

A Probabilistic Approach to Resource Allocation in Distributed Fusion Systems

Jan R. J. Nunnink Gregor Pavlin
Informatics Institute, Faculty of Science
University of Amsterdam
{jnunnink,gpavlin}@science.uva.nl

Abstract

In complex multi-agent fusion systems resource conflicts are very likely to occur. We propose an algorithm that determines the optimal sensing resource to fusion task assignment, based on the expected change in entropy. By exploiting the Bayesian network framework and the structure of our agent network, the algorithm operates in a distributed manner by combining descriptions of local fusion models in an efficient way, which provides significant advantages over centralized approaches.

1. Introduction

With the recently increasing interest in large multi sensor networks, resource management has become an active research topic. Often, approaches to resource management exploit information theoretic criteria such as Shannon entropy or Kullback-Leibler divergence. Basically, they do a search over all possible sensor to task assignments and for each combination estimate the next posterior probability distribution over the possible states of the world, which is then used for determination of the information gain formulated using the change in entropy or divergence. These methods, however, usually require centralized processing and centralized knowledge of the state and structure of large networks of information sources. Consequently, centralized approaches are inherently vulnerable to potential problems such as single point of failure, and communication and computational bottlenecks. In addition, these approaches are computationally expensive because of the search and the estimation of the next belief state.

This paper, on the other hand, is dealing with resource allocation in the context of diagnostic information fusion within a network of agents. The fusion process is based on distributed Bayesian Networks (BN), which feature tree-like topologies with a significant portion of conditionally independent branches. Information obtained through sensory systems or other sources is inserted into the network and

fused at different levels of a hierarchy of agents in order to update the belief in a certain hypothesis. Multiple hierarchical fusion structures can be active simultaneously and therefore we will often have to deal with resource conflicts. Such conflicts can occur when multiple agents want to access and control the same information source with conflicting sensor parameters.

Obviously, centralized approaches to resource allocation are not suitable for such distributed and highly dynamic fusion networks. Therefore, we propose an efficient resource allocation method which can cope with dynamic distributed fusion structures. It features an allocation criterion based on the entropy change, which can be determined in a distributed manner through collaborating agents, without explicit knowledge of the underlying fusion structure. Efficient computation is based on parametric descriptions of the local BNs contained in different agents, which can be seamlessly combined into a fusion model that provides a mapping between nodes located in different agents. Moreover, each resource conflict is resolved locally. Namely, an information source (e.g. an agent with a direct access to sensory information) receives the estimates of information impact from each of the potential fusion structures and provides its information to that fusion structure which will improve the knowledge about its hypothesis the most. In other words, no centralized knowledge of resource conflicts is required, since the agents controlling conflicting information sources decide locally by using the knowledge of the global information impact.

The paper is organised as follows: in Section 2 we give an introduction to distributed fusion systems based on probabilistic models, we discuss the resource allocation problems and propose an approach to local resource allocation based on the impact of observations on the fusion results. In Section 3 we derive an efficient mathematical framework for the computation of the allocation criterion. Then we provide two possible algorithms for allocation using this framework in Section 4, and compare their efficiency in Section 5. Finally, in Section 6, we experimentally confirm the suitability of the presented approach to resource allocation, and

we end with conclusions in Section 7.

2. Bayesian Fusion

Since we deal with heterogeneous information associated with significant uncertainties we make use of Bayesian networks (BN), which provide a mapping between observed evidence and hidden events. A Bayesian network \mathcal{B} , is defined by a tuple $\langle DAG, JPD \rangle$. DAG is a directed acyclic graph $\langle \mathcal{X}, \mathcal{L} \rangle$, where $\mathcal{X} = \{X_1, \dots, X_n\}$ is the set of nodes and \mathcal{L} is the set of directed edges $\langle X_i, X_j \rangle$ of the graph. The nodes represent different events, and the links the causal relationships between the events. JPD is the joint probability distribution, defined by

$$JPD(\mathcal{X}) = \prod_{X \in \mathcal{X}} P(X|\pi(X)), \quad (1)$$

where $P(X|\pi(X))$ is the conditional probability distribution (CPD) for a node X given its parents in the network $\pi(X)$. Thus, the complete JPD can be represented by a set of CPDs, one for each node in the dependency graph. In other words, the graph describes the possible relationships, and the CPDs the strengths of those relationships. [7, 5]

2.1. Distributed Perception Networks

We investigate the fusion of heterogeneous information in the context of Distributed Perception Networks (DPN) [6], networks of agents which can fuse heterogeneous information through cooperative Distributed Problem Solving [3]. Each DPN network is a fusion structure which updates belief in a single hypothesis through cooperation of different agents, which in turn can roughly be classified in two types. At the lowest level there are different Sensor agents with direct access to sensors and sensor data interpretation capabilities while at the higher levels Fusion agents use local world models for the information fusion. The local world models in Fusion agents are encoded through Bayesian networks and represent basic world modeling building blocks.

One of the main DPN features is that the world model required for the fusion can be assembled and updated at runtime on an as-needed basis. Namely, different DPN agents can organize autonomously in complex organizations, i.e. DPN fusion structures. During such self configuration the local world models of different agents can be integrated into arbitrarily complex world models through appropriate message passing between the cooperating agents. Each agent updates its belief in a single event represented by a single root node in a local BN. The fusion result, i.e. the marginal distribution over the root node, is passed to higher level agents, which use it as an input in order to update belief in events corresponding to higher level concepts. In gen-

eral, we assume that each agent can contain a local multiply connected BN with a single root node. Furthermore, the DPN agents form agent networks with simple tree topologies. In other words, the BNs assembled from local BNs contained in the agents participating in a certain DPN network feature topologies with a significant portion of conditionally independent branches. Note that this is not a significant limitation with respect to the fusion problems we are interested in.

2.2. Resource Conflicts

In general, there can be several DPN fusion networks running in parallel, monitoring different aspects of a certain environment. Consequently, it can happen that more than one DPN would try to integrate a particular sensor agent. If such sensor agents are passive, i.e. no sensing parameters are changed by the DPN, or all fusion agents work with the same sensing parameters, then several DPNs can share a single sensor agent without any conflict resolution. However, in advanced sensing systems, several DPNs might want to share a sensor agent by setting different sensing parameters, which will inevitably result in resource allocation conflicts (see Figure 1). For example, a data stream from a pan-tilt camera might be used by a DPN monitoring the area to the North while another DPN would like to point the camera to the South. In order to deal with such situations in an efficient and robust way, we allow the conflicting sensor agent, i.e. the agent with a direct access to the information source, to determine which fusion network will be granted the rights to change the sensing parameters (e.g. observation direction) and use the data. Such decisions are based on the estimation of the impact the information source would have on the fusion results of each DPN fusion structure that is interested in Sensor agent's information. Basically, in the case of conflicting information requests from several DPN fusion structures, the information source asks each interested fusion structure to provide an estimate of the impact that the eventually provided information would have on its fusion results. The sensor agent grants the access to its information and sensing parameters to the fusion structure for which the supplied information would have the greatest impact.

2.3. Information Impact

In conflicting situations, a choice has to be made by a sensor agent about which fusion agent gets control over the sensor, i.e. to which agent the sensing resource gets allocated. The DPN framework places several requirements on any approach to this problem:

- It should be able to handle changes in the agent network.

- It should not be problem-specific, since it must apply to any DPN.
- It should be able to work in a distributed manner, without needing global knowledge.
- It should improve the quality of the fusion results.

Especially the last requirement poses some problems. After all, what is a measure for the fusion quality, and how to compute it?

One observation we can make is that the fusion process is equivalent to decreasing the uncertainty about the state of the environment. In other words, while fusing new information, we gather evidence to change our belief in certain states. Some states will become less likely, while others become more likely. So the more the fusion process advances, the greater the contrast between the beliefs will become. Therefore, this contrast can be seen as an indication of the quality of the current fusion result, since it shows how far the fusion has advanced. A common measure for the contrast within a vector of beliefs or a probability distribution is the *Shannon entropy* [8], defined by

$$\mathcal{H}(P(X)) = - \sum_{x_i \in X} P(x_i) \log P(x_i), \quad (2)$$

where $P(X)$ is a probability distribution over the variable $X = \{x_1, \dots, x_n\}$. It was first proposed as a measure of *information gain* in [4], and has since been used in many information theoretic approaches to resource and sensor management. For our approach we have therefore chosen to maximize the total absolute change in entropy for the belief in the hypotheses of both DPNs. Since the allocation of a sensing resource will only change the entropy of the DPN it gets assigned to, this is equivalent to allocating the resource to the DPN for which it causes the greatest absolute entropy change, defined by

$$\Delta\mathcal{H}(P(H)) = |\mathcal{H}(P(H|\mathcal{E})) - \mathcal{H}(P(H|\mathcal{E} \cup X))|, \quad (3)$$

where H is the DPN's hypothesis variable, \mathcal{E} is the set of all current evidence, X is the new evidence that will be obtained through allocation of the resource.

We use the absolute value of the change, because we want to avoid biasing our policy towards a certain type of probability distribution. Namely, if we would use (3) without the absolute operator, we would be biased towards allocations that would change the distribution towards a less uniform state, even though this change might be very small compared to a possible change towards a more uniform distribution. A change towards a more uniform distribution is not necessarily bad. After all, the previous evidence, which led to the current distribution, could have been misleading.

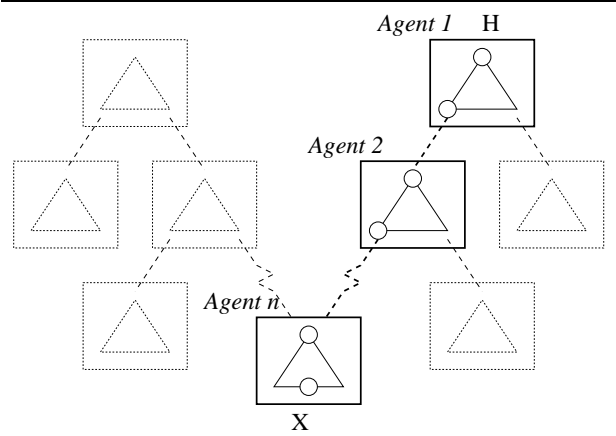


Figure 1. Two DPNs with one mutual agent n . The triangles represent the arbitrary networks within each agent. Highlighted is a chain of agents with the hypothesis node H , overlapping nodes (circles), and a leaf node under consideration X , which represents information from sensory systems or other sources. Agent n with the leaf node X can serve only one DPN fusion structure at a time.

3. Computation of Hypothesis Distribution

We define a chain of agents $\mathcal{A} = \{A_1, \dots, A_n\}$, where A_1 is the highest level agent containing the root hypothesis variable H , and A_n is the lowest level agent, containing the leaf variable under investigation, X . Each agent contains part of the complete Bayesian networks, and two adjacent agents in the chain overlap with exactly one variable (see Figure 1).

The hypothesis variable H can take the values h_1, \dots, h_m . We write R_j as the overlapping variable of agents A_{j-1} and A_j . This is also the root node of agent A_j . The set of all evidence is denoted by \mathcal{E} , where $X \in \mathcal{E}$.

The task is to provide the leaf agent A_n with knowledge of the posterior hypothesis probability distribution, $P(H|\mathcal{E})$, expressed as a function of the variable X . Using this function it is trivial for the agent to compute the change in entropy, $\Delta\mathcal{H}(P(H))$, for different values of variable X .

Let us first consider the monolithic BN, encoded by the complete chain of agents. We can compute the posterior probability as follows, using Bayes' rule:

$$P(H = h_i|\mathcal{E}) = \frac{P(H = h_i, \mathcal{E})}{\sum_{j=1}^m P(H = h_j, \mathcal{E})}. \quad (4)$$

For the entropy of the probability distribution over H , we need to compute (4) for each value h_i of H . This requires computing the joint probability for each value of H . Summing these joint probabilities over all values of H then gives the denominator of (4). In other words, agent A_n only needs to obtain knowledge of the function for $P(H = h_i, \mathcal{E})$ in terms of X , for each value of H .

While this mapping can be easily obtained in a monolithic BN, it is quite challenging to supply it to a leaf agent in an efficient and robust way. Namely, each agent has only knowledge of its own local model and it can compute only a partial mapping, namely the function of the probability of its root $P(R_j = r_i, \mathcal{E})$ in terms of its overlapping leaf node R_{j+1} , for each value r_i of R_j . In other words, the agents participating in a DPN fusion structure must be able to determine a mapping between arbitrary nodes in a chain of agents \mathcal{A} through collaboration.

3.1. Parametric Approach

Obviously, partial world models of all relevant agents must be combined in order to obtain mapping between the evidence and hypothesis nodes in a chain of agents \mathcal{A} . It turns out, that we can provide agent A_n with adequate mapping $P(H = h_i, \mathcal{E})$ in an efficient way by considering the fact that any joint probability of a BN can be viewed as a linear function of any network parameter (see [1, 2]). However, we are not interested in the function in terms of any model parameter, but rather in terms of the input variable X . This can be solved by treating the input node as a *soft evidence* node [7]. By adding a dummy child node to the input node, and encoding the soft evidence or likelihood into the corresponding CPD table, we effectively make the probability vector of the input node one of the network parameters. Therefore, we can write

$$\begin{aligned} P(R_j = r_i, \mathcal{E}) &= c_1 P(R_{j+1} = x_1) + \dots \\ &\quad + c_k P(R_{j+1} = x_k) \\ &= (c_1, \dots, c_k) \cdot \vec{P}(R_{j+1}), \end{aligned} \quad (5)$$

where (c_1, \dots, c_k) is a vector of coefficients. See Section 3.2 about how to compute these coefficients.

By expanding (5) to all values of R_j we get

$$\begin{aligned} P(R_j, \mathcal{E}) &= \begin{pmatrix} c_{11} & \dots & c_{1k} \\ \vdots & \ddots & \vdots \\ c_{n_j 1} & \dots & c_{n_j k} \end{pmatrix} \cdot P(R_{j+1}) \\ &= \mathbf{C}_{j+1}^j \cdot P(R_{j+1}), \end{aligned} \quad (6)$$

where n_j is the number of values of node R_j .

Thus, \mathbf{C}_{j+1}^j is the *local coefficients matrix* of agent A_j , where each row corresponds to a value of R_j , and each column to a value of R_{j+1} . In other words, \mathbf{C}_{j+1}^j is the trans-

formation matrix from input to output probability vectors, and thereby encodes the local network for one agent A_j .

We can use this equivalence for efficient computation of the mapping between any agents A_m and A_n in a chain \mathcal{A} , i.e. $P(R_m = r_i, \mathcal{E})$ as a function of R_{n+1} . Namely, we can show that we can compute for each agent A_i a matrix E_{i+1}^i specifying a local mapping in such a way that the mapping \mathbf{C}_n^m between arbitrary agents A_n and A_m can be computed through simple matrix multiplications $\mathbf{C}_{n+1}^m = \mathbf{C}_{m+1}^m \cdot \mathbf{E}_{m+2}^{m+1} \cdots \mathbf{E}_{n+1}^n$.

We show this property with the help of an example. Suppose that we have two adjacent agents A_j and A_{j+1} , each supporting local fusion. Agent A_{j+1} takes soft evidence, i.e. vector $w(R_{j+2})$, at its leaf node R_{j+2} as input and transforms it to its local joint distribution $P(R_{j+1}, R_{j+2})$ as follows:

$$\begin{aligned} P(R_{j+1}, w(R_{j+2})) &= P(R_{j+1}) \sum_{\Pi(R_{j+2})} P(\pi_k(R_{j+2})|R_{j+1}) \\ &\quad \cdot \sum_{R_{j+2}} P(R_{j+2}|\pi_k(R_{j+2}))w(R_{j+2}), \end{aligned} \quad (7)$$

where $\pi_k(R_{j+2})$ denotes the k -th parent node of the leaf node and $\Pi(R_{j+2})$ denotes the set of all parents of R_{j+2} . Similarly, agent A_j fuses soft evidence $w(R_{j+1})$

$$\begin{aligned} P(R_j, w(R_{j+1})) &= P(R_j) \sum_{\Pi(R_{j+1})} P(\pi_k(R_{j+1})|R_j) \\ &\quad \cdot \sum_{R_{j+1}} P(R_{j+1}|\pi_k(R_{j+1}))w(R_{j+1}). \end{aligned} \quad (8)$$

Moreover, agent A_{j+1} supplies agent A_j with an input vector which is computed as follows:

$$w(R_{j+1}) = P(R_{j+1}, w(R_{j+2})) / P(R_{j+1}), \quad (9)$$

which is a function of $w(R_{j+2})$. By computing $w(R_{j+1})$ in this way and plugging the result into (8) it is obvious that $P(R_j, w(R_{j+1}))$ of agent A_j is identical to $P(R_j, w(R_{j+2}))$, which would be obtained if we propagated the evidence vector $w(R_{j+2})$ through a network generated by merging the local networks of agents A_j and A_{j+1} :

$$\begin{aligned} P(R_j, w(R_{j+2})) &= P(R_j) \sum_{\Pi(R_{j+1})} P(\pi_k(R_{j+1})|R_j) \\ &\quad \cdot \sum_{R_{j+1}} P(R_{j+1}|\pi_k(R_{j+1})) \\ &\quad \cdot \sum_{\Pi(R_{j+2})} P(\pi_k(R_{j+2})|R_{j+1}) \\ &\quad \cdot \sum_{R_{j+2}} P(R_{j+2}|\pi_k(R_{j+2}))w(R_{j+2}). \end{aligned} \quad (10)$$

In addition, by considering the fact that we can compute a matrix encoding linear mapping between any pair of nodes in an arbitrary BN [1, 2] we can write mappings which are equivalent to equations (7), (8) and (10), respectively

$$P(R_{j+1}, w(R_{j+2})) = \mathbf{C}_{j+2}^{j+1} \cdot w(R_{j+2}), \quad (11)$$

$$P(R_j, w(R_{j+1})) = \mathbf{C}_{j+1}^j \cdot w(R_{j+1}), \quad (12)$$

$$P(R_j, w(R_{j+2})) = \mathbf{C}_{j+2}^j \cdot w(R_{j+2}), \quad (13)$$

where matrix \mathbf{C}_{j+2}^j describes the mapping (10) between the evidence $w(R_{j+2})$ and the hypothesis R_j in the monolithic BN, while matrices \mathbf{C}_{j+2}^{j+1} and \mathbf{C}_{j+1}^j correspond to equations (7) and (8), respectively. Obviously, we can use matrix \mathbf{C}_{j+2}^{j+1} to obtain the message $w(R_{j+1})$ passed between the agents A_{j+1} and A_j , which is identical to (9). We first obtain transformation matrix \mathbf{E}_{j+2}^{j+1} by dividing each row in \mathbf{C}_{j+2}^{j+1} , an $n \times m$ matrix, by the corresponding component of prior vector $P(R_{j+1}) = (P(r_{j+1}^1), \dots, P(r_{j+1}^m))$

$$(e_{k1}, \dots, e_{kn}) = (c_{k1}, \dots, c_{kn}) / P(r_{j+1}^k). \quad (14)$$

With \mathbf{E}_{j+2}^{j+1} we can compute $w(R_{j+1})$

$$w(R_{j+1}) = \mathbf{E}_{j+2}^{j+1} w(R_{j+2}). \quad (15)$$

Furthermore, we know that the combination of (7) and (8) through (9) is equivalent to (10). Given this equivalence and the fact that the result of (15) is identical to (9), we can conclude that the combination of (11) and (12) through (15) is equivalent to (13) and we can write:

$$\mathbf{C}_{j+2}^j w(R_{j+2}) = \mathbf{C}_{j+1}^j \mathbf{E}_{j+2}^{j+1} w(R_{j+2}). \quad (16)$$

Obviously, we can extend this factorization approach to any chain of N agents

$$\mathbf{C}_{j+N}^j w(R_{j+N}) = \mathbf{C}_{j+1}^j \mathbf{E}_{j+2}^{j+1} \dots \mathbf{E}_{j+N}^{j+N-1} w(R_{j+N}). \quad (17)$$

3.2. Computing the Coefficients

We can compute the coefficients as follows: recall that the function for the joint probability is

$$P(R_j = r_i, \mathcal{E}) = (c_1, \dots, c_k) \cdot P(R_{j+1}). \quad (18)$$

By choosing convenient values for the vector $P(R_{j+1})$ and doing any standard propagation, such as Junction Tree [5], on the network using that vector as evidence, we can compute the coefficients. For example, by setting $P(R_{j+1}) = (1, 0, \dots, 0)^T$, we get $P(R_j = r_i, \mathcal{E}) = c_1$, which makes c_1 equal to the propagation result. In a similar manner all the coefficients of the matrix \mathbf{C}_{j+1}^j can be computed.

Furthermore, the coefficients depend on both the network parameters and the evidence, and since the parameters are fixed the matrices can therefore only change if new (soft) evidence has been obtained. So if we describe the function of a joint probability in terms of all evidence nodes of a local network, then the coefficients of this function only need to be computed once. If we are then required to compute the coefficients for the function in terms of only one evidence node, we can simply fill in the last known evidence or likelihoods for the other evidence nodes. This will significantly decrease the computational complexity.

For example, suppose we have a local network with binary root node R and two binary evidence nodes, A and B . Then we can compute the transformation parameters c_1 , c_2 , c_3 and c_4 by solving the following system of linear equations:

$$P(r, \mathcal{E}) = c_1 P(a)P(\bar{b}) + c_2 P(\bar{a})P(b) + c_3 P(a)P(b) + c_4 P(\bar{a})P(\bar{b}). \quad (19)$$

In a situation where we need to compute $P(r, \mathcal{E})$ as a function of node A , we can obtain the required coefficients by filling in the (soft) evidence about node B . In this way we obtain the following function of A :

$$P(r, \mathcal{E}) = [c_1 P(b) + c_3 P(\bar{b})]P(a) + [c_2 P(b) + c_4 P(\bar{b})]P(\bar{a}). \quad (20)$$

4. Algorithms

As soon as an agent A_n supplying values based on input variable X with d possible states detects conflicting information requests from higher level DPN agents, it needs estimates of the impact of X on the uncertainty about the hypothesis H of each involved DPN. The comparison and resource allocation is based on the cumulative entropy change for each hypothesis $\sum_X \Delta \mathcal{H}(P(H))$, i.e. the sum of the entropy changes, each corresponding to a possible instantiation (i.e. hard evidence) of node X . Namely, prior to allocating the resource we do not know how the node X will be instantiated; depending on whether we add $X = x_i$ or $X = x_j$ to the evidence set, we will get different entropy changes. Finally, agent A_n grants the access to the control parameters and its information to the DPN fusion structure with the greatest $\sum_X \Delta \mathcal{H}(P(H))$. In addition, $\sum_X \Delta \mathcal{H}(P(H))$ depends on evidence nodes that were already instantiated when the allocation algorithm was executed. Since the evidence \mathcal{E} changes constantly, the allocation algorithm should repeatedly be executed within appropriate time intervals in order to reevaluate the potential impact of evidence X . Namely, information impact, i.e. the allocation criterion, can change due to new evidence from sources other than X and, consequently, an information source must be later reallocated to another DPN fu-

sion structure. In other words, the presented approach to allocation must adapt to the current instantiation. Clearly, repeated source allocation requires efficient algorithms for the determination of $\sum_X \Delta\mathcal{H}(P(H))$. We can identify two approaches to distributed computation of this quantity, both based on multi agent collaboration. One method exploits the parametric belief mapping described in the previous section, while another approach makes use of a traditional belief propagation mechanism, such as for example Junction Tree algorithm.

4.1. Algorithm 1

Using the analysis from the previous sections we can describe an algorithm, an efficient approach to distributed computation of $\sum_X \Delta\mathcal{H}(P(H))$, which is supplied to the conflicting agent. The algorithm can be splitted into three phases, as can be seen on the right.

4.2. Algorithm 2

In general, we could compute $\sum_X \Delta\mathcal{H}(P(H))$ directly through combining local belief propagations based on $\lambda - \pi$ or Junction Tree algorithms within each participating agent. Basically, given a leaf node X in agent A_n that can serve only one of the several interested higher level agents, we could gradually propagate the evidence through a chain of agents all the way to the hypothesis node for each possible instantiation of X , compute the entropy change and return the result to A_n . In general, for each of the possible d states of node X we have to run local belief propagation algorithm in agent A_{i+1} separately and send the partial fusion results to a higher level agent A_i . This approach to the determination of $\sum_X \Delta\mathcal{H}(P(H))$ is captured by the algorithm displayed below.

5. Performance Analysis

In this section we analyze and compare the performance of the presented algorithms in the context of computational as well as communication costs. We can show that determination of the evidence impact between arbitrary nodes in a distributed BN can be more efficient if we use the parametric method (Algorithm 1) instead of Algorithm 2 based on direct belief propagation.

It turns out that Algorithm 1 is more efficient than Algorithm 2 in terms computational costs, while both methods are associated with very similar communication costs. In order to facilitate further analysis we assume that all input and root nodes have the same dimension d .

Algorithm 1:

Preliminary Phase

If an agent is started up, compute the coefficients of the function for its local joint probability in terms of all evidence nodes (e.g. see (19))

Phase 1

if (a) a Sensor agent detects a conflict, or (b) a Fusion agent receives a request for the determination of the information impact **then**

if the agent has a parent agent **then**

Forward this request to the next agent on the path towards the hypothesis node

else

Using the coefficients from the Preliminary Phase, compute the local matrix \mathbf{C}_{j+1}^j by filling in the weights for the other evidence nodes (e.g. see (20)), and send it to the next agent on the path towards the conflicting agent

end

end

Phase 2

if an agent receives a matrix \mathbf{M} **then**

Using the coefficients from the Preliminary Phase, compute the local matrix \mathbf{C}_{j+1}^j by filling in the weights for the other evidence nodes (e.g. see (20))

Compute \mathbf{E}_{j+1}^j (see (14))

Compute $\mathbf{M}_{new} = \mathbf{M} \cdot \mathbf{E}_{j+1}^j$

if the agent is not the sensor agent **then**

Send \mathbf{M}_{new} to the next agent on the path towards the conflicting agent

else

Use \mathbf{M}_{new} to compute the entropy change for different instantiations and assign the resource by messaging the parent agents

end

end

5.1. Communication Effort

We define the communication costs through the number of parameters that have to be transported through inter-agent messages.

Given a chain of N agents, Algorithm 1 generates $N - 1$ messages, each containing $d \times d$ parameters of relative transformation matrices \mathbf{C}_{j+1}^j or \mathbf{E}_{j+1}^j . In addition, Phase 1 requires $N - 1$ messages propagated from the conflicting agent A_n all the way up to the hypothesis node. Thus, the communication costs C_1^c for Algorithm 1 can be expressed as:

$$C_1^c = (N - 1)(d \times d + 1). \quad (21)$$

In Algorithm 2, on the other hand, each agent in the chain of N agents receives d distributions from the sup-

Algorithm 2:

- 1 For each possible state of X agent A_n specifies hard evidence and computes the posterior distribution over its local root R_n . Thus, for each of the possible d states of X we obtain a distribution over states of R_n ;
 - 2 All d distributions of R_n are sent to the agents that are requesting information from A_n ;
 - 3 If agent A_i receives a message with d distributions of the root node R_{i+1} from the lower level agent A_{i+1} , then each of these distributions is used to compute d posterior distributions for the root node R_i ;
 - 4 **if** A_i has a parent A_{i-1} **then**
 All d distributions of R_i are sent via a message to A_{i-1}
else
 Compute $\sum_X \Delta\mathcal{H}(P(H))$ and send it back to agent A_n via the chain of agents that were participating in the gradual fusion process
end
-

plying agent, and computes d distributions over its local root node. Thus, the inputs as well as outputs are described through $d \times d$ parameters. If an agent does not have a parent, then it computes $\sum_X \Delta\mathcal{H}(P(H))$ for each of the d distributions received from the supplier agent and returns the result via the chain of agents to the conflicting agent. Therefore, the communication costs C_2^c for Algorithm 2 can be expressed as:

$$C_2^c = (N - 1)(d \times d + 1), \quad (22)$$

which is identical to C_1^c .

5.2. Computational Effort

We define the computational costs through the number of multiplications. In Algorithm 1 agent A_j has a set of parameters which are obtained at the initialization phase, prior to the actual fusion process (see for example (19)). These parameters and instantiations of leaf nodes of A_j are used for the computation of the local transformation matrix \mathbf{E}_{j+1}^j . In addition, the agent must multiply \mathbf{E}_{j+1}^j with $d \times d$ matrix $\mathbf{C}_2^1 \cdots \mathbf{E}_j^{j-1}$ supplied by agent A_{j-1} . This operation requires d^3 multiplications. Finally, as agent with the conflicting leaf node generates the mapping between the leaf node and the hypothesis node $\mathbf{C}_2^1 \cdots \mathbf{E}_{n-1}^{n-2} \mathbf{E}_n^{n-1}$, a $d \times d$ matrix which requires additional d^3 multiplications. Thus, the overall computational cost C_1^p for Algorithm 1 can be expressed as:

$$C_1^p = \sum_{j=1}^N (M_j^{a1} + d^3), \quad (23)$$

where M_j^{a1} denotes the number of multiplications required to compute \mathbf{E}_{j+1}^j based on the current instantiation of leaf nodes that do not belong to the path connecting the conflicting node and the hypothesis node. Note that this computation is based on the coefficients that were precomputed in the Preliminary Phase of Algorithm 1. It can be shown that M_j^{a1} , is exponential in the number of leaf nodes.

Algorithm 2, on the other hand, requires d local propagations, i.e. computations of the root node's distributions, in each agent. The net computational costs can be formulated as follows:

$$C_2^p = \sum_{j=1}^N d \cdot M_j^{a2}, \quad (24)$$

where M_j^{a2} denotes the number of multiplications required for local variable elimination in agent A_j .

It is well known that M_j^{a2} is exponential in the number of variables in the cliques of the moralized graph of the local BN. In addition, it is linear in the number of cliques. For each variable in a BN we can find a clique, containing this variable and its parents. Consequently, the cliques grow with the number of loops in a local BN. In other words, M_j^{a2} grows exponentially with the number of loops.

We see that the global computational cost C_1^p can be higher than C_2^p if a significant portion of the local BNs feature high branching factors and few nodes with multiple parents. Conversely, C_1^p can be smaller than C_2^p , if in a large portion of local BNs there exist cliques whose number of variables exceeds the number of local leaf nodes. Given these properties, we could improve Algorithm 1, by computing the local \mathbf{E}_{j+1}^j matrix via traditional local variable elimination if an agent used a BN with high branching factors and simple BN structure without loops. In this case, the difference of local computational costs would equal d^3 , corresponding to the additional local matrix multiplication in Algorithm 1, which is negligible if we deal with small d .

In addition, Algorithm 1 supports further improvement of efficiency if some agents do not receive any new evidence between two subsequent computations of $\sum_X \Delta\mathcal{H}(P(H))$. In such cases agents which did not receive new evidence do not have to recompute the local matrix \mathbf{E}_{j+1}^j . For these agents the corresponding $M_j^{a1} = 0$, which means additional reduction of computational costs. Algorithm 2, on the other hand, must recompute all root distributions, irrespective of local leaf node instantiations.

Also, Algorithm 2 requires a strict sequence of local variable eliminations throughout the chain of agents \mathcal{A} , since the elimination at agent A_j depends on the elimination at agent A_{j+1} . Obviously, this can result in timing bottlenecks. In the case of Algorithm 1, on the other hand, the local matrices \mathbf{E}_{j+1}^j are computed independently at every agent, which can speed up the determination of the information

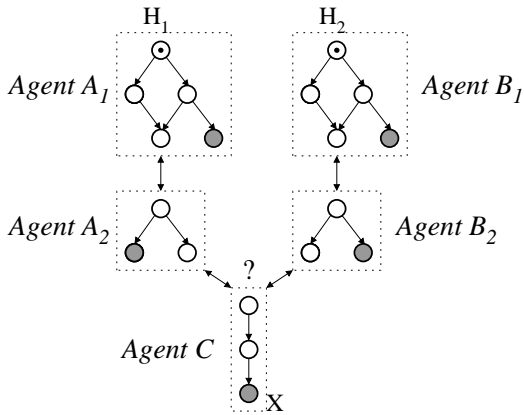


Figure 2. Two small example DPNs with a conflict over agent C . Shaded circles denote evidence nodes, and the dotted circles are hypothesis nodes.

impact significantly. Namely, the local matrices \mathbf{E}_{j+1}^j can be computed immediately after the agents received a request for the determination of the information impact, which was propagated from the conflicting agent \mathcal{A}_n throughout the chain of agents \mathcal{A} . In this way, each agent has time to compute its local matrix until it receives the matrix $\mathbf{C}_2^1 \cdots \mathbf{E}_j^{j-1}$ from agent A_{j-1} . In other words, a significant portion of the expensive processing steps required for the determination of $\sum_X \Delta \mathcal{H}(P(H))$ is carried out in parallel.

6. Example

In this section we show a simple example of how Algorithm 1 is used and how it improves decision making performance compared to a simple allocation policy. Consider Figure 2. Two DPNs are involved, the first consisting of agents A_1 and A_2 and the second of agents B_1 and B_2 . The grey nodes are evidence nodes and the dual headed arrows represent inter-agent communication channels. In this particular situation both DPNs have already observed their other evidence nodes' values and now want to incorporate agent C into their network. However, agent C is able to satisfy the information request of only one of the two, because they require agent C to use its sensor with different, and conflicting, parameters. A choice has to be made by agent C .

In this experiment we compare two allocation policies. The first is based on the change in entropy of the hypothesis variables of both DPNs, as computed by the algorithm explained in Section 4. The second policy simply connects agent C permanently to whichever DPN requested it first. Effectively, this is equivalent to a random allocation.

Allocation Policy	Correct Threshold
Entropy Based	34,3%
Random	24,6%

Table 1. Results of the experiment, showing the percentage of times a threshold was reached using different allocation policies.

We created 2000 DPNs, using the structure depicted in Figure 2, but with random CPDs, and compared the effects of applying the policies to the allocation problem. Since we want to have the system make decisions based on the belief in the hypotheses, we take as the quality criterion the percentage of times a correct threshold was reached. Here, 'correct' means the threshold corresponding to the class from which the particular data point was sampled.

The choice for this threshold is not trivial. If we set it too high, the percentages will be so small that the randomness of the experiment becomes too influential. On the other hand if we set it too low, both policies will probably reach it easily, and the difference will not be very clear. See Table 1 for the results. While the absolute difference is only 10 %, the relative difference is almost 40 %. Clearly, the maximum entropy change based policy outperforms the random policy.

7. Conclusion

We have proposed an approach to resource allocation in a certain class of multi agent Bayesian fusion systems, based on the impact of the new resource on the fusion result. We use the cumulative entropy change as a measure for this impact in DPN fusion structures, and our experimental results show that this is indeed an appropriate criterion for effective and efficient resource allocation.

In addition, we exploit the multi agent framework to compute the evidence impact in a collaborative manner. The allocation of information resources is carried out locally by the sensor agent that receives conflicting information requests from fusion agents. In other words, we avoid centralized allocation control, which results in improved efficiency and robustness.

Also, we proposed a parametric approach to computation of the cumulative entropy changes, which is very suitable for distributed fusion and can be significantly more efficient than traditional evidence propagation methods.

Acknowledgements

This work was done within the Combined Systems project at the Decis Lab, Delft, and is supported by

the technology program of the Dutch Ministry of Economic Affairs.

References

- [1] E. Castillo, J. M. Gutiérrez, and A. S. Hadi. Sensitivity analysis in discrete Bayesian Networks. *IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans*, 27:412–423, 1997.
- [2] V. M. H. Coupé and L. C. van der Gaag. Practicable sensitivity analysis of bayesian belief networks. In *Joint Session of the 6th Prague Symposium of Asymptotic Statistics and the 13th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 81–86, Prague, 1998.
- [3] E. Durfee, V. Lesser, and D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, 1989.
- [4] K. J. Hintz and E. S. McVey. Multi-process constrained estimation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(1):237–244, 1991.
- [5] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.
- [6] G. Pavlin, M. Maris, and J. Nunnink. An agent-based approach to distributed data and information fusion. In *IEEE/WIC/ACM Joint Conference on Intelligent Agent Technology*, pages 466–470, 2004.
- [7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [8] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana IL, 1949.