily be measured with cheap and reliable sensors (e.g. temperature, humidity, wind speed, etc.). In addition, context variables serve as "global" variables used by many agents. Consequently, we might introduce a sensor of very high quality, whose measurements are communicated to several fusion agents with the corresponding context variable; i.e., in such cases investment in expensive equipment pays off.

On the other hand, domains can be so complex that modular approaches do not support efficient solutions. Namely, the DPN and other distributed approaches, such as multi-agent MSBNs [29], can be very inefficient if the corresponding domain models are densely connected. In such cases MSBNs correspond to Junction trees with large separators and hyper-nodes with many states. In the DPN approach, on the other hand, Markov boundaries tend to be large. This means that many variables in such boundaries must be observed which might not be economical either.

5.2. Robustness of Fusion Systems

Often real-world domains are so complex that not all relevant phenomena can be observed efficiently; i.e., we are confronted with the acquisition intractability [29]. Due to the lack of expertise or data some dependencies in models might be ignored. Similarly, because of the lack of data the strength of modeled relations might not correctly capture conditional probabilities over events taking place in the nature; i.e., it is very difficult to precisely determine the true probability distributions over related variables. In other words, the models are likely to be erroneous to a certain degree.

It turns out that BNs can, under certain conditions, support very robust detection despite such modeling errors. Namely, it has been shown that simple models assuming conditional independence can support good classification performance even if certain dependencies are ignored [30, 7]. In other words, occasional faults in causal structures represented by distributed fusion systems might not have a critical impact on the fusion quality.

In addition, by considering the theory of BNs, it has been shown that for a significant class of networks we can achieve very accurate detection despite imprecise parameters. The recently introduced theory of Inference Meta models (IM) is based on very coarse assumptions and exposes properties of BNs that are relevant for the construction of inherently robust fusion systems [23]. With the help of IM, we can show that the expected detection (i.e., classification) performance improves with the growing number of fragments that are conditionally independent given the variable representing the hypotheses[11], even if the modeling parameters (i.e., CPTs) deviate from the true distributions significantly. This is the case if the modeling parameters in each conditionally independent fragment correctly capture simple greater-than/smaller-than relations between the probabilities in the true distributions[12]. Often it is plausible to assume that such relations can easily be identified by experts or extracted from relatively small data sets with the help of machine learning techniques. This is especially relevant for real world applications where it is often very difficult or even impossible to obtain precise domain models. For example, it might be very difficult to obtain modeling parameters that precisely describe the true probability of obtaining certain types of reports from humans (e.g. $P(E_m^{Smell}|GasX)$). Similarly, it is very difficult to find parameters which precisely describe distributions over the combinations of the sensor states and states of the related phenomena, such as for example $P(S_1^C|CondC)$. However, if many sensors of different types are available, we can obtain BNs with

[11] A special case are causal models featuring tree topologies with great branching factors with respect to the hypothesis variable

[12] For example, assume the true conditional probabilities over binary variables E and C: $P(e|c) = 0.7$, $P(\overline{e}|c) = 0.3$, $P(e|\overline{c}) = 0.4$ and $P(\overline{e}|\overline{c}) = 0.6$. We say that a conditional probability table correctly captures relations between these probabilities if its parameters (i.e. conditional probabilities $\hat{P}(E|C)$) satisfy very simple relations: $\hat{P}(e|c) > 0.5$ and $\hat{P}(e|\overline{c}) < 0.5$.

great numbers of conditionally independent network fragments, which makes fusion very reliable even if we use CPT parameters that deviate from the true distributions significantly. Also, the IM theory provides a guidance for the integration of sensors, such that the fusion robustness is improved; we add sensors, whose corresponding causal models will increase the number of network fragments rooted in the hypothesis variable. In addition, IM supports techniques for runtime analysis of fusion processes, such as detection of potentially misleading fusion results and identification of modeling components that do not support accurate estimation [22].

5.3. Collaborative Design of Complex Domain Models

As we already explained in previous sections, a possible way of designing local DPN models could be decomposition of a monolithic BN. However, in complex domains construction of complex monolithic models can require contributions from many different experts and machine learning processes, which can result in inefficient development of fusion systems. Thus, an interesting question is whether it is possible to create a complex distributed model which correctly captures dependencies of the modeled process by using simple building blocks, without knowing the corresponding true monolithic BN; i.e., can we generate useful local BNs which were not obtained through decomposition. It is plausible to assume that each designer can specify a model which correctly captures probabilistic relations over a limited set of variables $\mathcal{V}_i$. However, such a local model must be consistent with other local models in a distributed fusion system. According to the design rules from Definition 6, a local model can contain process and context variables which can have direct influence on process variables in other local models. Such variables must be represented by consistently defined nodes in multiple models.

Construction of consistent local models requires synchronization of label names and clear semantics of nodes in different models. In other words, the use of sound ontologies is indispensable. A recently introduced ontology PR-OWL [11] supporting description of uncertain concepts seems to be a promising tool for the synchronization of labels and description of probabilistic concepts in a collaborative design process. Moreover, in the targeted domains, such synchronization is likely to be tractable. Namely, in systems involving heterogeneous passive sensors, each sensor is likely to measure a single phenomenon influenced by the critical hidden event, such as conductivity caused by a gas. Such a phenomenon corresponds to a single uninstantiated process variable. In addition, it is likely that there exist few context variables and it is plausible to assume that direct dependencies between phenomena captured by different local models are sparse; components and reports of a passive sensor do not influence components and reports from other sensors. In other words, each local model of a sensor shares with other local models a small set of variables. Thus, through minimal coordination between the designers we could obtain a set of local models which would correctly capture relations between all variables in a fusion organization composed according to the assembly rules from Definition 8.

Another question is whether we can easily create a faulty fusion system Ω, which corresponds to a monolithic graph with directed cycles. This might happen through unfortunate choice of link directions in separate local models or erroneous variable names used in the service discovery processes. Given the assembly rules in Definition 8, any Ω must correspond to a tree. Thus, the separator variables cannot form unintended loops or directed cycles. On the other hand, the context variables cannot form cycles either. Namely, they are not influenced by the monitoring processes and, consequently, they must be represented by roots in the monolithic BN as well as local BNs. But these nodes can introduce cycles neither in local BNs nor in assembled BNs since all links are pointing from these variables (see [28]). If more general relations in Markov boundaries were allowed, however, sophisticated cycle verification techniques must be used as for example proposed in [28].

Moreover, fusion systems consisting of DPN agents facilitate maintenance. If the expertise about a certain sub-domain changes, only the partial BNs implementing that expertise must be replaced without any further actions such as compilation of Junction trees, etc. In other words, DPNs support hot plugging and hot swapping.

6. Conclusions

The introduced design principles support construction of modular Bayesian fusion systems which can efficiently process large amounts of very heterogeneous and noisy information in a theoretically sound way. Such systems are assembled from basic fusion modules, each contributing a limited fusion functionality. Local fusion capabilities of each module are based on a specific domain model captured by a causal BN. At runtime fusion modules establish distributed domain models which correctly capture probabilistic causal relations between hidden phenomena of interest and reports from very different information sources, such as sensors, humans, etc. Systems of collaborating fusion modules implement globally coherent inference processes which are equivalent to exact belief propagation in monolithic causal BNs that correctly capture the underlying causal processes. This is achieved without any centralized control of local fusion processes. Moreover, since the presented modular approach to fusion supports hot-swap of fusion modules, the systems can efficiently cope with changing constellations of information sources. This is an important feature in many applications where the information sources are often not known prior to the operation; e.g. advanced sensors can be delivered via mobile platforms at runtime. In addition, since the system consists of weakly coupled modules, the processing load can efficiently be distributed throughout a system of networked devices.

Distributed fusion systems with aforementioned properties can efficiently be constructed by applying simple design and assembly rules derived with the help of the theory of Bayesian networks. By constructing local BNs according to the proposed design rules all causal influences between local models of different modules are preserved. With the help of the assembly rules, on the other hand, we obtain weakly coupled modular systems which support globally coherent exact inference without compilation of secondary inference structures spanning multiple fusion modules. This is achieved through a systematic instantiation of variables which reduces direct dependencies between variables in different fusion modules. Both, the design and assembly rules exploit the locality of probabilistic relations captured by causal Bayesian networks. In particular, Markov boundaries provide a systematic guidance for efficient reduction of dependencies through instantiation.

The presented approach is in principle applicable to arbitrarily complex BNs. However, it is efficient if the intersections of Markov boundaries of different local BNs are relatively small. Fortunately, this is often the case with the causal models of monitoring processes. Such BNs typically consist of several network fragments which are conditionally independent given small sets of variables.

The design principles introduced in this paper are central to Distributed Perception Networks (DPN), a multi-agent approach to distributed information fusion. DPN agents can efficiently interpret large amounts of heterogeneous information through cooperation. In principle, DPN systems are complex, self configurable classifiers consisting of DPN agents. DPN agents with limited domain expertise can autonomously form organizations that implement complex distributed fusion systems. The decentralized self configuration mechanism is based on service provider-consumer relations and exploits the locality of causal influences. In this way DPN systems can cope with settings where components can enter or leave a DPN organization at runtime. Thus, DPN systems can be viewed as a subclass of Networked Adaptive Hybrid Interactive Systems described in [12]. Agents wrap information sources and provide uniform communication and fusion protocols. As new information sources "wrapped" by DPN agents enter the scene, the overall domain model of a fusion system is adapted on the fly, without any centralized control. Such agents supply local BNs, arbitrarily sophisticated sensor models, which are plugged into the overall system as the agents join the fusion organization. In other words, each agent contributes a local model which relates sensor's observations with the rest of the distributed causal model.

The presented approach to distributed fusion is complementary to well known approaches to inference with distributed graphical models [29, 17]. In contrast to our method, however, these approaches cannot efficiently cope with domains where information source constellations are not known prior to the operation and can change at runtime. The multi-agent Multiply Sectioned Bayesian Networks (MSBN) approach [29] is a sophisticated computational framework where agents collaboratively compile Linked Junction Forests, secondary probabilistic structures spanning several agents. This approach requires adaptation of local Junction trees in agents, such that the globally coherent inference can be achieved. This, however, requires recursive processes, such as collaborative moralization, triangulation throughout entire agent systems which

involve expensive processing and message passing. This can be impractical if constellations of information sources change rapidly. Namely, addition of new inference modules corresponding to new information sources might require new compilation cycles within time intervals which are too short to obtain viable global inference structures. In addition, it turns out that causal stochastic models of monitoring processes are often connected in such a way that globally coherent inference can be achieved by instantiating a few easily observable variables while the compilation of Junction forests or distributed Junction trees introduces unnecessary computational complexity. The multi-agent MSBN principles have been used also as a basis of the approach in [1], which makes use of Markov boundaries to reduce the complexity of inference processes in MSBNs. In this approach agents collaborate to find observable Markov boundaries. However, approach in [1] does not use Markov boundaries in the design phase and it requires compilation of secondary structures spanning several agents. In our approach, on the other hand, Markov boundaries are used primarily to avoid compilation of global secondary structures. Another well known approach to distributed inference in Bayesian networks is [17], which introduces runtime compilation of Junction trees describing relations between spatially distributed sensors. The Junction tree is adapted at runtime to the quality of communication channels between the sensor nodes. However, this approach requires Markov networks capturing correlations between the neighboring sensors in particular settings. In other words, all information sources and specific correlations between them must be known prior to the operation which is clearly a serious drawback in domains where sensors are added to the system at runtime. Also Multi Entity Bayesian Networks (MEBN) approach is relevant for adaptive fusion systems, since it supports construction of complex probabilistic models by using BN fragments [14]. However, in contrast to the approaches presented in this paper and in [17, 29], in the MEBN framework monolithic BNs are created out of simpler fragments, standard Junction trees are compiled and centralized inference is executed. Beside exact belief propagation, approximate inference methods which avoid compilation of secondary inference structures seem to be relevant for modular fusion systems. Particularly interesting is loopy belief propagation [19] which, however, does not guarantee a good convergence and requires massive message passing.

The introduced theory has been implemented in a software package, which supports construction of complex agent-based fusion systems. An important feature of this package is that the fusion functionality is model driven and does not require any programming; all agents use identical software components while the fusion functionality is determined through construction of local BNs, which in turn can be designed with the help of arbitrary editors for BNs, such as HUGIN [2] or NETICA [4]. Local BNs are fed to DPN agents which analyze the BNs and automatically determine the service and input variables, i.e. variables for which the agents compute or request probability distributions, respectively.

Currently the DPN framework is being adapted to early warning systems for toxic/annoying gases in the port of Rotterdam. This work is carried out in cooperation with DCMR Milieudienst Rijnmond, an environmental management agency in Rotterdam. The future research will focus on the use of DPNs in dynamic models. An interesting approach seems to be integration of DPNs with discrete Bayesian filters [8]. In such settings DPNs can be used as complex sensor models which interpret complex patterns of observations by using dynamic process models.

Acknowledgements

The presented work was partly supported by the ICIS project funded by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024. The ICIS project is hosted by the D-CIS Lab (http://www.decis.nl), the open research partnership of Thales Nederland, the Delft University of Technology, the University of Amsterdam and the Netherlands Foundations of Applied Scientific Research (TNO).

References

[1] X. An, Y. Xiang, and N. Cercone. Revising markov boundary for multiagent probabilistic inference. In *Procs. IEEE/WIC/ACM Inter. Conf. on Intelligent Agent Technology*, pages 113–119, 2004.

[2] Hugin Expert A/S. Hugin. http://www.hugin.com/.

[3] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[4] Norsys Software Corp. Netica. http://www.norsys.com/netica.html.

[5] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, Berlin-Heidelberg-New York, 1999.
[6] 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.
[7] P. Domingos and M. J. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *International Conference on Machine Learning*, pages 105–112, 1996.
[8] D. Fox, J. Hightower, H. Kauz, L. Liao, and D. Patterson. Bayesian techniques for location estimation. In *Proceedings of the Workshop on Location-aware Computing, part of UBICOMP Conf.*, Seattle, WA, October 2003.
[9] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. In *Autonomous Agents and Multi-Agent Systems*, volume 1, pages 7 – 38. July 1998.
[10] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.
[11] Paulo C. G. Costa K. Blackmond Laskey and Terry Janssen. robabilistic ontologies for knowledge fusion. In *Proceedings of the 11th IEEE/ISIF International Conference on Information Fusion*, Cologne, Germany, July 2008.
[12] L. Kester. Model for networked adaptive interactive hybrid system. In *in Proc. of COGIS'06*, 2006.
[13] Kschischang, Frey, and Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47, 2001.
[14] Kathryn B. Laskey and Suzanne M. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, Rhode Island, USA, 1997.
[15] James Llinas, Christopher Bowman, Galina Rogova, Alan Steinberg, Ed Waltz, and Frank White. Revisiting the jdl data fusion model ii. In *Int. Conf. on Info. Fusion*, Stockholm, Sweden, 2004.
[16] M.G. Maris and G. Pavlin. Distributed perception networks for crisis management. In *Int. Conference on Information Systems for Crisis Response*, Newark, N.J., 2006.
[17] Paskin Mark and Guestrin Carlos. Robust probabilistic inference in distributed systems. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 436–445, Banff, Canada, 2004. AUAI Press.
[18] C. Meek. Strong completeness and faithfulness in Bayesian networks. In *Proceedings of Eleventh Conference on Uncertainty in artificial Intelligence*, pages 411–418, Montreal, QU, August 1995. Morgan Kaufmann.
[19] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *In Proceedings of Uncertainty in AI*, pages 467–475, 1999.
[20] R.E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.
[21] J. Nunnink and G. Pavlin. A probabilistic approach to resource allocation in distributed fusion systems. In *Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS-05)*, pages 846–852, Utrecht, Netherlands, 2005.
[22] Jan Nunnink and Gregor Pavlin. Fault localization in bayesian networks. Technical report, 2006.
[23] G. Pavlin and J. Nunnink. Inference meta models: Towards robust information fusion with bayesian networks. In *Proceedings of the 9th IEEE/ISIF International Conference on Information Fusion*, Florence, Italy, July 2006.
[24] Gregor Pavlin, Patrick de Oude, Marinus Maris, Jan Nunnink, and Thomas Hood. A distributed approach to information fusion based on causal probabilistic models. Technical report, 2007.
[25] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
[26] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Computers*, 29(12):1104–1113, 1980.
[27] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, LTD, 2002.
[28] Y. Xiang. Verification of dag structures in cooperative belief network based multi-agent systems, 1998.
[29] Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.
[30] Huajie Zhang and Charles X. Ling. Geometric properties of naive bayes in nominal domains. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 588–599, London, UK, 2001. Springer-Verlag.