The Rumor Problem in Probabilistic Multiagent Systems CSCE Technical Report TR-2002-006

Mark Bloemeke
Marco Valtorta*
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208, U.S.A.
DRAFT VERSION 1.3

September 26, 2002

Abstract

We address the problem of rumors (i.e., redundant information) in systems of agents that use probabilistic models. The motivation for this work is to achieve an automatic way of compensating for rumors in multi-agent systems for data interpretation and fusion that communicate by exchanging beliefs, as in the Agent Encapsulated Bayesian Network (AEBN) model. After describing this model, we present two solutions. The *communication solution* requires passing extended messages that contain joint probabilities, while the *model solution* requires the modification of the Bayesian networks in each agent and the occasional passing of very large messages. We analyze the complexity of the two solutions and their trade-off using simple parameters.

Keywords: Bayesian Networks, Graphical Probabilistic Models, Multiagent Systems, Evidential Updating, Redundant Information, Redundant Influence, Rumors.

OUTLINE

1. INTRODUCTION AND MOTIVATION
2. RELATED WORK ON PROBABILISTIC AGENT SYSTEMS
   (A) XIANG'S AGENT TREES
   (B) COVARIANCE INTERSECTION
3. PROBABILISTIC MULTIAGENT SYSTEMS
   (A) AGENT ENCAPSULATED BAYESIAN NETWORKS
   (B) COMMUNICATION GRAPHS

*Corresponding author, mgv@cs.sc.edu

1 Introduction and Motivation

We address the problem of redundant influences in system of agents that use probabilistic models, which we call the rumor problem.

The motivation for this work is our desire to let multiagent systems for distributed interpretation and data fusion that use probabilistic models (and especially Bayesian nets) communicate with each other by exchanging beliefs.

Our agent model is called the Agent-Encapsulated Bayesian Network (AEBN) model and is originally due to Bloemeke [Blo98]. Each agent in an AEBN model uses as its model of the world a single Bayesian network (which we also call an AEBN). The agents communicate via message passing. Each message is a distribution on variables shared between the individual networks.

The variables of each AEBN are divided into three groups: those about which other agents have better knowledge (input set), those that are only used within the agent (local set), and those of which the agent has the best knowledge, and which other agents may want (output set). The variables in the input set and the output set are shared, while those in the local set are not. An agent consumes (or subscribes to) zero or more variables in the input set and produces (or publishes) zero or more variables in the output set.

The mechanism for integrating the view of the other agents on a shared variable is to replace the agent's current belief in that variable with that of the communicating agent. When an agent receives a message from a publisher, it modifies the probabilities in its internal model, so that its local distribution either becomes consistent with the other agent's view or is inconsistent with it.

When the communication graph (defined precisely in Section 3.2 is not a tree, great care must be taken to deal with undirected cycles (loops). Such cycles lead to possible double counting of information, known as the problem of redundant influences or the problem of redundant information or, simply, the rumor problem, which is the subject of this paper.

The rest of paper is organized as follows. In Section 2, we contrast our approach to those used two models for probabilistic distributed interpretation, Xiang's Multiply Sectioned Bayesian Networks (MSBNs, Section 2.1) and Julier and Uhlmann's Covariance Intersection Method (Section 2.2). A detailed presentation of our model for probabilistic multiagent systems, including the definition of communication graph, is given in Section 3. Section 4 contains an algorithm for finding redundant influences in communication graphs. Sections 4,5, and 6, contain the two solutions we developed, which we call the communication and model solutions, respectively. They are compared in Section 7. Section 8 contains a summary and evaluation of our work and suggestions for future work.

The paper is best read in conjunction with [VKV02].

2 Related Work

Although no one simple definition of an agent exists, most views of agents share two properties: one, agents are autonomous and possess a local view of the world; and two, agents pursue a local, possibly selfish, set of goals. Agent systems are thus built up from a group of individual agents by allowing them to communicate and gain information from each other.

This information can in turn help individual agents refine their world view and cause them to re-evaluate their goals. In this way a large reasoning problem in AI can be broken down into a series of smaller tasks that are performed locally with their results being shared globally using some kind of communication structure.

Therefore we can characterize an agent system by defining two things. The first is an internal method for knowledge representation used by the agent to maintain its view of the world and to tell it what actions to perform. The second trait of an agent system is the communication language and methods used to facilitate the sharing of information between agents within the system.

Three main models exist that attempt to distribute probabilistic calculation across multiple agents. The first one, proposed by Xiang [Xia96], involves using Bayesian networks as the local knowledge model and is most closely related to this work. It will be discussed in Section 2.1. The second approach is an extension to the well known Kalman filter technique [Kal60, Har90]. Known as the covariance intersection technique, it provides a way to fuse multiple estimates with no prior knowledge of the cross-correlation between those two estimates [JU97]. It is the subject of Section 2.2. The third approach (Multiple Entity Bayesian Network, MEBN), proposed by Laskey, Mahoney, and collaborators [LMW01], uses Bayesian network fragments, which are combined upon demand into a situation-specific Bayesian network, using a knowledge-based network construction algorithm.

2.1 Agent Trees

The most closely related approach to using Bayesian networks in an agent framework is summarized in Xiang's 1996 work entitled "A Probabilistic Framework for Cooperative Multi-Agent Distributed Interpretation and Optimization of Communication" [Xia96] and several other papers on Multiply Sectioned Bayesian Networks (MSBNs). Despite the similarity of this work, as implied by the title of the article, Xiang starts with a different set of assumptions that lead him to a much different definition of the agent system. In a recent paper [XL00], Xiang (with Lesser) describes his assumptions (which he calls basic commitments) and their consequences rigorously and in admirable detail. The assumption that we do not follow in our work is the second, i.e., that agents exchange their beliefs on shared variables *in any order* whatsoever. We elaborate on this point and its implication in the remainder of this section.

Xiang uses the agent terminology to simplify his discussion of distributed Bayesian network propagation. In his work there is no implication of the agents being individual entities for the purposes of design or construction of the global agent system. Instead each node of a clique tree, whose formation is not discussed, is assigned to a processor and is called an agent. Thus as opposed to having one distinct, locally designed, Bayesian network encapsulated within each agent, each agent in fact just contains a piece of a larger, globally designed, Bayesian network.

For Xiang's method to be valid, the global Bayesian network (i.e., the agent system with all agent based boundaries removed) must conform to a hypertree structure. This implies that each agent is itself a junction tree and that the overall structure is a tree, or forest, of junction trees. If this condition is met, then the global Bayesian network is said to have *soundness of sectioning*, and Xiang's distributed algorithm will work.

Local design is severely affected by the requirement of soundness of sectioning. The most obvious restriction is that any agent that shares more than one variable must be certain that its sharing of variables will not cause the agent graph to become multiply connected. In other words if a situation arises where two individual agents, who share an ancestor, want to access the same variable, then either only one may be allowed access, or the two agents must be merged, most likely along with some of their ancestor and descendant, into a single new agent.

Given the definition of the junction forest, the majority of Xiang's work focuses on co-ordinating message passing and synthesizing correct belief from the messages which are themselves the joint probabilities on the variables shared between two agents. In the synchronous form of Xiang's work, four local procedures are introduced: DistributeEvidence, UnifyBelief, UpdateBelief, and DistributeBelief. These procedures either start the messages or receive the messages and update the local junction tree based on the information contained in the message. They are triggered either at the beginning to initialize the system or locally whenever a new observation is made. Given these four functions (in fact there are several more used for optimizing the system with respect to initialization and allowing multiple entries of evidence asynchronously), Xiang provides algorithms that co-ordinate the passing of the messages and minimize off-line time (the time when an agent is updating locally and can't receive any more messages) of each agent.

Xiang's work has the advantage of allowing what we calls subscribers to influence the final belief in a variable, as opposed to the approach presented in this paper, where we assume that only one agent, the publisher, has correct knowledge of value of each shared variable. In addition to the above restriction that each agent system form hypertree with its implications on design, it is also highly likely that Xiang's work requires greater communication cost, since it must always communicate the joints of the shared variables, instead of the single beliefs in each variable, and this must be done multiple times for a single update.

Conversely, the work presented in this paper has several limitations with respect to Xiang's work. The major one is the assumption of additional independence relations not implied by the definition of a Bayesian network. This means that the beliefs in variable values calculated locally in each of the agents are not the same as those that would be calculated if all the agents were merged into a large single Bayesian network. These additional independencies follow from the assumption that certain agents are the sole arbiters of some variables beliefs. That is, an agent has oracular knowledge of some variables values and no other agent has any influence on them, except through contributions of its own belief on other shared variables. With this assumption of oracular knowledge the additional independence relationships are assumed between the variables being shared by one agent and the internal variables in the receiving agent.

The final difference between the two approaches (based on AEBNs and MS-BNs, respectively) is that Xiang need not concern himself with the rumor problem, because redundant influences only affects message propagation when multiple (directed) paths exist between two agents in the communication graph (formally defined in Section 3.2, and therefore they do not occur in tree structured graphs.

To summarize, the two approaches are based on essentially different assumptions. It seems to us that MSBNs were introduced as a method for parallel distributed inference within a single Bayesian network. This original purpose shows in the basic commitments that define MSBNs and negatively affects their suitability as a model for collaboration in multiagent systme. On the other hand, AEBNs seek to merge several distinct Bayesian networks and allow the sharing of information between them.

2.2 Covariance Intersection

In the area of data fusion and control theory there exists a well known method for maintaining a probability distribution describing the location of an object, or likelihood of an event, as a multi-variate Gaussian [Kal60, Har90]. This method, the Kalman filter, maintains two distributions, one representing the previous knowledge of the object and the other representing the current knowledge of the object, and at each step in time fuses these together to form a new belief in the object.

This fusion is accomplished using a simple linear combination of the two original distribution's mean vectors and co-variance matrices. An additional cross-covariance matrix is used to correct the linear approximation in the presence of correlation between the two original variances. In this way an estimate of the distribution of interest can be generated that is less corrupted by error than either of the two original estimates.

From a graphical models point of view, the Kalman filter may best be described as a special kind of continuous Gaussian Bayesian network, whose structure is described in Figure 1. The directed acyclic graph (DAG) in Figure 1 also describes the Hidden Markov Model (HMM) as a special kind of Bayesian network. The difference is in the fact that the HMM uses discrete probability throughout, as in (non-Gaussian) Bayesian networks.

Therefore the Kalman filter can be viewed as a special case of the Bayesian network within the fairly limited domain of data fusion. It does not, however, have the general expressive meaning of a Bayesian network or the ability to capture any non-parametric, or even just non-Gaussian, relation between the variables that the space is defined on.

As Maybeck states, some of the restrictions (linear model with *white*, i.e., uncorrelated, Gaussian noise) "can be relaxed, yielding a qualified optimal filter. For instance, if the Gaussian assumption is removed, the Kalman filter can be

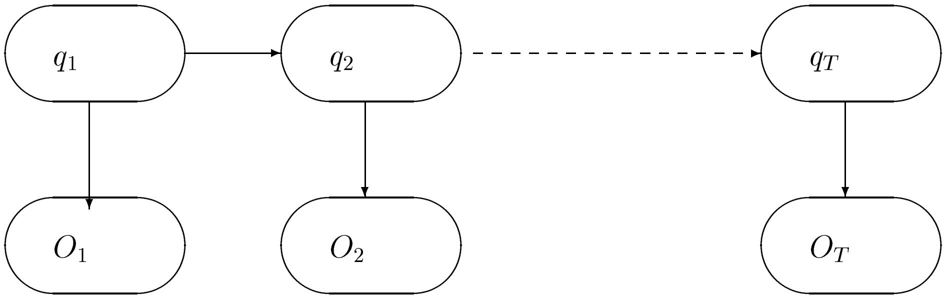


Figure 1: The Kalman filter as a Bayesian network

shown to be the best (minimum error variance) filter out of the class of linear unbiased filters" [May79, p.7].

There is one expansion to the Kalman filter that provides an interesting solution to the rumor problem. When there exist multiple estimates of the target distribution, which can be represented in a DAG structure similar to that used to represent the independencies in a Bayesian network, they can be fused using a method known as covariance intersection. This method takes advantage of the fact that for linearly fused Gaussian distributions the resulting covariance matrix always falls within the convex hull of the points of intersection of covariances of the original two distributions. This restriction on the final form of the distribution holds regardless of the cross-correlation between the two matrices [JU97], and therefore a method exists to create correct distributions from two arbitrarily correlated initial distributions.

This means that multiple beliefs in the distribution's true state can be fused in any order without any worry about the resulting distribution containing redundant information. The covariance intersection is unique in having the property that it works regardless of the correlation between beliefs. It can therefore be used in fusing estimates about a quantity in decentralized data fusion systems even when both the cross-covariance (degree of dependence) between estimates and the topology of the communication graph are not known. This, however, occurs not because the rumors are accounted for but rather due to the previously mentioned property of linear combinations of Gaussian distributions. In other words the rumor problem goes away not by dealing with it and removing the influences, but rather as a result of the fact that the combination algorithm always returns an estimate that is superior to either of the initial, un-fused, beliefs.

3 Probabilistic Multiagent Systems

This Section introduces a method to link multiple, individually designed, Bayesian networks together to form an agent system. The underlying premise is that each agent will use for its knowledge model of the world a single Bayesian network. The networks in a system will communicate with each other through the passing of messages, which in turn will be distributions on variables shared between the individual networks.

3.1 Agent Encapsulated Bayesian Networks

An Agent Encapsulated Bayesian Network (AEBN) is an agent that relies on a Bayesian network for its internal representation of the world. How this representation is used in decision support or goal based planning is unimportant so long as the world view is updated based only on local observations and observations derived from probabilistic messages. In this way the internal Bayesian network is functioning in a normative manner that is familiar to makers of Bayesian network based expert systems.

In an AEBN system the agents communicate through the transmission of probabilities along shared variables. This allows the division of the Bayesian network of each agent into three sets of variables: those about which other agents have better knowledge (I); those which are used only within the agent (L); and those that this agent has the best knowledge of and which other agents may want (O). This effectively produces two classes of variables in the agent: its local variables, L, and its shared variables, I and O.

The mechanism for integrating the view of the other agents on a shared variable is to simply replace the agent's current belief in the variable with that of the communicating agent. For this reason all communication in the graph occurs through the passing of messages that essentially contain the "correct" views on some shared variables. When an agent receives one of these messages, it modifies its internal model so that its local distribution either becomes consistent with the other agents' view or becomes inconsistent by entering a zero probability configuration.

This method of sharing belief can be characterized as a producer-consumer relationship in which one agent has complete knowledge of the correct state, or distribution, of a variable and shares that with one or more agents who wish to integrate that belief into their local models. The agent who has this knowledge of a variable is called the publisher and the agents receiving these messages are known as subscribers. When a new observation is made by a publisher, the agent sends a message indicating the observation to its subscribers. The subscribers in turn adjust their internal view of the world and send their published variables on to their subscribers. It is in this way that the agent system returns to equilibrium (assuming no directed loops in the communication).

At first this restriction that one of the agents has an oracular knowledge of the value of one or more variables may seem to be an excessively restrictive assumption. It seems to imply that there can be no mediated belief, among multiple agents, over a common variable. However, this is not the case. It is entirely permissible for multiple agents to have their own view of a common variable so long as each such view bears its own label.

For example consider a situation in which four distinct agents are working together in an attempt to locate a target in a two dimensional space. Local design of the four agents might yield the four Bayesian Networks, of three nodes each, seen in Figure 2 whose variables are defined in Figure 3.

Within this collection of Bayesian Networks there exist four separate beliefs in the location of the target. Each of the separate beliefs represents a local, sometimes shared, belief, with the final belief being computed by fusing those local beliefs. In this way multiple, individually labeled, beliefs in the same event

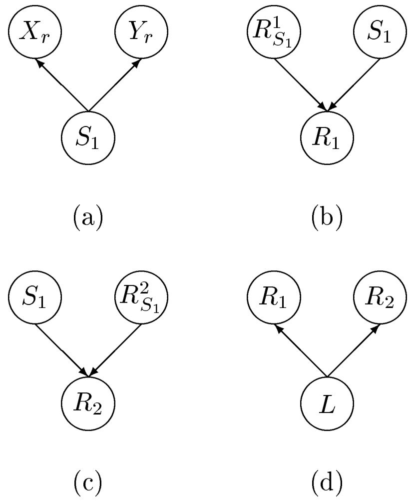


Figure 2: Four agents in a tracking scenario.

(a target's location) occur in the agent system without violating the producer-consumer view of knowledge. Thus the assumption of oracular knowledge is more readily stated as each agent having absolute confidence that the message received is the un-shakeable belief of the producing agent. Thus each agent is not so much omniscient as obstinate.

In a related paper [VKV02], we describe how to update an agent's probability distribution upon receipt of messages from other agents. (The messages include what is called *soft evidence* in [VKV02].) In particular, we will assume that *observation variables* have been introduced as needed. Therefore, each agent that receives messages from other agents obtains soft evidence for one or more observation variables.

Using the oracular view of agents stated above, an alternate view of the Bayesian network within each agent is now presented. Since each agent functions as a means to publish a set of variables (the output variables O) given another set of variables (the inputs I) each agent's Bayesian Network functions like a conditional probability table of O given I (i.e., $P(O|I)$). This "table" is computed using a local propagation algorithm on the agent's internal Bayesian network given its local observations on its local variables L.

This view of using conditional probability to update a Bayesian networks distribution arises naturally from the producer/consumer definition of shared variables and the assumption of d-separation of the observation variables. To update an agent's distribution $P(V)$ with new evidence $Q(E_1, E_2, \ldots, E_n)$ for some set of variables $\{E_1, E_2, \ldots, E_n\} = I$ one calculates the joint probability $P(V)$, dividing by the marginal probability $P(W)$, and multiplying it by the new distribution of $\{E_1, E_2, \ldots, E_n\}$,

$$Q(E_1, E_2, \ldots, E_n) = Q(E_1) \cdot Q(E_2) \cdot \ldots \cdot Q(E_n), \tag{1}$$

Agent A(figure 2.a) :		
X_1	–	A distance sensor along the X-axis.
Y_1	–	A distance sensor along the Y-axis.
S_1	–	A discrete distribution along a 2-dimensional space generated by fusing X_1 and Y_1, that indicates the likely location of the target.
Agent B(figure 2.b):		
S_1	–	The location of the object as reported by Agent A.
$R^1_{S_1}$	–	Some local estimate of the reliability of Agent A.
R_1	–	The report generated by Agent B based on Agent A's belief in the location of the target and Agent B's belief in the reliability of Agent A.
Agent C(figure 2.c):		
S_1	–	The location of the object as reported by Agent A.
$R^2_{S_1}$	–	Some local estimate of the reliability of Agent A.
R_2	–	The report generated by Agent C based on Agent A's belief in the location of the target and Agent C's belief in the reliability of Agent A.
Agent D(figure 2.d):		
R_1	–	Agent B's belief in the location of the target
R_2	–	Agent C's belief in the location of the target.
L	–	Agent D's fused belief in the location of the target (Note: causal reasoning is shown in the assumption that the true location influences the reports received instead of the reverse approach of building the belief in the location from the reports).

Figure 3: Definition of the variables for Bayesian networks of Figure 2

thus obtaining:

$$Q(V) = P(V \setminus I|E_1, \ldots, E_n) \cdot Q(E_1, \ldots, E_n) = \frac{P(V \setminus I)}{P(E_1 \ldots, E_n)} \cdot Q(E_1, \ldots, E_n). \tag{2}$$

In the case in which the input variables are not d-separated in the receiving agent, Equation 1 does not hold. (See [VKV02, Section 5] for a detailed discussion of this point.) Lemma 1 in [VKV02] allows the replacement of Equation 2 by

$$Q^*(V) = P(V \setminus I|E_1, \ldots, E_n) \cdot Q(E_1, \ldots, E_n) = \frac{P(V \setminus I)}{P(E_1 \ldots, E_n)} \cdot Q^*_I(E_1, \ldots, E_n), \tag{3}$$

where Q^*_I is the I_1-projection of probability distribution P on the set of all distributions defined on I and having $Q(E_i)$, $i = 1, \ldots, n$, as theis marginals. In practice, $P(V)$ could be updated to $Q^*(V)$ using the *big clique algorithm* of [VKV02].

Thus a mechanism similar to that which is already used for updating probabilities in a Bayesian network adjusts the world view of the agent, $P(V)$, into a conditional probability table $P(O|I)$. It then combines that table with the external view of the inputs, $Q(I)$, to allow the calculation of the new values for the output variables $Q(O)$.

Given this view of the purpose of each agent in the overall system, the

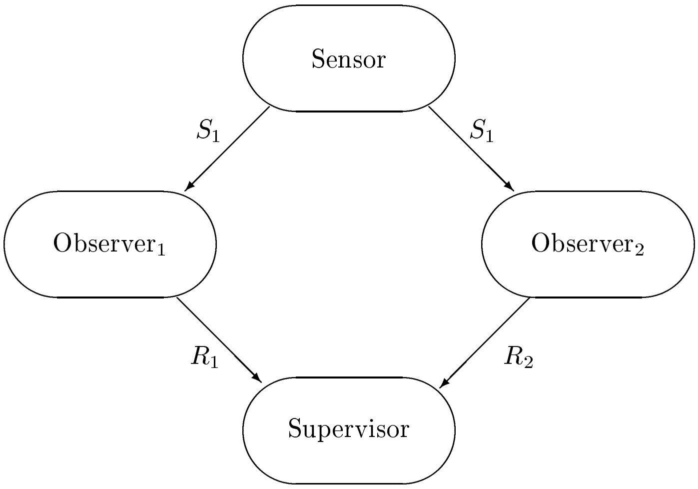


Figure 4: Redundantly Observed Sensor Example (R.O.S.E.) Communication Graph

natural view is that an agent system is a simple expansion of the Bayesian network formalism to a DAG in which one conditions on sets of variables (the input variables) rather than individual variables. This is not strictly the case for two reasons.

First, the oracular assumption imposes the additional constraint that, in the agent system, unlike a Bayesian network, all parents are independent of their descendants. This notion will be made more precise when we introduce the More precisely, the only variables that may affect the variables in an agent are (1) those in the agent itself and (2) those in a preceding agent. The notion of "preceding agent" will be made more formal in Section 3.2.

Second, when input variables are not independent in the receiving agent, then the calibration equation 2 must be replaced by the formally identical, but substantially and computationally more complex equation 3.

3.2 Communication Graphs

In order to represent the message passing and updating implications of AEBN's, we define a graphical representation of the agent system, called a *communication graph*. This graph is a DAG whose nodes are the agents and where edges are drawn from a publisher of a shared variable to each of the variable's subscribers. These edges are in turn labeled with the variable that they share. It should be noted that in the communication graph it is possible for more than one edge to exist between two nodes, though each of those edges will be uniquely labeled.

We can now formalize the constraint stated at the end of Section 3.1 that, in the agent system, all variables are parents are independent of their descendants. Let A_i and A_j be two distinct agents, let V_i, V_j be the sets of variables in agent A_i and A_j, respectively, and let $W_i \subseteq V_i$, $W_j \subseteq V_j$, Then, $W_i \perp\!\!\!\perp W_j$ if there is no directed path in the communication graph from W_j to W_i.

Using our example of an agent system from figures 2 and 3, a communication graph is constructed by first identifying the shared variables (S_1,R_1, and R_2), then directing labeled edges from the producing agents to the consuming agents. In our case edges are directed from Agent A to Agent B labeled with S_1, Agent A to Agent C labeled with S_1, Agent B to Agent D labeled with R_1, and Agent C to Agent D labeled with R_2. This leads to the communication graph seen in Figure 4. Henceforth this example will be referred to as the Redundantly Observed Sensor Example (R.O.S.E.) with Agent A being known as Sensor$_1$, Agent B as Observer$_1$, Agent C as Observer$_2$, and Agent D as Supervisor.

It is within the communication graph that the nature of the rumor problem can be clearly understood. In this graph the problem centers on the fact that the Supervisor's view of the world, held in its Bayesian network, is doubly influenced by the initial sensor reading. This can be illustrated clearly by using the alternate view of agents as conditional probability tables, proposed in Section 3, to look at what the Supervisor would compute. The Supervisor computes its belief in L, the location of the target, as

$$P(L) = \sum_{I} P(L|I)P(I) \tag{4}$$

which expands to

$$P(L) = \sum_{R_1,R_2} P(L|R_1,R_2)m(\text{Observer}_1)m(\text{Observer}_2) \tag{5}$$

where $m(\text{Observer}_1)$ and $m(\text{Observer}_2)$ are passed in as messages from Observer$_1$ and Observer$_2$. Expanding $m(\text{Observer}_1)$ yields

$$m(\text{Observer}_1) = \sum_{S_1} P(R_1|S_1)m(\text{Sensor}) \tag{6}$$

with $m(\text{Observer}_2)$ being calculated as

$$m(\text{Observer}_2) = \sum_{S_1} P(R_2|S_1)m(\text{Sensor}) \tag{7}$$

Finally, expansion of $m(\text{Sensor})$ yields

$$m(\text{Sensor}) = P(S_1) \tag{8}$$

Substitution of equations 6 and 7 into equation 5 leaves us with the intermediate equation

$$P(L) = \sum_{R_1,R_2} P(L|R_1,R_2)\sum_{S_1} P(R_1|S_1)m(\text{Sensor})\sum_{S_1} P(R_2|S_1)m(\text{Sensor}) \tag{9}$$

Substituting equation 8 into 9 yields

$$P(L) = \sum_{R_1,R_2} P(L|R_1,R_2)\sum_{S_1} P(R_1|S_1)P(S_1)\sum_{S_1} P(R_2|S_1)P(S_1) \tag{10}$$

Finally, pulling the sums out leaves the following equation for $P(L)$

$$P(L) = \sum_{R_1,R_2,S_1} P(L|R_1,R_2)P(R_1|S_1)P(R_2|S_1)P(S_1)P(S_1) \tag{11}$$

as opposed to

$$P(L) = \sum_{R_1,R_2,S_1} P(L|R_1,R_2)P(R_1|S_1)P(R_2|S_1)P(S_1) \tag{12}$$

where Equation 12 is the desirable outcome of message passing in the agent system and Equation 11 is the actual outcome. Further, this problem can be made arbitrarily worse simply by adding additional paths between the Sensor and the Supervisor.

The principal objective of this paper is to allow the handling of rumor problem in an automatic way. This will be done using algorithms that function using the communication graph to identify redundant influences (Section 4) and then using either purely communication based solutions (Section 5) or automatic local model modification (Section 6) to eliminate them.

4 Redundant Influences

This Section describes how to identify redundant influences. As we saw at the end of Section 3.2, redundant influences arise in the graph any time the combination of messages at some node causes the belief in some shared variable to be over included. In order to rectify this situation, we first have to identify the routes through which the influences arrive.

More precisely, we say that there is a *redundant influence between nodes V_i and V_j* relative to some communication graph G if there exist multiple node-disjoint paths between V_i and V_j in G. (We will routinely drop G when we use this definition.)

We define the *order* of the redundant influence between V_i and V_j as the number of node-disjoint paths between V_i and V_j. This definition is justified by the following considerations.

We start this Section by characterizing thoroughly the routes through which redundant influences arrive at a node in the communication graph.

Assume that we have a graph such that between V_i and V_j there are only n node disjoint paths $\{V_i \to V_{1_1} \to \ldots \to V_{1_k} \to V_j, \ldots, V_i \to V_{n_1} \to \ldots \to V_{n_l} \to V_j\}$, where k is the length of path 1 minus the endpoints and l is the length of path n minus the endpoints. Yet we have more than n redundant influences. Clearly since redundant influences must occur along some series of paths, there must be some cycles that are not node disjoint between V_i and V_j causing the additional redundancies. Each of these additional cycles must take on one of four forms (assume $1 \leq q, p \leq n$):

1. It starts at one vertex along node disjoint path q and ends at a different vertex along path p.

2. It starts at one vertex along path q and ends at V_j.

3. It starts at V_i and ends along a node disjoint path q.

4. It starts and ends along node disjoint path q.

In all four cases the same essential argument applies that the redundant influences occur in some subgraph that does not include either V_i or V_j. In the first case, there exist a subgraph rooted at V_i and ending at some node $V_{j_?}$ along which we have (at least) two node-disjoint paths starting at V_i and ending at $V_{j_?}$ which are amplifying the influence, but these influences do not carry through to V_j. They are compensated for locally in the subgraph rooted at V_i and ending at $V_{j_?}$.

In the second case we once again have a subgraph anchored at $V_{i_?}$ and ending at V_j which would amplify the influence of V_i on V_j. Once again these influences do not carry through from V_i and they as they are compensated for locally in the subgraph rooted at $V_{i_?}$ and ending at V_j.

The third and fourth cases are handled by a similar argument in that the paths belong to a subgraph, either from V_i to $V_{i_?}$ or from V_{i_k} to V_{i_l}, which is a proper sub-graph of the one rooted at V_i and terminating at V_j, and therefore are compensated for.

A consequence of these considerations is that only node disjoint paths need to be identified and labeled to compensate for redundant influences. The following algorithm can be used to detect and label node disjoint paths in an communication graph. Once this algorithm has been run a new graph, known as the redundancy graph, is constructed. This graph will be used, along with the original communication graph, in both solutions to the rumor problem.

The following algorithm extends the communication grapy into a *redundancy graph*. The redundancy graph has the same nodes and edges as the communication graph, but its labels are extended if and only if there are redundant influences. Start by creating a copy of the communication graph that will serve as the redundancy graph, and whose labels will be extended as described below. For each pair of vertices v_s and v_t in the redundancy graph take each variable s_i that is produced by v_s and:

1. Create a copy of the graph for use in the flow problem.
2. Designate v_s as the source for the flow problem and v_t as the sink for the flow problem.
3. Set the maximum flow of each edge to 1.
4. Set the maximum flow to 0 for all edges flowing out of v_s that are labeled with a variable other than s_i in the communication graph.
5. Run the flow problem.
6. If the flow into the sink is greater than 1, then redundant influences exist between the two vertices and the next step should be applied. Otherwise go to the next v_s, v_t, s_i combination and run this algorithm again.
7. For each edge in the flow problem solution that has flow greater than zero (and therefore by definition is on a node disjoint path), add the shared variable s_i to the label of the corresponding edge in the redundancy graph.

The above algorithm is correct if and only if the flow problem indicates a positive flow for all edges, and only the edges, along node disjoint paths from v_s to v_t. This is known to be the case [Koz92, Chap. 16], and therefore the above algorithm will expand the label of all node disjoint paths, between v_s and v_t, which by definition are also the redundant influences, by the shared variable that causes redundant influence through that edge.

The time complexity of the flow problem is $O(nm\log(\frac{n^2}{m}))$ where $n = |V|$ and $m = |E|$. Since we can perform both initialization and set up of the graph information in $O(n+m)$, $O(nm\log(\frac{n^2}{m}))$ dominates the loop body. Since the loop body is executed for each pair of nodes, the total time for the above algorithm is $O(n^3 m\log(\frac{n^2}{m}))$. This time is within $O(n^5)$ because m in the worse case is n^2. Given that this is a distributed system, algorithms exist to find the maximum flow in $O(n^2\log^3 n)$ time using $O(n^2(\log^3 n + \sqrt{m}))$ communication complexity [Mot95] assuming that each node in the communication graph has its own processor and knowledge of the whole graph (which can be accumulated in no worse than $O(m)$ time).

It is important to note that this is the only step in an AEBN system that requires global knowledge of the network structure. After the edges have been labeled no further knowledge outside of the immediate neighborhood is necessary. Further, once the redundancy graph has been built, all remaining algorithms can be run asynchronously. The only restriction is that each agent can update its local distribution from only one message at a time, and thus some lag time may arise in propagation.

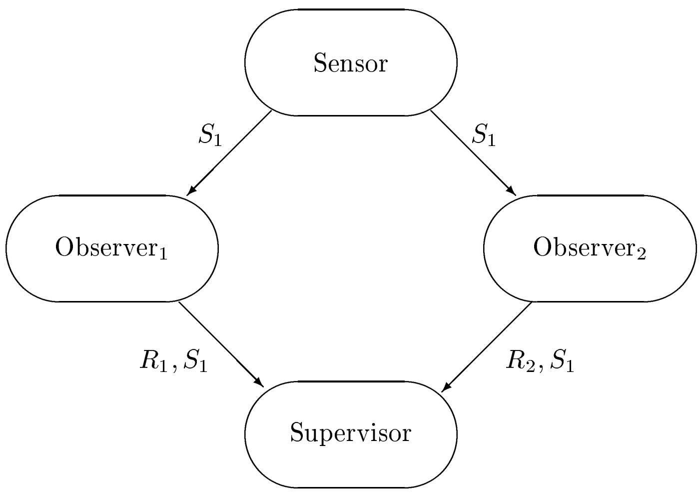


Figure 5: The R.O.S.E. Redundancy Graph

Considering the R.O.S.E. example of Figure 4, the algorithm in the previous section produces the redundancy graph of Figure 5. In this graph only two edges have expanded labels. This arises because only in the case of the cycle (i.e., v_s = Sensor, v_t = Supervisor) will the flow problem return a maximum flow greater than 1. In this case all four of the edges between the Sensor node and the Supervisor node will have a flow of 1 and therefore will have S_1 added to there label.

5 Communication Solution

This Section proposes a method for compensating for redundant influences. Figure 5 illustrates the R.O.S.E. model after the above algorithm has been run to detect redundant influences. In this model communication has been expanded to pass joint probabilities (i.e., $P(R_1, S_1), P(R_2, S_1)$) along the appropriately labeled links. This requires that the algorithm that is used in each table to compute the probability of the output parameters $P(O)$ be flexible enough to allow the calculation of joints involving a fixed input and the output variables.

The calculation of joints is not trivial, especially in the presence of soft evidence. The *factor tree* algorithm of [Blo98, BV98] was designed especially to support the computation of one or several joint probabilities and of all single-variable marginals. It was extended in [BV] to handle soft evidence. Alternatively, one of the two soft evidential update algorithms in [VKV02], which modify the junction tree method, could be used. These algorithms are designed primarily to compute all single variable marginals, but they can be used to compute one or several joint probabilities using techniques such as value or variable propagation, described in [Jen95, Section 5.1].

Given the labeling, the correct (i.e., not redundantly influenced) probabilities can be retrieved at each node without any modification, so long as the message that travels each node is the joint of its labels. Since no local Bayesian Network model modification is necessary, this is known as the communication solution. The algorithm for the retrieval of these correct probabilities follows.

For any of the m nodes, n_i $(1 \leq i \leq m)$ in the communication graph, there are k_i, $0 \leq k_i \leq m-1$, incoming edges, $e_{(i,j)}$, $0 \leq j \leq k_i$. Each of these edges carries a message $m_{(i,j)}$ that is a joint probability table across the shared variables that make up the edge's label in the redundancy graph ($\Psi(m_{(i,j)})$ is a function to retrieve this label). Start by defining $\mathcal{R}_i$ to be the set of all variables that arrive in an incoming message to the agent represented by n_i but that are not used by the local Bayesian Network (i.e., not members of I). More formally $\mathcal{R}_i$ is

$$\mathcal{R}_i = \bigcup_{0 \leq j \leq k_i} \Psi(m_{(i,j)}) - I_i.$$

For each variable $\rho_l \in \mathcal{R}_i$ perform the following steps:

1. Set c_l to the number of sets $\Psi(m_{(i,j)}), 0 \leq j \leq k_i$, that have ρ_l as an element.

2. If c_l is larger that one:

   (a) Select one of the messages $m_{(i,j)}$ that contains ρ_l and marginalize it to retrieve $P(\rho_l)$.

   (b) Condition all other messages containing ρ_l with $P(\rho_l)$, i.e

$$m_{(i,j)} \leftarrow \frac{m_{(i,j)}}{P(\rho_l)}$$

At this point there are k_i tables each of the form $P(I_{(i,j)}, R_j | X_j)$, $0 \leq j \leq k_i$, where $I_{(i,j)}$ is the variable that the incoming edge $e_{(i,j)}$ represents in the communication graph (i.e., pre-redundant influence marking), R_j is a sub-set of $\mathcal{R}_i$ such that no variable appears in more than one R_l, $0 \leq l \leq k_i$, and

$X_j \subset \mathcal{R}_i \cup I$. Since this means that each variable exists in one and only one message as a conditioned upon variable (i.e., left side of the $|$), all messages can be safely combined together, and the un-contaminated joint across $\mathcal{R}_i$ can be retrieved and marginalized onto the needed input probabilities.

Returning to the R.O.S.E. example, consider the problem of correcting the redundant influence on the Supervisor node. The two incoming messages correspond to $m_{(4,1)} = P(R_1, S_1)$ and $m_{(4,2)} = P(R_2, S_1)$ which lead to the erroneous probability calculation of equation 11. To remove the influence $\mathcal{R}_4$ is first calculated,

$$\mathcal{R}_4 = \{S_1\}.$$

Letting ρ_i take on each value in $\mathcal{R}_4$ gives us $\rho_1 = S_1$. Step one yields $c_1 = 2$, so step 2 executes parts a and b. Choosing $m_{(4,1)}$ for step 2.a, $P(S_1)$ is retrieved,

$$P(S_1) = \sum_{R_1} P(S_1, R_1),$$

which leads us to condition all the remaining messages with R_1 (i.e., $m_{(4,2)}$) such that

$$m_{(4,2)} \leftarrow P(R_2|S_1) = \frac{m_{(4,2)}}{P(S_1)}$$

leaving us with the two messages $m_{(4,1)}$ and $m_{(4,2)}$ being $P(R_1, S_1)$ and $P(R_2|S_1)$, which can be clearly combined into

$$\begin{aligned} P(R_1, R_2, S_1) &= m_{(4,1)} \times m_{(4,2)} \\ &= P(R_1, S_1) \times P(R_2|S_1). \end{aligned}$$

This can in turn be marginalized to:

$$P(R_1, R_2) = \sum_{S_1} P(R_1, R_2, S_1),$$

from which the desired probability $P(L)$ can be computed as

$$P(L) = \sum_{R_1,R_2} (P(L \mid R_1, R_2) \cdot \sum_{S_1} P(R_1 \mid S_1) P(R_2 \mid S_1) P(S_1)) = \sum_{R_1,R_2} P(L \mid R_1, R_2) P(R_1, R_2)$$

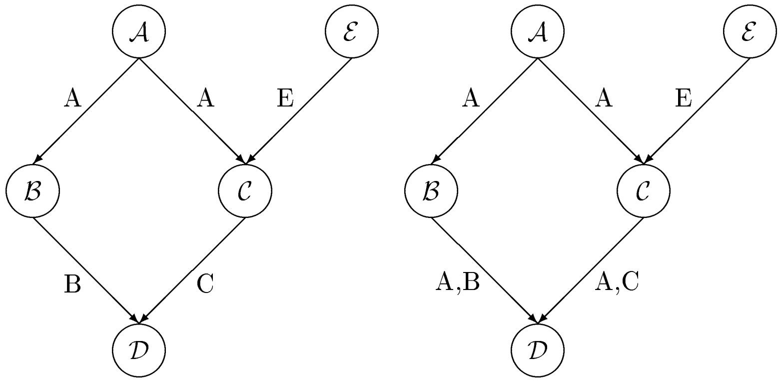
Alternatively, $P(R_1, R_2, S_1)$ can be marginalized to

$$\begin{aligned} P(R_1) &= \sum_{S_1,R_2} P(R_1, R_2, S_1) \\ P(R_2) &= \sum_{S_1,R_1} P(R_1, R_2, S_1) \end{aligned}$$

and these can be used for an approximate answer.

6 Model Solution

An alternative for the removal of the redundant influences exists. This solution involves automatic construction of a secondary local Bayesian Network of



(a) (b)

Figure 6: Another sensor model communication graph (a) and redundancy graph (b).

three layers. These layers are generated based on the labels of the communication graph, and the conditional probability tables for those layers are generated through the passing of messages. These conditional probability tables will themselves occasionally require re-calculation whenever changes occur in the local observations of agents above it in the communication graph. However, when the changes occur only among the shared variables, the messages need only contain the probability tables on the single variable that labeled the edge in the original communication graph. This causes a significant communication cost saving over the passing of joints which grow exponentially with the size of the label.

The theory behind the construction of the multi-layer network is simple. The top layer represents the current values of the input variables I_i to agent i. The middle layer is a conditional probability table for the redundantly expressed variables given the input variables. This layer calculates the value of the redundantly expressed variables given the input parameters. Finally the bottom layer contains the reverse of the middle tables, the probability of the input variables given the redundant variables. These tables are calculated in a naive fashion from messages passed along the edges, and the local propagation algorithm takes care of removing the redundancies. Once the propagation in this model is completed, the resulting belief in the inputs I_i can be fed into the correct location in the agent's actual Bayesian Network.

For use as an example, let us consider the slightly more complex sensor model of Figure 6. This model has two sensors $\mathcal{A}$ and $\mathcal{E}$ each of which produce a single variable A and E respectively. These sensor readings are in turn absorbed by two intermediate supervisors $\mathcal{B}$ and $\mathcal{C}$ who produce reports B and C. These reports are fed into the supervisor node $\mathcal{D}$ that has a fused belief in the location of the target.

Given the communication graph and a redundancy graph on m agents $N = \{n_1,\ldots,n_m\}$, and a set of shared variables $\mathcal{S}$ between these agents, the following three functions are defined:

$$
\begin{aligned}
\Lambda &: \mathcal{S} \rightarrow N \\
\Upsilon &: N \rightarrow S_i, S_i \subseteq \mathcal{S} \\
\Gamma &: N \times \mathcal{S} \rightarrow S_i, S_i \subseteq \mathcal{S} \\
\Omega &: N \rightarrow S_i, S_i \subseteq \mathcal{S}
\end{aligned}
$$

where $\Lambda(s_i)$ is the agent that produces the shared variable s_i, $\Upsilon(n_i)$ is the set of input variables to agent n_i, $\Gamma(n_i, s_j)$ is the set of labels along the marked paths that lead from $\Lambda(s_j)$ to n_i in the redundancy graph, and $\Omega(n_i)$ is the set of redundant influences on the agent n_i.

Looking at the agent and redundancy graphs of Figure 6, consider the following examples of the functions:

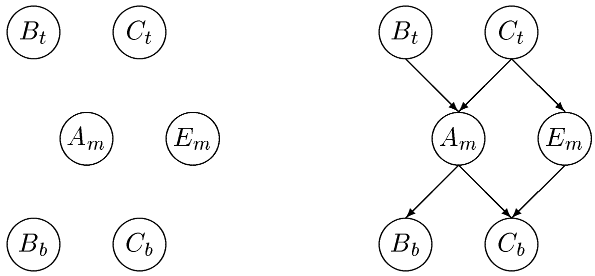
$$
\begin{aligned}
\Lambda(C) &= \mathcal{C} \\
\Upsilon(\mathcal{D}) &= \{B, C\} \\
\Upsilon(\mathcal{A}) &= \phi \\
\Gamma(\mathcal{D}, A) &= \{A, B, C\} \\
\Omega(\mathcal{D}) &= \{A\} \\
\Omega(\mathcal{B}) &= \phi
\end{aligned}
$$

Given Λ, Υ, Γ, and Ω construction of the model involves adding four types of variables (Note: in this notation variables in the communication graph are represented as lower case letters and variables being placed into the new Bayesian Network, that correspond to variables from the communication graph, are labeled with the same lower case letter subscripted with a letter that indicates which level of the new network they exist in). These layers are added in four steps:

1. Add a node v_t, $v \in \mathcal{S}$, to the top layer for each $v \in \Upsilon(n_i)$.
2. Add a node v_m, $v \in \mathcal{S}$, to the middle layer for each $v \in \Omega(n_i)$.
3. Add a node v_m, $v \in \mathcal{S}$, to the middle layer for each $v \in \Upsilon(\Lambda(w))$ for all $w \in \Upsilon(n_i)$.
4. Add a node v_b, $v \in \mathcal{S}$, to the bottom layer for each $v \in \Upsilon(n_i)$.

This fulfills our view of the system as presented at the start of the Section with only nodes from step 3 not being mentioned in the initial description of the system. It turns out that the parents of the input variables are also needed in the middle level so that the correct estimation of the bottom layer can occur. This necessity is most easily shown with an example. If we assume that this is not, so then in the modified R.O.S.E. model of Figure 6 this would imply that the probability of shared variable C can be inferred from a prior on A and a conditional of C given A. This directly contradicts the implications of the communication graph that indicates that both the priors of A and E, as well as the conditional of C given A and E are necessary. In other words failure to include nodes from step 3 implies that the knowledge of one input variable always causes the output parameters to be independent of the remaining input variables. This restriction is not necessary or wanted.

As for why only knowledge of $A(A_m)$ and $E(E_m)$ are all that is necessary to calculate the correct belief in $B(B_b)$ and $C(C_b)$, consider if this were not the case. This would imply that some other variable has an influence on either B or C. By definition of the agent system this variable must be an ancestor of either A or E, and although no such nodes exist in this graph if it did it would still not influence B or C. This stems from the fact that the oracular assumption coupled with the nature of message passing causes the separation of a variable from its ancestors through its parents.



(a) (b)

Figure 7: Nodes in model built for Communication Graph of Figure 6, (a), with edges (b).

Returning to the modified R.O.S.E. model of Figure 6, six total variables are added to the model in the following way.

1. B_t and C_t are added to the top of the model, the input variables to agent D.
2. A_m is added to the middle, as this is a redundant influence on agent D.
3. E_m is added to the middle, as this is an input variable of an input variable through agent C.
4. B_b and C_b are added to the bottom, as these are the input variables to agent D.

When completed the three layer network, without edges, is visible in Figure 7.a.

Once the nodes have been placed in the network, the edges are added in the following three steps.

1. Add an edge from v_t to w_m for any existing w s.t. $w \in \Gamma(\Lambda(v), x)$ and $x \in \Omega(n_i)$.
2. Add an edge from v_t to w_m for each $w \in \Upsilon(\Lambda(v))$.
3. Add an edge from v_m to w_b for each $v \in \Upsilon(\Lambda(w))$.

Using this algorithm, six edges are added to the R.O.S.E. example graph in Figure 7.b.

1. $B_t \rightarrow A_m$ and $C_t \rightarrow A_m$.
2. $C_t \rightarrow E_m$.
3. $A_m \rightarrow B_b$, $A_m \rightarrow C_b$, and $E_m \rightarrow C_b$.

All that remains for this method is a way to build the conditional probability tables necessary for the model. To construct these tables, each edge that is labeled as carrying a redundant influence must carry a message that is the joint probability of not only its label but also the the input variables of the agent at the head of the edge. For example the message passed along the edge $C \rightarrow D$ in Figure 6.b carries a probability table that not only has A, C but also E. Thus to set the probability tables requires a communication cost larger than the regular communication solution cost. However, as stated earlier, once these messages have been passed, there is no need to pass a joint probability table with more than the original labels (see Figure 6.a) of the edges, unless a change occurs that causes some of the dependencies encapsulated within the joint to change. This is the case whenever an agent makes observations about its internal nodes and must update the rest of the agent system to reflect this.

Given that messages of the type described above are being passed, one of four methods is used to assign the conditional probability tables to each of the nodes in the model.

1. The top level nodes v_t take their probability tables from the messages that are received as input. Setting the tables to a uniform distribution for the purposes of place holding is all that is necessary.
2. The middle level nodes tables are built by the following algorithm.
   (a) For some node v_m build a set M of all messages (joint probability tables) that contain the variables v or w where $w_t \in \Pi(v_m)$.
   (b) For each message $m_i \in M$ marginalize away all variables that are not members of T_{v_m} where
   $$T_{v_m} = \{v\} \cup \bigcup_{w_t \in \Pi(v_m)} \{w\}$$
   (c) Derive $P(v)$ by marginalizing one of the messages onto it.
   (d) Divide the remainder of the messages by $P(v)$.
   (e) If any of the other variables in the w appears in the labels of more than one message,
      i. Calculate $P(w)$ by marginalizing one of the original messages onto it.
      ii. Adjust all but one of the messages containing w by $P(w)$.
   (f) Combine all the messages in M to get the conditional table for $P(v, W)$, where $W = \{w_i : w_{i_t} \in \Pi(v_m)\}$.
   (g) Derive $P(W)$ by marginalizing away v_m from one of the messages.
   (h) Set $P(v_m|\Pi(V_M)$ to the combined table divided by $P(W)$.

3. By the nature of the construction there should exist one, and only one, message m_{v_b} which contains the joint probability table across the variables whose conditional probability table is required for a bottom node v_b. To derive the necessary table, simply

   (a) Marginalize v out of the message m_{v_b} to get the joint probability $P(W)$ where $W = \{w_i : w_{i_m} \in \Pi(v_b)\}$.

   (b) Divide the message m_{v_b} by $P(W)$ to get the conditional table $P(v_b|\Pi(v_b))$.

Returning to our example of building the model in Figure 7, two messages enter the agent D. The first is $m_1 = P(A, B)$ while the second is $m_2 = P(A, C, E)$. Step 1 tells us to set the probability tables of each of the top nodes to a uniform distribution so

$$\begin{aligned} P(B_t = b_{t_i}) &= \frac{1}{|B_t|} \quad \forall b_{t_i} \in B_t \\ P(C_t = c_{t_i}) &= \frac{1}{|C_t|} \quad \forall c_{t_i} \in C_t. \end{aligned}$$

The middle nodes are set using the above algorithm. Consider the case of setting the table for $P(A_m|B_t, C_t)$. Both messages contain some of the same variables as the desired table so we get $M = \{m_1, m_2\}$. Removing the variables that are not in the desired table causes

$$\begin{aligned} m_2 &= \sum_E m_2 \\ m_2 &= \sum_E P(A, C, E) \\ m_2 &= P(A, C) \end{aligned}$$

Choosing m_1 to calculate $P(A)$ gives

$$\begin{aligned} P(A) &= \sum_B m_1 \\ P(A) &= \sum_B P(A, B) \end{aligned}$$

which is used to condition message m_2 to give us

$$\begin{aligned} m_2 &= \frac{m_2}{P(A)} \\ m_2 &= \frac{P(A, C)}{P(A)} \\ P(C|A) &= \frac{P(A, C)}{P(A)}. \end{aligned}$$

These two messages m_1 and m_2 are now combined to yield

$$m = m_1 \times m_2$$

$$\begin{aligned} m &= P(A,B) \times P(C|A) \\ P(A,B,C) &= P(A,B) \times P(C|A) \end{aligned}$$

which in turn has A marginalized away to give

$$P(B,C) = \sum_A P(A,B,C).$$

Finally the actual table is computed as

$$\begin{aligned} T &= \frac{P(A,B,C)}{P(B,C)} \\ P(A|B,C) &= \frac{P(A,B,C)}{P(B,C)} \\ P(A_m|B_t,C_t) &= \frac{P(A,B,C)}{P(B,C)}. \end{aligned}$$

The table for $P(E_m|C_t)$ is computed in a similar manner. The only tables remaining to be computed are the tables for the bottom layer. Consider the computation of the table $P(B_b|A_m)$ this is simply computed using message 1 as follows:

$$\begin{aligned} P(B_b|A_m) &= \frac{m_1}{P(A)} \\ P(B_b|A_m) &= \frac{P(A,B)}{\sum_B m_1} \\ P(B_b|A_m) &= \frac{P(A,B)}{\sum_B P(A,B)} \end{aligned}$$

7 Comparison of Model and Communication Approaches

This Section will compare the cost of the communication solution (Section 5) and the model solution (Section 6) and calculate a metric by which one can be chosen over the other. The analysis begins with the definition of several communication and redundancy graph dependent terms.

1. n – the number of agents in the communication graph.
2. m – the number of edges in the communication graph.
3. s_i, $1 \leq i \leq m$ – the number of variables along an edge in the redundancy marked version of the communication graph.
4. s_m – the largest label in the communication graph i.e.,

$$s_m = \max_{1 \leq i \leq m} \{s_i\}.$$

5. b – the number of states in the largest shared variable.

6. d_i – the maximum in-degree of any agent in the communication graph.

Given these constant parameters, a comparison of the relative communication costs can be undertaken. In this comparison no attention will be paid to the cost of the additional local computation involved to correct the redundancies, since the local cost of correction for either the communication or model based approach has negligible impact on the total agent system propagation time for two reasons. Firstly, it is most often the case that communication is slower than local computation, with the speed difference usually in the order of several thousand times.

Secondly, both methods, communication and model based, use about the same amount of local computation in the correction of the redundant information. Obviously, the communication solution requires the formulation of a joint probability table with b^{s_m} entries and that requires $O(b^{s_m})$ operations to build. Furthermore, it can be shown (cf. [BV98]) that the propagation algorithm for the model solution will also require, and be dominated by, a probability table of b^{s_m} entries that will be used to set the probabilities of the incoming shared variables. This leads to the fact that the entire propagation cost of this local model is also $O(b^{s_m})$.

The comparison of the two costs begins with the analysis of the communication solution in Section 7.1. This will be followed by the analysis of the model cost solution in Section 7.2 and a comparison of the two in Section 7.3.

7.1 Cost of Communication Solution

The communication solutions cost is easily evaluated. It requires one probabilistic message to be sent along each edge in the graph. Each of these messages contains a table which has a single floating point number for each entry, with the number of entries in the table being equivalent to the size of the cross-product of the variables on the edge label. Thus using the transmission of a single floating point number as our base unit of communication cost we have

$$\sum_{i=1}^{m} b^{s_i} \tag{13}$$

as the upper bound on the total cost of communication. This is clearly bounded above by

$$\sum_{i=1}^{m} b^{s_m} \tag{14}$$

since s_m dominates s_i for $1 \leq i \leq m$.

Now since the term being summed is constant with respect to the index, the total computation cost for the model solution is

$$mb^{s_m}. \tag{15}$$

7.2 Cost of Model Solution

The cost of the model solution is a little tougher to formulate. Using again as our basis for calculation the transmission of a single floating point number the normal communication cost of the algorithm is

$$mb, \tag{16}$$

which corresponds to each edge carrying a single table on the one variable that labels it in the communication graph.

However, from time to time, the conditional probability tables of the local propagation models must be updated using a much larger communication load. This enlarged propagation occurs whenever the distribution of one of the agents is modified by a local observation. In the worst case this process will require all of the tables in all of the models of the network to be re-calculated. The average case is of course much better in that only local models that contain the shared variable whose producing agent has been modified will need to be updated.

Once again the cost of this communication is the cost of the transmission of an enlarged probability table. In this case the table is defined not only on the variables that label the edge in the redundancy graph but also on the parents of the node that originate the edge. Allowing l_i to be a variable that labels the edge $e_i(v_i^+, v_i^-)$ in the communication graph, letting $\Pi(v)$ be the set of parents of v, and defining the function $\Xi(n_i, s_i)$ ($\Xi : N \times \mathcal{S} \rightarrow S_i$, $S_i \in \mathcal{S}$) as the label, from the redundancy graph, of the edge that is labeled by s_i and enters n_i in the communication graph, the cost is

$$\sum_{i=1}^{m} b^{|\Xi(v_i^-, l_i) \cup \Pi(v_i^+)|}. \tag{17}$$

Since the set produced by the union of the label and the grand-parent set is no smaller than the sum of the elements in each individual set, i.e.,

$$|\Xi(v_i^-, l_i) \cup \Pi(v_i^+)| \leq |\Xi(v_i^-, l_i)| + |\Pi(v_i^+)| \leq s_m + d_i,$$

formula 17 can be re-written as

$$\sum_{i=1}^{m} b^{s_m + d_i} \tag{18}$$

which since s_m and d_i are constant with respect to the summation is

$$mb^{s_m + d_i}. \tag{19}$$

To get the true cost of the model solution for the general case, it is assumed that some probability, p, exists which indicates the chance of an update being caused by an internal change in a table, and therefore requiring the larger communication cost, versus a propagation. Since there are only two possibilities (i.e., full update and regular propagation), the probability of the regular propagation occurring is $1 - p$. Taking a weighted sum of the two costs (formulas 16 and 19) yields a total cost of

$$(1-p)\left[md\right] + p\left[mb^{s_m + d_i}\right] \tag{20}$$

7.3 Comparison

Setting the two costs (formulas 15 and 20) equal yields

$$(1-p)\,[mb] + p\left[mb^{s_m+d_i}\right] = mb^{s_m}. \tag{21}$$

Dividing equation 21 by m and expanding the first term on the left hand side gives

$$b - pb - pb^{s_m+d_i} = b^{s_m}. \tag{22}$$

Solving equation 22 for p yields

$$p = \frac{b^{s_m} - b}{b^{s_m+d_i} - b}. \tag{23}$$

Thus a purely graph dependent function is available that gives us the break even value for p. If the frequency of large conditional probability table updates is greater than the above formula for p, then the communication solution is time superior. Otherwise the model solution is better with respect to time. This result is perhaps not surprising in that consideration of the asymptotically dominant terms yields

$$p = \frac{b^{s_m}}{b^{s_m+d_i}} = \frac{1}{b^{d_i}}.$$

which shows us that, as the graph parameters increase, the value of p, the probability of an update, for which the two solutions break even decrease exponentially with respect to the indegree. This is to be expected from the formula 20 since the full update communication cost is so much greater than the regular propagation communication costs for large values of the graph parameters.

8 Conclusions and Future Work

We addressed the rumor problem in probabilistic agent systems that use Bayesian networks to represent their view of the world, focusing on the Agent Encapsulated Bayesian Network (AEBN) model. We reviewed related work on probabilistic multiagent systems for interpretation and fusion of information. We introduced the notion of communication graph, within which the rumor problem may be clearly appreciated.

We presented an algorithm to identify redundant influence in communication graphs and two algorithms to compensating these influences, the communication solution and the model solution. The trade-off between the solutions was investigated, leading to a simple formula (equation (7.3) that relates the probability of update based on local variables with simple graph parameters. These algorithms allow the designer of an agent encapsulated Bayesian network to design it locally and delay the consideration of rumors to a later stage, thus simplifying the design task. Moreover, rumors are compensated in such a way that agents do not need to be merged, again simplifying the design task.

Further work should also include the careful software implementation of the algorithms presented in this paper.

Acknowledgments

This work has been supported in part by the U.S. Office of Naval Research through project "Dynamic Decision Support for Command Control, and Communication in the Context of Tactical Defense" (BA97-006 and Grant Number N00014-97-1-0806) and by the U.S. Department of Defense through the DARPA Autonomous Negotiating Teams (ANTS) project (AO Number H359 and Contract Number F30602-99-2-0513). Useful discussion with the original members of the DAG team at the University of South Carolina, and especially John Rose, Young-Gyun Kim, Miguel Barrientos, Clif Presser, and Wayne Smith, is gratefully acknowledged. M.V. thanks Jiří Vomlel for conversations on the topic of this paper, and especially on the issue of dependence among input variables. M.V. thanks in a special way Finn V. Jensen and his Decision Support Systems Group in the department of Computer Science at the University of Aalborg for providing the quiet and productive research environment that allowed this paper to be completed during a sabbatical visit in May 2000.

References

[Blo98] Mark Bloemeke. *Agent Encapsulated Bayesian Networks.* PhD thesis, Department of Computer Science, University of South Carolina, 1998.

[BV] Mark Bloemeke and Marco Valtorta. The factor tree for probability update and several extensions. Under review.

[BV98] Mark Bloemeke and Marco Valtorta. A hybrid algorithm to compute marginal and joint beliefs in Bayesian networks and its complexity. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 208–214, Madison, WI, July 1998.

[Har90] A.C. Harvey. *Forecasting, structural time series models and the Kalman Filter.* Cambridge University Press, New York, NY, 1990.

[Jen95] Finn V. Jensen. *An Introduction to Bayesian Networks.* Springer-Verlag, New York, NY, 1995.

[JU97] S.J. Julier and J.K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *The American Control Conference*, pages 27–31, San Francisco, California, 1997. Morgan Kaufmann Publishers. Also available on line at http://ait.nrl.navy.mil/people/uhlmann/CovInt.html.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, pages 35–46, 1960.

[Koz92] Dexter C. Kozen. *The Design and Analysis of Algorithms.* Springer-Verlag, New York, 1992.

[LMW01] Kathryn Blackmond Laskey, Suzanne M. Mahoney, and Ed Wright. Hypothesis management in situation-specific network construction. In *Proceedings of the Seventeenth Annual Conference on Uncertainty* *in Artificial Intelligence (UAI-01)*, pages 301–309, Seattle, WA, August 2001.

[May79] Peter S. Maybeck. *Stochastic Models, Estimation, and Control, Volume 1*. Academic Press, New York, NY, 1979. The first chapter is available at www.cs.unc.edu/~welch/Kalman/index.html.

[Mot95] L. Motyckova. Maximum flow problem in distributed environment. In *SOFSEM '95: Theory and Practice of Informatics*, pages 425–430, Berlin, 1995. Springer-Verlag.

[VKV02] Marco Valtorta, Young-Gyun Kim, and Jiří Vomlel. Soft evidential update for multiagent systems. *International Journal of Approximate Reasoning*, 29(1):71–106, January 2002.

[Xia96] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, 86:295–342, 1996.

[XL00] Y. Xiang and V. Lesser. Justifying multiply sectioned Bayesian networks. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, pages 349–356, Boston, MA, July 2000.