Agent-Encapsulated Bayesian Networks and the Rumor Problem

Scott Langevin
Department of Computer Science and Engineering
University of South Carolina
langevin@cec.sc.edu

Marco Valtorta
Department of Computer Science and Engineering
University of South Carolina
mgv@cec.sc.edu

Mark Bloemeke
LogicBlox, Inc.
Atlanta, GA
mark.bloemeke@logicblox.com

ABSTRACT

We present a multiagent organization for data interpretation and fusion in which each agent uses an encapsulated Bayesian network for knowledge representation, and agents communicate by exchanging beliefs (marginal posterior probabilities) on shared variables. We call this organization an Agent-Encapsulated Bayesian Network (AEBN) system. Communication of probabilities among agents leads to rumors, i.e. potential double counting of information. We show how to compensate for rumors in AEBN systems by passing extended messages that contain joint probabilities. We analyze the complexity of the proposed solution using simple parameters of the probabilistic multiagent system.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Uncertainty, "fuzzy," and probabilistic reasoning*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Intelligent agents, Multiagent systems*

General Terms

Algorithms, Design, Theory

Keywords

Communication protocols, Distributed problem solving, Knowledge representation, Reasoning (single and multi-agent)

1. INTRODUCTION

It is well known that communication of probabilities among agents leads to potential double counting of information. This fundamental problem is due to mishandling of dependent or correlated variables and is known as the rumor problem [13].

We first present a method to link multiple, individually designed, Bayesian networks together to form a multiagent system. Each agent in the system will use a probabilistic model as its internal model of the world and communicates with other agents in the system by passing messages. The messages sent and received are distributions on variables of interest that are shared between the agents. We then describe the rumor problem in the context of our chosen agent model. Finally, we present a solution to the rumor problem for our chosen agent model that compensates for double counting of information by extending agent communication. We show that the solution is efficient.

Cite as: Agent-Encapsulated Bayesian Networks and the Rumor Problem, Author1, Author2 and Author3, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

2. AGENT ENCAPSULATED BN SYSTEMS

An Agent Encapsulated Bayesian Network (AEBN) [1, 9] is an agent that utilizes a Bayesian network for its internal representation of the world. How the agent utilizes this representation in decision support or goal based planning is unimportant so long as the world view is updated based only on local observations and observations received in the form of probabilistic messages from communicating agents.

In an AEBN system the agents communicate through the transmission of probability distributions on shared variables. The topology of the communication in the multiagent system forms a DAG structure. The Bayesian network of each agent can be divided into three distinct sets of variables: I, those about which other agents have better knowledge; L, those that are used only within the agent; and O, those that this agent has the best knowledge of and that other agents may want. 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 AEBN system 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 beliefs 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 the knowledge with one or more agents who wish to integrate the 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.

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 mediating 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 unique label.

We will assume that *observation variables* have been introduced as needed and that an agent's probability distribution upon receipt of messages from other agents, using the approach of soft evidential update [12, 2]. Therefore, each agent that receives messages from other agents obtains soft evidence for one or more observation variables. To update an agent's distribution $P(V)$ with new evidence $Q(E_1, E_2, ..., E_n)$ for some set of variables $\{E_1, E_2, ..., E_n\} = I$ one calculates the joint probability $P(V)$, dividing by the marginal probability $P(I)$, and multiplying it by the new distribution of $\{E_1, E_2, ..., E_n\}$, this corresponds to the application of Jeffrey's rule (also known as the rule of probability kinematics),

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

thus obtaining:

$$Q(V) = P(V \backslash I|I) \cdot Q(I) = \frac{P(V)}{P(I)} \cdot Q(I). \tag{2}$$

In the case in which the input variables are not independent in the receiving agent, Equation 1 does not hold. (See [12, Section 5] for a detailed discussion on this point.) Lemma 1 in [12] allows the replacement of Equation 2 by:

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

where Q^*_I is the I_1-projection of probability distribution P on the set of all distributions defined on I and having $Q(E_i), i = 1, ..., n$, as their marginals. In practice, $P(V)$ could be updated to $Q^*(V)$ using the *big clique algorithm* of [12, 5], *lazy big clique algorithm* of [7], or the *wrapper methods* of [10].

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 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 not affected by their descendants. 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. In order to provide a formal definition of "preceding agent," we introduce the notion of communication graph in Section 3.

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. 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 is possible for more than one edge to exist between two nodes, though each will be uniquely labeled.

We can now formalize the constraint that, in the agent system, all variables that are parents are not affected by 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 if there is no directed path in the communication graph from A_j to A_i, any changes (whether by observation or by intervention) in the state of the variables in W_j does not affect the state of the variables in W_i. This is a very strong condition on the distribution of the variables in different agents of the agent system. This is *not* a symmetric relation, and therefore cannot be represented by any independence relation, since every independence relation is symmetric. There is an analogy to be made with casual Bayesian networks [11]. In a causal Bayesian network, when a variable is set (by external intervention), the parents of that variable are disconnected from it; more precisely, the result of the intervention is to create a new Bayesian network in which we remove the edges incoming into a variable that is set. The analogy, however, is not complete. In a causal Bayesian network, when a variable is set by intervention, some of the parent variables may be affected through backdoor paths, as explained in [11, section 3.3]. In an AEBN, there is no possibility for a variable in an agent to be affected by a descendent agent.

Consider as an example a four-agent system, where a supervisor agent fuses reports from two observer agents, each of which reports information from a single sensor agent. The communication graph shown in Figure 1 is constructed by first identifying shared variables (S, L_1, and L_2), then directing labeled edges from the producing agents to the consuming agents. The labels for the edges correspond to the shared variable. In this example, the edges directed from the *Sensor* agent to the $Observer_1$ and $Observer_2$ agents are labeled with S, and the edges from $Observer_1$ and $Observer_2$ to the *Supervisor* agent are labeled with L_1 and L_2, respectively. Henceforth this example will be referred to as the Redundantly Observed Sensor Example (ROSE).

4. REDUNDANT INFLUENCES

It is within the communication graph that the nature of the rumor problem can be clearly understood. Using the ROSE communication graph as an example, we can see the problem centers on the fact that the supervisor agents view of the world, held in its Bayesian network, is doubly influenced by the initial sensor reading. The supervisor computes its belief in L, the location of the target, as

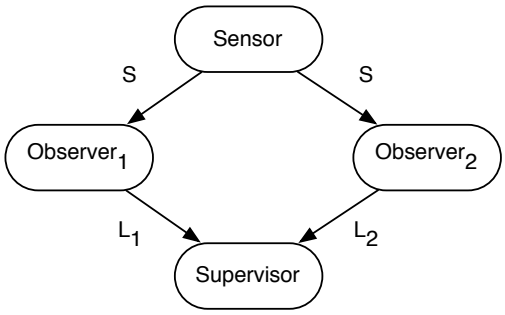

Figure 1: Redundantly Observed Sensor Example (ROSE) communication graph.

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

which expands to

$$P(L) = \sum_{L_1,L_2} P(L|L_1,L_2)m(Observer_1)m(Observer_2) \tag{5}$$

where $m(Observer_1)$ and $m(Observer_2)$ are messages the supervisor agent receives from Observer_1 and Observer_2. Expanding $m(Observer_1)$ yields

$$m(Observer_1) = \sum_{S} P(L_1|S)m(Sensor) \tag{6}$$

with $m(Observer_2)$ being calculated as

$$m(Observer_2) = \sum_{S} P(L_2|S)m(Sensor) \tag{7}$$

Finally, expansion of $m(Sensor)$ yields

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

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

$$P(L) = \sum_{L_1,L_2} P(L|L_1,L_2) \sum_{S} P(L_1|S)m(Sensor)\times \sum_{S} P(L_2|S)m(Sensor) \tag{9}$$

Substituting equation 8 into 9 yields

$$P(L) = \sum_{L_1,L_2} P(L|L_1,L_2) \sum_{S} P(L_1|S)P(S) \sum_{S} P(L_2|S)P(S) \tag{10}$$

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

$$P(L) = \sum_{L_1,L_2,S} P(L|L_1,L_2)P(L_1|S)P(S)P(L_2|S)P(S) \tag{11}$$

In equation 11, $P(S)$ is redundantly incorporated in the supervisor agent, resulting in a redundantly influenced calculation of $P(L)$, because the correct expression of $P(L)$ (calculated using the chain rule) is

$$P(L) = \sum_{L_1,L_2} P(L|L_1,L_2)P(L_1,L_2) \tag{12}$$

Since there exists no directed path between L_1 and L_2 in the communication graph, neither affects the other in the ROSE AEBN system.

$$P(L_1,L_2) = \sum_{S} P(L_1|S)P(L_2|S)P(S) \tag{13}$$

Substitution of equation 13 into equation 12 leaves the correct equation for calculating $P(L)$

$$P(L) = \sum_{L_1,L_2,S} P(L|L_1,L_2)P(L_1|S)P(L_2|S)P(S) \tag{14}$$

where equation 14 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 agents.

Redundant influences arise in a communication graph whenever the combination of messages received by an agent causes the belief in some shared variable to be over included.

The principal objective of this paper is to allow the handling of the rumor problem in an automated fashion. This will be achieved using algorithms that first identify redundant influences using the communication graph (Section 4.1) and then using a communication based solution (Section 5) to eliminate them.

4.1 Identifying Redundant Influences

This section describes a method for identifying redundant influences in a communication graph. *V_i redundantly influences V_j* if there exists multiple node-disjoint paths from V_i to V_j. There is a *redundant influence* between nodes V_i and V_j in some communication graph G if either V_i redundantly influences V_j or V_j redundantly influences V_i

Redundant influences are external to an agent and are dependent on the communication graph topology, therefore, it cannot be assumed they will be known in advance when the agents internal model is designed. Hence, the redundant influences are orthogonal to the dependencies that are encoded in the Bayesian network that is contained within an agent and it is appropriate to support the processing of redundant influences separately from the construction of individual agents.

We claim, given an AEBN communication graph $G = (V, E)$ with nodes $V_i, V_j \in V$, where V_i is an ancestor of V_j, it is sufficient to identify all node-disjoint paths from V_i to V_j in order to see all routes of redundant influence from V_i to V_j. Following is a skeleton of a proof of our claim.

First we characterize thoroughly the routes through which redundant influences arrive at a node in the communication graph.

Assume that we have a communication graph G, as defined above, such that between V_i and V_j there are only n node disjoint paths $\{V_i \to V_{11} \to ... \to V_{1k_1} \to V_j, ..., V_i \to V_{n1} \to ... \to V_{nk_n} \to V_j\}$, where k_i, $1 \leq i \leq n$ is the length

of path i minus the endpoints (Figure 2). Further, assume we have more than n redundant influences. Clearly, since redundant influences must occur along some series of paths, there must be some paths that are not node-disjoint between V_i and V_j causing the additional redundancies. Each of these additional paths must take on one of four forms (assume $p, q \in [1, n]$):

1. It starts at one node along node-disjoint path q and ends at a different node along node-disjoint path p. (Figure 3)
2. It starts at one node along path q and ends at node V_j. (Figure 4)
3. It starts at node V_i and ends at a node along node-disjoint path q. (Figure 5)
4. It starts and ends along node-disjoint path q. (Figure 6)

In all of these cases, the additional redundant influences are due to a subgraph that does not include both V_i and V_j. The subgraph effectively amplifies the redundancy between V_i and V_j, but the cause is redundant influences in the subgraph caused by multiple node disjoint paths between two nodes in the subgraph. If we were to compensate for these additional redundancies in the subgraph then we are left with the redundant influences from the n node disjoint paths between V_i and V_j. In other words, we can recursively remove redundant influences in subgraphs between V_i and V_j and be left with n redundant influences corresponding to the n node disjoint paths. The remaining n redundancies could be compensated for similarly. This is a recursive argument and means we can identify all redundant influences by examining all pairwise node disjoint paths in the graph. The *Create Redundancy Graph* algorithm, described below, can be used to detect and label node disjoint paths in a communication graph. Once this algorithm has been run, a new graph, known as the *redundancy graph*, is constructed. The redundancy graph has the same nodes and edges as the communication graph, but its edge labels are expanded if and only if there are redundant influences. This graph will be used, along with the original communication graph, to compensate for redundant influences in the communication solution described in Section 5.

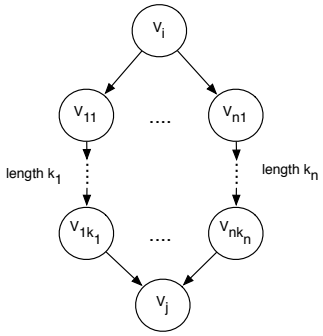

Figure 2: n node disjoint paths between V_i and V_j.

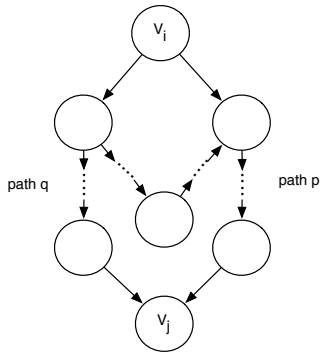

Figure 3: Case 1, extra redundant influences that are not node-disjoint.

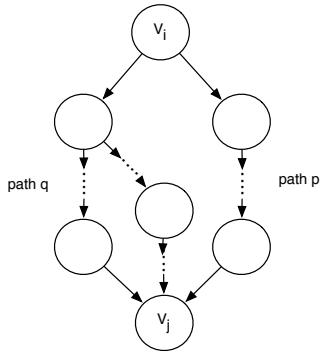

Figure 4: Case 2, extra redundant influences that are not node-disjoint.

Create Redundancy Graph Algorithm:

Start by creating a copy of the communication graph that will serve as the redundancy graph, and whose labels will be expanded 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 communication graph for use in the maximum-flow problem [3].
2. Modify the graph by replacing each node v that has multiple incoming edges with two nodes v_1, v_2. Replace each incoming edge to v, $< x, v >$, with a new edge $< x, v_1 >$, and each outgoing edge from v, $< v, y >$, with a new edge $< v_2, y >$. Finally, create a directed edge $< v_1, v_2 >$[1].
3. Designate v_s as the source for the flow problem and v_t as the sink for the flow problem.
4. Set the maximum flow of each edge to 1.
5. Set the maximum flow to 0 for all outgoing edges from v_s that are labeled with a variable other than s_i in the communication graph.
6. Run the maximum-flow problem.

[1]Without this step, the algorithm would find edge-disjoint paths instead of node-disjoint paths.

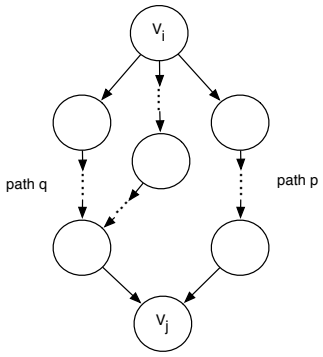

Figure 5: Case 3, extra redundant influences that are not node-disjoint.

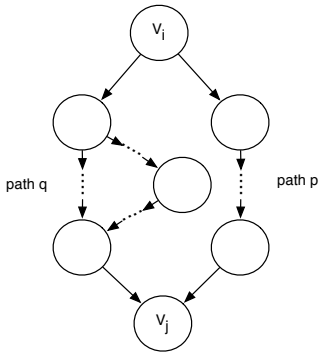

Figure 6: Case 4, extra redundant influences that are not node-disjoint.

7. 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.

8. For each edge in the maximum-flow problem solution that has flow greater than zero (and therefore 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 maximum-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 [6, 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 worst 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 [8] 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.

Considering the ROSE example of Figure 1, the *Create Redundancy Graph* algorithm produces the redundancy graph of Figure 7. 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 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 added to their edge label.

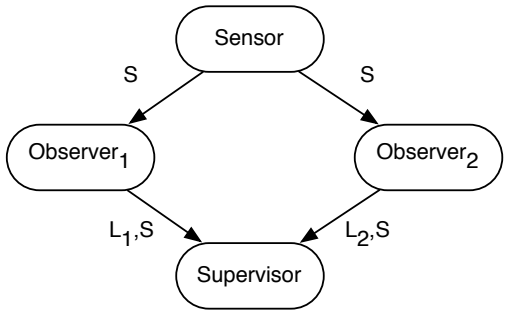

Figure 7: The ROSE redundancy graph.

5. COMMUNICATION SOLUTION

This section proposes a method of compensating for redundant influences where agent communication has been expanded to pass joint probabilities along the appropriately labeled links in the redundancy graph, without any change in the local Bayesian networks of each agent. Computing the expanded messages requires the probability update algorithm used by the agents to be flexible enough to allow the calculation of joints involving fixed input and output variables.

The calculation of joint probabilities is not trivial, especially in the presence of soft evidence. The soft evidential update algorithms such as the big clique [12] and lazy big clique [7] are based on the junction tree method, and were designed 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 [4, Section 5.1]. Additionally, the wrapper methods [10] can be used similarly with a junction tree algorithm, or with a direct query-based algorithm such as bucket elimination.

Care must be taken when removing redundant influences to ensure all redundant influences are correctly compensated for. This is done by ordering the removal of the redundant influences. Consider an example where an agent a_i receives three messages from neighboring agents: $\phi_1(A, B, C)$, $\phi_2(A, B, D)$, and $\phi_3(A, E)$. Agent a_i subscribes to the in-

put variables C, D and E, hence its local Bayesian network calculates $P(O|C,D,E)$ and therefore needs to calculate the joint probability, $P(C,D,E)$, from the received messages. The three messages contain redundant influences: $\phi_1(A,B,C)$ and $\phi_2(A,B,D)$ share redundant influences on variables A and B, and $\phi_3(A,E)$ shares a redundant influence with $\phi_1(A,B,C)$ and $\phi_2(A,B,D)$ on variable A (Figure 8). If we first remove the redundant influence common to all, A, we can divide two of the messages by P(A) and have:

$$\phi_1'(B,C|A) = \frac{\phi_1(A,B,C)}{\sum_E \phi_3(A,E)} = \frac{\phi_1(A,B,C)}{\phi(A)}$$

$$\phi_2'(B,D|A) = \frac{\phi_2(A,B,D)}{\sum_E \phi_3(A,E)} = \frac{\phi_1(A,B,D)}{\phi(A)}$$

Next, we eliminate the remaining redundant influence, B, that is common to $\phi_1'(B,C|A)$ and $\phi_2'(B,D|A)$, by dividing one of the messages by P(B):

$$\phi_1''(C|A,B) = \frac{\phi_1'(B,C|A)}{\sum_{A,D} \phi_2(A,B,D)} = \frac{\phi_1'(B,C|A)}{\phi(B)} = \frac{\phi_1(A,B,C)}{\phi(A)\phi(B)}$$

The correct joint probability is retrieved by combining the updated messages, $\phi_1''(C|A,B)$, $\phi_2'(B,D|A)$, and $\phi_3(A,E)$,

$$\phi(A,B,C,D,E) = \phi_1''(C|A,B)\phi_2'(B,D|A)\phi_3(A,E)$$

Finally, the required joint probability of the inputs is calculated by marginalizing,

$$P(C,D,E) = \sum_{A,B} \phi(A,B,C,D,E)$$

This is correct, if the shared redundant influence $\phi(A,B) = \phi(A)\phi(B)$, which means A and B are independent. However, this is not correct in general and therefore removal of the redundant influences in this order is incorrect. See Figure 9 and 10 for an example of different dependence characteristics of multiple redundant influences based on the topology of the redundancy graph. The removal of redundant influences must be ordered so as to not lose any dependence relations among the redundant influences. In this example, this can be achieved by first removing the redundant influence (A,B) from $\phi_1(A,B,C)$, and then removing the redundant influence A from $\phi_3(A,E)$. This can be restated, more generally, as removing the largest shared redundant influences first, followed by the second largest, and so on. Thus, the redundant influences can be ordered by decreasing size and removed in this order from the received messages. A formal description of this procedure is now presented.

For each agent $a_i (1 \leq i \leq n)$ in the communication graph, there are $k_i (0 \leq k_i \leq n-1)$ incoming edges, $e_{(i,j)} (0 \leq j \leq k_i)$. Each of these edges carries a message $\phi_{(i,j)}$ that is a joint probability table over the variables that make up the corresponding label in the redundancy graph. Start by defining $\mathcal{R}_i$ to be the set of all messages received by agent a_i that contain a redundant influence (i.e., messages that contain any variables in common with another message). More formally $\mathcal{R}_i$ is

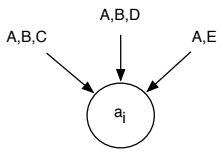

Figure 8: Multiple overlapping redundant influences.

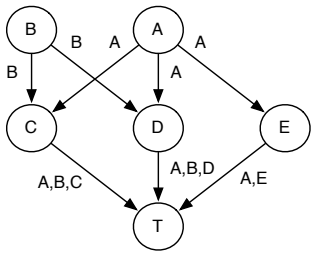

Figure 9: Agent T has redundant influences A and B that are independent.

$$\mathcal{R}_i = \bigcup_{0 \leq j \leq k_i} \phi_{(j,i)} | dom(\phi_{(j,i)}) \cap dom(\phi_{(l,i)}) \neq \emptyset, 0 \leq l \leq k_i, l \neq j$$

To compensate for redundant influences received in messages at agent a_i the procedure *Remove Redundant Influences* should be invoked by a_i before absorbing messages into its local Bayesian network. This procedure will preprocess the messages and remove any redundant influences. The resulting *redundancy-free* messages are stored in the nodes of the redundancy filter tree and can safely be combined together with the messages containing no redundant influences and marginalized to the needed input probabilities. Finally, the resulting joint of the inputs are then absorbed into the local Bayesian network using any probabilistic update method that supports soft evidential update.

Create Redundancy Filter Tree Algorithm:

Create a fully connected graph where the nodes of the graph have a one to one correspondence with the messages in $\mathcal{R}_i$. Associate with each node the corresponding message. The weight of an edge between a node n_i and n_j is $|dom(\phi_i) \cap dom(\phi_j)|$, where ϕ_i is the message associated with node n_i and ϕ_j is the message associated with node n_j. Find a maximum spanning tree for this graph and set the root of the tree to a node with the largest domain. We call this tree the *redundancy filter tree*.

We note that the redundancy filter tree only needs to be constructed once. If new messages arrive, the new messages need to be associated with the corresponding nodes of the redundancy filter tree before the *Remove Redundant Influences* procedure is invoked. This can be done in $O(n)$ if a mapping of message source to redundancy tree node is maintained, where n is the number of messages that contain redundant influences.

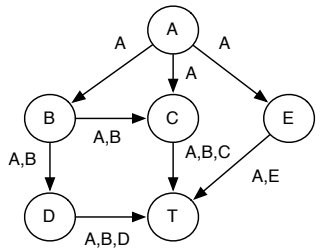

Figure 10: Agent T has redundant influences A and B, where B is dependent on A.

Remove Redundant Influences Algorithm:
The set of messages Φ received by agent a_i is split into two disjoint sets Φ_R and Φ_S, where Φ_R contains all messages containing redundant influences and Φ_S are the remaining messages $\Phi_S = \{\Phi \backslash \Phi_R\}$. Since Φ_S does not contain any redundant influences, it does not require any pre-processing before absorption into agent a_i's local Bayesian network. To remove the redundant influences in Φ_R, construct a *redundancy filter tree* using procedure *Create Redundancy Filter Tree*. Redundant influences are removed starting from the root of the tree and passing messages down the tree towards the leaves. This is achieved by the root node invoking *Remove Redundancy* on each of its neighbors. After all messages have been processed, the set Φ_R^* of *redundancy-free* messages are retrieved from each node in the redundancy filter tree and combined with Φ_S. Finally, the appropriate marginal of the resulting joint can be absorbed into the local Bayesian network.

Remove Redundancy Algorithm:
Let n_i and n_j be two adjacent nodes in the redundancy filter tree and ϕ_i and ϕ_j be their respective redundantly influenced messages. If n_i invokes *Remove Redundancy* on n_j, then:

1. Let $\mathcal{D}_i = dom(\phi_i)$ and $\mathcal{D}_j = dom(\phi_j)$
2. Set the message $\phi_{\mathcal{D}_i \cap \mathcal{D}_j}$ from n_i to n_j to
$$\phi_{\mathcal{D}_i \cap \mathcal{D}_j} = \sum_{\mathcal{D}_j \backslash \mathcal{D}_i \cap \mathcal{D}_j} \phi_i$$
3. n_j invokes *Remove Redundancy* on all adjacent nodes except n_i
4. n_j absorbs the message $\phi_{\mathcal{D}_i \cap \mathcal{D}_j}$ by updating ϕ_j:
$$\phi_j = \frac{\phi_j}{\phi_{\mathcal{D}_i \cap \mathcal{D}_j}}$$

Returning to the ROSE example, consider the problem of correcting the redundant influence on the Supervisor agent. The two incoming messages correspond to $m(Observer_1) = \phi(L_1, S)$ and $m(Observer_2) = \phi(L_2, S)$ which lead to the double counting of $P(S)$. To remove the redundant influence, Φ_R and Φ_S are calculated as,

$$\Phi_R = \{\phi(L_1, S), \phi(L_2, S)\}, \Phi_S = \emptyset$$

From Φ_R, the redundancy filter tree (shown in Figure 11) is constructed. Set the root of the tree to the node corresponding to the message $\phi(L_1, S)$, and then invoke *Remove Redundancy* on the roots neighboring nodes. The result of message passing in the tree is shown in Figure 12. The set of *redundancy-free* messages, Φ_R^*, are retrieved from each node in the tree,

$$\Phi_R^* = \{\phi(L_1, S), \phi(L_2|S)\}$$

Combining Φ_R^* and Φ_S gives

$$\begin{aligned} P(L_1, L_2, S) &= \prod \Phi_R^* \prod \Phi_S \\ &= \phi(L_1, S)\phi(L_2|S) \end{aligned}$$

This can in turn be marginalized to

$$P(L_1, L_2) = \sum_S P(L_1, L_2, S)$$

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

$$\begin{aligned} P(L) &= \sum_{L_1, L_2} P(L|L_1, L_2) \sum_S P(L_1|S)P(L_2|S)P(S) \\ &= \sum_{L_1, L_2} P(L|L_1, L_2)P(L_1, L_2) \end{aligned}$$

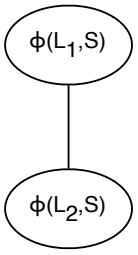

Figure 11: Redundancy filter tree for the Supervisor agent in the ROSE example.

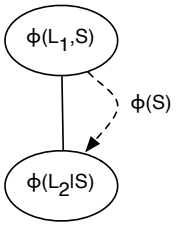

Figure 12: Message passing in the Supervisor agent redundancy filter tree. The node labels reflect the updated agent messages.

5.1 Cost of Communication Solution

In order to discuss the cost of the communication solution, we introduce the following communication and redundancy graph dependent terms:

n - the number of agents in the communication graph.

m - the number of edges in the communication graph.

$s_i, 1 \leq i \leq m$ - the number of variables along an edge in the redundancy graph.

s_{max} - the largest label in the redundancy graph, i.e.,

$$s_{max} = \max_{1 \leq i \leq m} s_i$$

b - the number of states in the largest shared variable

From these defined constants, the communication solution cost is easily evaluated. It requires one probabilistic message to be sent along each edge in the communication 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 that are in the label of the corresponding edge in the redundancy graph. Thus, using the transmission of a single floating point number as our base unit of communication cost, we have,

$$\sum_{i=1}^{m} b^{s_i} \tag{15}$$

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

$$\sum_{i=1}^{m} b^{s_{max}} \tag{16}$$

since s_{max} dominates s_i for $1 \leq i \leq m$.

Simplifying equation 16, the total communication cost for the communication solution is,

$$mb^{s_{max}}. \tag{17}$$

The time complexity of the communication solution can be simplified by ignoring the cost of the creation of the redundancy filter tree since it is the complexity of building a maximal spanning tree of a graph of at most $O(n^2)$, and it is done only once. The cost of the *Remove Redundant Influences* is $O(mb^{s_{max}})$, since there are at most m nodes in the redundancy filter tree and therefore m messages need to be computed. The last step of the communication solution requires all redundancy-free messages to be combined into a joint probability and marginalized to the required distribution of input variables: this also takes time $O(mb^{s_{max}})$. So, the overall time complexity of the communication solution is $O(mb^{s_{max}})$.

6. CONCLUSIONS

The elimination of rumors in probabilistic agent communication is a difficult, longstanding problem that limits the applicability of graphical probabilistic models for knowledge representation in multiagent systems. Xiang [13] showed that, under a number of postulated assumptions, a correct probabilistic solution requires the topology of the communicating agents to be a tree. We relaxed some of the assumptions by postulating an oracular assumption, which states that the probability distribution of a variable published by an agent cannot be changed by the agents that subscribe to it. Under this new assumption, it is possible to compensate for rumors. We described how to do this in detail, using several graphical devices and algorithms that operate on them. We also analyzed the time complexity of our solution. It is our hope that the proposed techniques will allow more agent systems to take advantage of fully probabilistic knowledge representation formalisms, such as Bayesian networks. In future work, we plan to analyze and evaluate the tradeoffs between our approach and that of others (especially Xiang). We will also explore extending the AEBN communication protocol to allow bidirectional influences; our initial approach is to serialize cycles in communication.

7. REFERENCES

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

[2] H. Chan and A. Darwiche. On the revision of probabilistic beliefs using uncertain evidence. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 99–105, Acapulco, Mexico, 2003.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2003.

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

[5] Y.-G. Kim, M. Valtorta, and J. Vomlel. A prototypical system for soft evidential update. *Applied Intelligence*, 21(1), July–August 2004.

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

[7] S. Langevin and M. Valtorta. Performance evaluation of algorithms for soft evidential update in Bayesian networks: First results. In *Proceedings of the Second International Conference on Scalable Uncertainty Management (SUM-08)*, pages 284–297, Naples, Italy, October 2008.

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

[9] R. Pan. *Semantically-Linked Bayesian Network: A Framework for Probabilistic Inference over Multiple Bayesian Networks*. PhD thesis, Department of Computer Science and Electrical Engineering, University of Maryland, 2006.

[10] R. Pan, Y. Peng, and Z. Ding. Belief update in Bayesian networks using uncertain evidence. In *ICTAI*, pages 441–444. IEEE Computer Society, 2006.

[11] J. Pearl. *Causality: Modeling, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000.

[12] M. Valtorta, Y.-G. Kim, and J. Vomlel. Soft evidential update for probabilistic multiagent systems. *International Journal of Approximate Reasoning*, 29(1):71–106, January 2002.

[13] Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, Cambridge, 2002.