Communication Management Using Abstraction in Distributed Bayesian Networks

Jiaying Shen and Victor Lesser
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
{jyshen, lesser}@cs.umass.edu

ABSTRACT

Techniques were developed in previous work for managing communication in a controlled satisficing manner in two layer distributed Bayesian Networks. DEC-MDPs were used to sequence the information transferred in order to guarantee the required confidence level. In this paper, we introduce multiple abstraction layers into the Distributed Bayesian Network as a way of carrying more useful information in transmitted data to further reduce the number of messages that need to be sent. An algorithm is developed to automatically generate appropriate abstraction data. Techniques are introduced to effectively incorporate this abstraction data set into the DEC-MDP framework. We show that the appropriate addition of abstraction data actions simplifies the DEC-MDP while reducing the expected communication cost. This work provides us with a formal view of the use of abstraction in agent cooperation and begins to give us an understanding of when the less abstract data needs to be transmitted.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

DESIGN, PERFORMANCE

Keywords

Abstraction, Communication, Decentralized MDP

1. INTRODUCTION

In complex distributed applications, such as distributed interpretation (DI), a problem is often decomposed into a set of subproblems and each subproblem is distributed to an agent who will be responsible for solving it. The existence of interactions between subproblems means that the agents cannot simply solve the subproblems individually and then combine local solutions together. In such systems, the amount of communication among agents may be very significant in order to guarantee global optimality or even global consistency. Thus, "satisficing" approaches have been developed that trade off optimality for reduced communication [4]. One approach is for agents to generate local solutions based on their own data and then transmit these high level solutions to other agents. Based on the consistency and credibility of these local solutions, new local solutions may be generated or more detailed data sent until a sufficient level of consistency and credibility has been achieved among the agents. An important characterization of such distributed protocols is how much communication is required and the likelihood that the solution will be the same as that generated by an optimal centralized algorithm which uses all available information.

Most approaches to managing communication trade off solution quality for reduced communication, but only from a statistical point of view. The behavior of the algorithms are often analyzed over a collection of problems to say that p percent of the time they will get the required solution quality q with an average amount of communication c [4].

Shen et al. [11] took this satisficing approach to the next step by designing a parameterized algorithm where one can predict, for a desired confidence level in the final solution, the expected amount of communication the agents need. They studied these issues in terms of a two layered Distributed Bayesian Network (DBN), as shown in Figure 1. A decision-theoretic framework was proposed to model this multi-agent coordination decision problem. A decentralized Markov Decision Process (DEC-MDP) [1] can be constructed from the Bayesian Network structure to find a joint communication policy that minimizes the expected communication cost.
Each agent needs to decide its communication actions based only on its local observations. The framework takes the amount of communication into account, as well as the quality of the final solution. The agents use only the necessary amount of communication to achieve the required level of solution quality.

In this paper, we introduce abstraction layers into the Distributed Bayesian Network as a way of carrying more useful information in one piece of transmitted data to further reduce the number of messages that need to be sent. In order to effectively use the abstraction technique, there are two questions that need to be answered: How to generate the abstract information to be transferred, and When to transfer the abstract information. To answer the first question, we present an algorithm that automatically generates

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

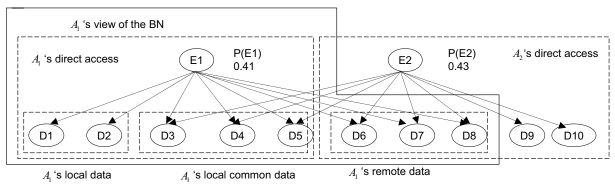


Figure 1: There are two events E_1 and E_2. Data $D_1, D_2, ...D_{10}$ are distributed between two agents. A_1 has access to $D_1, ...D_5$ and is responsible for solving E_1, while A_2 can see only $D_6, ...D_{10}$ and is responsible for solving E_2. The value of E_1 is dependent on A_1's data and vice versa. The objective is for A_1 and A_2 to decide what E_1 and E_2 are with required confidence while minimizing the expected communication cost.

appropriate abstraction data, which reduces the expected communication cost necessary to achieve the required confidence level. This is a greedy approach since the abstraction data enables the agent to immediately reach the desired confidence level. We chose this type of abstraction data because it is intuitive and straightforward to generate, yet it enables us to study the concept of abstraction and its use in reducing communication cost from a formal perspective. There are other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local solution. We plan to look into these different types of abstraction in future work.

To answer the second question, we examine three different approaches to incorporate the new abstract communication actions to the existing DEC-MDP. The *all data action selection* approach allows abstract communication actions as well as the normal communication actions in all the states. The *abstraction data action selection* approach allows only the abstract communication actions in all the states. In the *hierarchical action selection* approach, before an agent has acquired all of the abstraction data, the agents are not allowed to transfer any of the information available at the next more detailed level. Experimental work demonstrates that the hierarchical approach simplifies the DEC-MDP while still reducing the expected communication cost. Furthermore, experiments on larger networks with multiple abstraction layers show that the hierarchical approach is able to solve more and larger networks than the other approaches and therefore is more suitable for scaling up the approach.

Most work studying communication in DEC-MDP and its related models studies synchronizing communication, where every communication action synchronizes the agents' views of the world state [8, 10]. In contrast, this work studies the problem of what to communicate, where only part of the local observation is transferred between the agents. Abstraction has often been used in hierarchical planning community to represent abstract plans that can be further refined into specific actions. In particular, Clement and Durfee [6] used summary conditions, a form of abstraction, to coordinate the concurrent interactions of plans at different levels of abstraction between agents. Abstraction has also been used in distributed diagnosis research to identify the sources of problem. Chung and Barrett [5] demonstrated how Boolean expressions can facilitate finding minimal cost diagnoses in linear time. However, neither work studies the tradeoff between abstraction and communication. In the original Distributed Problem Solving work [7], abstraction was used as a mechanism for controlling the information needing to be communicated. However, the use of abstraction was not formalized. Nor was there a clear understanding of when the lower level data needed to be transmitted. Carver and Lesser [3] studied the use of multiple levels of abstraction to reduce the necessary communication. Like [7], the abstraction layers were predefined and the use of abstraction was not formalized. In contrast, the algorithm we introduce allows the system to generate appropriate abstraction data automatically without predefinition. Our study of the addition of abstraction layers into the Bayesian networks and transferring abstraction data as actions in the DEC-MDP provides us with a formal view of the use of abstraction in the management of communication cost in a distributed problem solving system.

The next section formally defines the problem we study and summarizes the DEC-MDP model we developed to generate the communication strategy. Section 3 introduces the algorithm that automatically generates the abstraction data that when transmitted can greatly facilitate the achievement of the confidence level. Section 4 discusses how the abstraction data can be incorporated to improve the system performance. Section 5 extends the different approaches discussed so far to larger networks with multiple abstraction layers. We conclude with a discussion of future work.

2. PROBLEM DEFINITION AND DEC-MDP MODEL

In an interpretation system there is a set of observable data that is caused by some possible events, which agents do not have the means to directly observe. The agents need to observe the data and identify the events that are most likely the cause of them. In many environments, the problem is inherently distributed and the network is fairly large. In those cases, it is common to distribute not only the data but also the interpretation task among several agents. Inevitably, there is a close interaction between the agents since the local sub-problems an agent is responsible for are often dependent on some of the data collected by other agents. The existence of subproblem interaction means that the agents will need to communicate during problem solving. Therefore, agent communication strategies can have a major effect on the cost of problem solving.

We use a two-layer Bayesian Network to represent the problem structure (Figure 1). The top level nodes are the events that are the possible causes of the observed data, while the leaves are the raw data gathered by various agents. There are two agents, each of whom has direct access to only a part of the observable data. The interpretation task is distributed to the two agents as well with each agent responsible only for part of the overall problem. Additionally, an agent only has knowledge of the network structure that is directly relevant to its interpretation task. The agents can either request or send data. The objective is to identify the most likely interpretation of the causal events with a target level of confidence using as little communication as possible. For example, in Figure 1, there are two agents A_1 and A_2. A_1 is responsible for interpreting the event E_1 and therefore knows about the part of the causal network that is relevant to E_1 . Out of the relevant data $D_1, ..., D_8$, it can directly observe only D_1 through D_5. D_6, D_7 and D_8 are *remote data* and may need to be transferred in order for A_1 to solve its local problem with confidence.

The **evidence** ϵ_{A_i} of an agent A_i are the values of the data that the agent has collected so far. They can be the values that are observed directly by that agent or acquired from the remote agent. An agent A_i is only able to find the most likely interpretation of the local events given its current evidence set, i.e., $MAPI(\epsilon_{A_i}) = argmax_h P(E_{A_i} = h|\epsilon_{A_i})$ [4]. Ideally the decentralized system should generate the interpretation that a centralized system will generate given the complete data values $MAPI(\epsilon^*_{A_i})$, where $\epsilon^*_{A_i}$ is the complete evidence set relevant to A_i. Unfortunately, with only partial knowledge of the relevant data values, it is not always guaranteed that the current local MAPI is the global MAPI. On the other hand, with the conditional probability table given by the BN, an agent can predict the probability of different relevant data configurations given its current evidence $P(\epsilon^*_{A_i}|\epsilon_{A_i})$. Hence we have the following definition.

Definition 1. **Confidence** *is the likelihood of the local MAPI being the global MAPI of* E_A *given the current known evidence of A.*

$$\begin{aligned} C(MAPI(\epsilon_A)) &= P(MAPI(\epsilon_A) = MAPI(\epsilon^*_A)) \\ &= \sum_{e \in \{\epsilon^*_A | MAPI(\epsilon_A) = MAPI(\epsilon^*_A)\}} P(e|\epsilon_A) \end{aligned}$$

What is important about the solution quality of a distributed system is not the likelihood of an event being the local MAPI, but whether the local MAPI agrees with the global MAPI, i.e., whether the decentralized system will generate the same result that a centralized system would. That is exactly what this definition of confidence achieves. Using confidence as a measurement of the solution quality of a decentralized system as compared to that of a centralized one, we can make a tradeoff between the quality of the solution and the communication cost. Another interesting and useful property of our definition of confidence is that it is guaranteed to reach 100% when the values of all the relevant data are known. As a result, a given confidence level can always be satisfied.

Given a problem structure in a two level BN such as the one in Figure 1 and a specified confidence threshold, we need to find a communication strategy. Such a communication strategy should specify what communication action each agent should take based on their current knowledge at each stage. In this paper we are considering only the case of synchronous communication. In other words, the two agents take turns to decide their actions in the sequence of $A_1, A_2, A_1, A_2, ...$ until the confidence threshold is reached.

Definition 2. *A* **communication strategy** $\pi_{\langle A_1, A_2 \rangle}$ *is a pair of policies* π_{A_1} *and* π_{A_2}. *Each policy* π_{A_i} *is a mapping from local evidence and communication history to a local action:* $\pi_{A_i} : \epsilon_{A_i} \times H \rightarrow a_i$.

There are three classes of actions for each agent: SEND, REQUEST and NULL. The content of the SEND and REQUEST actions at each stage are restricted by the communication history so far. An agent can send only the part of the local common data that the other agent has not yet acquired, and can request only a part of the remote data that it has no knowledge of. When the communication history indicates that it is not A_i's turn to act, the policy of that agent will always be NULL. A REQUEST action and its response by the other agent occur in the same stage. For example, in Figure 1, let us assume that initially $\epsilon_{A_1} = \{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1\}$ and $H = \emptyset$. A valid action for A_1 at this step would be "SEND D_3 and D_4". It could also "REQUEST D_8 and D_9". On the other hand, since it is not A_2's turn, the only valid action for A_2 is NULL. The cost model for this framework is very general. Each communication has its own cost, based on its action class and the particular data sent. The only restriction is that the cost of a REQUEST must also include the cost of the immediate reply by the other agent. For example, in our experiments SEND costs 1 per data sent. A REQUEST action costs 1 for the request plus a reply cost equal to a SEND action of the same data. "REQUEST D_8 and D_9" would cost 3.

To summarize, the problem can be defined by a two level BN and a required confidence threshold t. The goal is to find a communication strategy $\pi_{\langle A_1, A_2 \rangle}$ that allows the agents to interpret their local events with a confidence greater than, or equal to t, i.e., $C(MAPI(\epsilon_{A_1})) \geq t$ and $C(MAPI(\epsilon_{A_2})) \geq t$, while minimizing the expected communication cost.

To solve this problem, we model it as a DEC-MDP. Unlike a centralized MDP, in a DEC-MDP the process is controlled by multiple agents, each with a different view of the current global state. At each time step the agents' observations together uniquely determine the global state, though frequently no one agent has this complete view. A local policy for an agent is a mapping from its local histories of observations to an action. The goal of a decentralized MDP is to find an optimal joint policy for all of the agents that maximizes the expected total reward.

We model our problem in the framework of a decentralized MDP as follows.

- Every global state is in the form of $s = \langle \epsilon^*, i \rangle$, where ϵ^* is all of the low level data values observed by the system, and i is an external feature that indicates it is A_i's turn to communicate. s_0 is a dummy start state $\langle \emptyset, 0 \rangle$. When all the sensors collect their data, the start state transitions to one of the possible real global states $\langle \epsilon^*, 1 \rangle$ before any communication takes place.
- Ac_1 and Ac_2 are action sets of the two agents. $Ac_i \in \{$ NULL, SEND x, REQUEST $y\}$, where x is a subset of the local common data of A_i and y is a subset of the remote data. When it is not A_i's turn to communicate, its action is the NULL action. A joint action $\langle Ac_1 \neq NULL, NULL \rangle$ transitions a state $\langle \epsilon^*, 1 \rangle$ to $\langle \epsilon^*, 2 \rangle$, and a joint action $\langle NULL, Ac_2 \neq NULL \rangle$ transitions a state $\langle \epsilon^*, 2 \rangle$ to $\langle \epsilon^*, 1 \rangle$. If at state s, both agents have reached confidence level t for its local interpretation task, i.e., when $C(MAPI(\epsilon_{A_i})) \geq t$, $i = 1, 2$, then s is a final state.
- P is a transition probability table. $P(s'|s, a_1, a_2) = P(\langle \epsilon^*, i \rangle | \langle \epsilon^*, j \rangle, a_1, a_2) = 1$ iff $i \neq j$. Otherwise, $P(s'|s, a_1, a_2) = 0$. As a special case, the transition probability from s_0 to $s = \langle \epsilon^*, 1 \rangle$ after both agents execute NULL action is $P(s|s_0, NULL, NULL) = P(\epsilon^*)$.

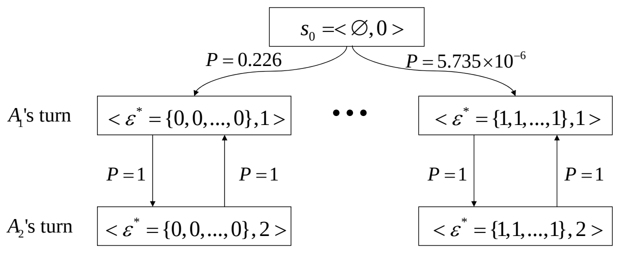

$s_0 = <\emptyset, 0>$
$P = 0.226$
$P = 5.735\times10^{-6}$
A_1's turn
$<\varepsilon^* = \{0,0,...,0\},1>$
$<\varepsilon^* = \{1,1,...,1\},1>$
$P=1$
$P=1$
$P=1$
$P=1$
A_2's turn
$<\varepsilon^* = \{0,0,...,0\},2>$
$<\varepsilon^* = \{1,1,...,1\},2>$

Figure 2: A portion of a typical decentralized MDP.

It can be calculated given the BN structure.

- R is a reward function. $R(s, a_1, a_2, s') = R(a_1, a_2) = -c(a_1) - c(a_2)$, where $c(a_i)$ is the cost of the communication action a_i, $i = 1, 2$.
- Ω_1 and Ω_2 are the sets of observations for the two agents. An observation $o_i \in \Omega_i$ is the data value just sent or received by A_i. As a special case, after s_0, each agent observes the values of its local data. An observation sequence $\overline{o}_i = \langle \epsilon^0_{A_i}, H\rangle$, where $\epsilon^0_{A_i}$ is the initial observation of A_i's local data set before any communication occurred in the system, and H is the communication history of the system.
- $O(s, a_1, a_2, s', o_1, o_2) = 1$ iff o_1 and o_2 are the new data sent or received by the agents after them taking actions a_1 and a_2 at state s. Otherwise it equals to 0.
- A local policy π_i for agent A_i is a mapping from the local observation sequence $\overline{o}_i$ to an action $a_i \in Ac_i$. A joint policy $\pi = \langle \pi_i, \pi_j\rangle$ is a pair of local policies, one for each agent.

For example, Figure 2 is the DEC-MDP built for the problem in Figure 1. A possible global state s when both agents just observed their local sensor data might be $\langle\{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1, D_6 = 0, D_7 = 1, D_8 = 1, D_9 = 0, D_{10} = 0\}, 1\rangle$. The possible actions for A_1 are NULL, SEND x, and REQUEST y, where $x \subseteq \{D_3 = 0, D_4 = 1, D_5 = 1\}$ and $y \subseteq \{D_6, D_7, D_8\}$. Since it is not A_2's turn to communicate, its only action is NULL. If A_1 chooses REQUEST $\{D_7, D_8\}$, then the next state is $\langle\{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1, D_6 = 0, D_7 = 1, D_8 = 1, D_9 = 0, D_{10} = 0\}, 2\rangle$ and both agents observes $\{D_7 = 1, D_8 = 1\}$. The observation sequence for A_1 is $\overline{o}_1 = \{\langle D_1 = 1, D_2 = 0, \ldots, D_5 = 1\rangle, \langle D_7 = 1, D_8 = 1\rangle\}$, and the observation sequence for A_2 is $\overline{o}_2 = \{\langle D_6 = 0, D_7 = 1, \ldots, D_{10} = 0\rangle, \langle D_7 = 1, D_8 = 1\rangle\}$. The policy π_i decides which action to take for each agent based on its current $\overline{o}_i$. Remembering the entire communication history H for both agents have two purposes. First, it includes the newly acquired evidence received from the other agent. Second, even for the agent who just SEND part of its own data without receiving any new evidence, it is important to remember this so that in the future it does not send the same information unnecessarily.

3. GENERATING THE ABSTRACTION LAYER

So far we have discussed two layer networks, which means that there are no intermediate abstraction data. Many domains, however, have structures that can be exploited to improve the performance of both local processing and inter-agent merging of evidence. Domain structure can result from a number of factors such as independencies among subsets of events or between events and certain data subsets and cases where only a fraction of the combinations of some data are informative. Many of these situations are best captured by using BNs that include intermediate nodes that lie between the event and data levels.

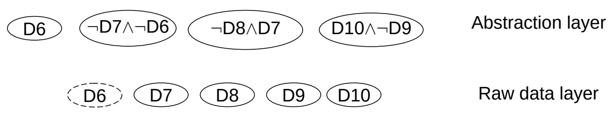
```
1.1 Input: agent $A$'s view of BN, the values of
    the local data, the required confidence
1.2 Output: a set of logic expressions of the
    remote data that, if true, put the local
    confidence above the required confidence
1.3 Set DataList = all the remote data of $A$,
    potential = null
1.4 Initialize LogicTree
1.5 while DataList.next is not null do
        currentLeaf = DataList.next
        add currentLeaf and $\neg$currentLeaf as new
        children to LogicTree.root
        if confidence given currentLeaf $\geq$ required
        confidence then
            mark currentLeaf
            continue
        if confidence given $\neg$currentLeaf $\geq$
        required confidence then
            mark $\neg$currentLeaf
            continue
        if potential == null then
            potential.add(currentLeaf)
            potential.add($\neg$currentLeaf)
            continue
        for each $p$ in potential do
            potential.add(addChildren
            (currentLeaf, $p$), addChildren
            ($\neg$currentLeaf, $p$))
1.6 convert the marked part of LogicTree to a set
    of logic expressions and return it
1.7 addChildren (currentLeaf, $p$)
    begin
        Add $p$ to LogicTree as a child of
        currentLeaf in a depth first fashion, check
        the confidence at every step. If confidence
        $\geq$ required confidence, stop going deeper
        into this branch and mark it.
        Output: the unmarked subtree of
        LogicTree starting from currentLeaf
    end
```

Algorithm 1: The algorithm for generating the abstraction layer

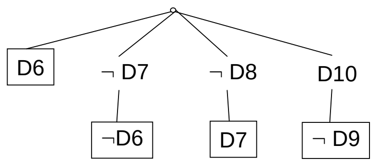
Research on learning Bayesian Networks from data [2, 9] focuses on finding the BN structure that best matches the collected data. These techniques are important in constructing the BN structure that represents the problem we are trying to solve. However, we need other methods to find an appropriate abstraction layer from the existing BN that, when transmitted from the remote agent, more efficiently conveys the necessary information to facilitate the local problem solving. In other words, this abstraction layer, when acquired, should be able to reduce the expected communication necessary to achieve the required confidence level.

D6
D7
D8
D9
D10

(a) The action options for the *raw data action selection* approach



(b) The action options for the *all action selection* approach. The data that can be transferred are divided into two categories.

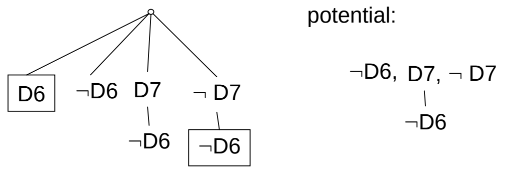


(c) The corresponding **LogicTree** to the abstraction data.

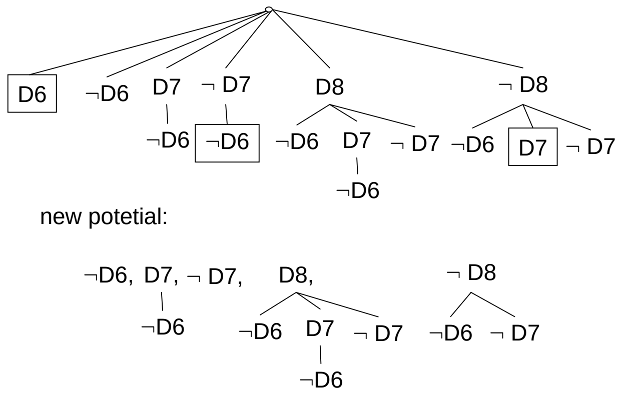
Figure 3: An example of the abstraction layer. Remote data includes $\{D6, \cdots, D10\}$. The required confidence level is 75%

We achieve this goal by developing an algorithm (Algorithm 1) that automatically generates an abstraction layer given a value combination of an agent's local data and the desired confidence level. The basic idea behind the algorithm is to find a set of logic expressions consisting of the remote data such that if at least one of the expressions is true the required confidence is reached. For example, an agent has 5 pieces of remote data $\{D6, \cdots, D10\}$ and needs to achieve a confidence level of 75%. In the original system, the only possible actions are to transfer some or all of these raw data. We will call this approach *raw data action selection*. However, if one or more of $D6$, $\neg D7 \wedge \neg D6$, $\neg D8 \wedge \neg D7$ and $D10 \wedge \neg D9$ are true, then the agent immediately has a local solution with a confidence level of 75%. These data carry more abstract information than the raw data themselves and therefore are more efficient, though the information is not as refined. If these data values can be acquired as well as the raw data, it can potentially yield huge savings on the communication cost. Additionally, the values of the abstraction data also give valuable information about what the values the raw data may have. For example, if the value of $\neg D7 \wedge \neg D6$ is transferred and is true, then the process can immediately terminate because the confidence level of 75% has been reached. If the value is false, even though the desired confidence level is not immediately achieved the agent does now have the knowledge that $D6$ and $D7$ cannot be false at the same time. This information can be retained by updating the BN. Figure 3(a) shows the action options of the *raw data action selection* approach for this example while Figure 3(b) shows the action options if the abstraction data can also be transferred.

The desired set of logic expressions can be generated by simply enumerating all of the possible combinations of the remote data values and selecting those that will enable the agent to reach its desired confidence level. Our algorithm improves efficiency by a logarithmic factor over the exponential brute force enumeration by reusing the subgraphs of logic expressions.



(a) **LogicTree** and **potential** before adding $D8, \neg D8$ to **LogicTree**.



(b) **LogicTree** and **potential** after adding $D8, \neg D8$ to **LogicTree**.

Figure 4: An example of the key step of Algorithm 1.

Figure 4 gives an illustration of an episode showing how the algorithm is run for the example in Figure 3. We use a tree, which we call a **LogicTree**, to keep track of all the data values examined so far. The nodes along the same path are connected by the $\wedge$ operator, and the different branches represent different value assignments. The marked branches are the ones that pass the confidence test, i.e., if true, the desired confidence level is reached. The set **potential** is used to keep track of the subtrees examined so far that did not pass the confidence test. Only the members of **potential** should be examined as a subgraph of longer expressions. Figure 3(c) shows the final **LogicTree** for this example.

If the values of all the remote data are acquired, the local confidence is guaranteed to reach 100%. Therefore, we limit the depth of the **LogicTree** to be less than the number of remote data. When the maximum depth is the number of remote data minus 1, the algorithm generates all of the expressions that have the desired property. Reducing the depth of the **LogicTree** results in a smaller number of abstraction nodes generated and therefore a smaller DEC-MDP. This tradeoff will be further discussed in the next section. In general, the number of abstraction data generated should be no more than the number of raw data. For each path in the **LogicTree**, we record the likelihood of the abstraction data represented by the branch being true, and we choose the n abstraction data with the highest likelihood of being true to incorporate into the DEC-MDP, where $n <$ the number of raw data.

When given a BN and a desired confidence level, the agent generates an abstraction layer for each raw data value combination and adds them to its action options for the states that have the corresponding raw data values. The expanded DEC-MDP can be solved to generate a communication strategy. We call this approach the *all data action selection* approach. Since the raw data communication actions are competing with the abstraction actions, an agent chooses to transfer an abstraction data if and only if it will result in a lower expected communication cost than transferring any of the raw data. Therefore, the new system performance should be no worse than the original system in terms of the expected communication cost.

We compared the performance of the *all action selection* approach and the *raw data action selection* approach. The cost of sending a piece of abstraction data was equal to the cost of sending raw data. We used the Iterative Algorithm introduced in [11] to solve the DEC-MDP. We ran experiments on 100 problem structures with 2 high level events and 10 raw data (5 local to each agent) for different confidence levels. All of the networks were fully connected, which means that for both agents to have the complete evidence, 10 pieces of data needed to be transmitted. The corresponding curves in Figure 5 shows a comparison of the minimum expected communication cost generated by both systems. Column (a) in Table 1 shows the percent improvement in the expected communication cost when transmitting abstraction data in addition to the raw data. As shown, the *all data action selection* has a noticeable improvement over the *raw data action selection* approach. This illustrates that the addition of the abstraction data does help reduce the communication cost. The improvement is most significant when the required confidence is between 70% and 80%. When the confidence level is low, the expected communication cost is already low so that it is difficult to achieve a significant improvement. On the other hand, when the required confidence level is high, there are much fewer abstraction data generated resulting in a smaller improvement.

4. HIERARCHICAL ACTION SELECTION

Introducing the communication actions that transmit the values of the new abstraction data leads to both a larger state space and a larger action space for the generated DEC-MDP. As a result, the time needed to solve the new DEC-MDP can be significantly greater than the original DEC-MDP, which only has the raw data transmission as its actions. Column (c) in Table 1 shows the average time the *all data action selection* approach took to solve the DEC-MDP, where the average time needed for the *raw data action selection* approach equals 1.00. In this section, we introduce techniques that address this problem.

First we examine the case where the agents only transfer the abstraction data between them. We call this approach the *abstraction data action selection* approach. While the size of the DEC-MDP generated is often much smaller than that of the original DEC-MDP, one major drawback of this approach is that we can no longer guarantee that the required confidence level can be reached. Only when at least one of the abstraction data is true will the desired confidence level be reached. The corresponding curve in Figure 5 shows the expected communication cost achieved for the cases where the desired confidence level can be reached. When the required confidence level is high, there are fewer abstraction data generated, and therefore the expected cost is much lower than those of the two approaches we discussed in the previous section. Notice, however, that this is a false savings as the desired confidence level cannot always be reached, which is not reflected in the figure.

We seek to combine the advantages of the *all data action selection* approach and the *raw data action selection* approach so that we can save time on solving the DEC-MDP as well as guarantee the required confidence level. We achieve this by restricting legal actions for different states. Before an agent has acquired all of the abstraction data, the agents are not allowed to transfer any of the raw data. We call this approach the *hierarchical action selection* approach. An agent prefers to transfer abstraction data because it abstracts from multiple pieces of raw data and thus is a more efficient information carrier. Only when the acquisition of all of the abstraction data cannot achieve the desired confidence level will an agent start to acquire the raw data in order to get the necessary information.

Like the *all data action selection* approach, the *hierarchical action selection* approach guarantees that the desired confidence level can be achieved. Figure 5 illustrates the comparison of the performance of the approaches we have discussed. As we expected, the *all data action selection* approach performs the best. In fact, it generates the optimal solution for any DEC-MDP whose action space in-

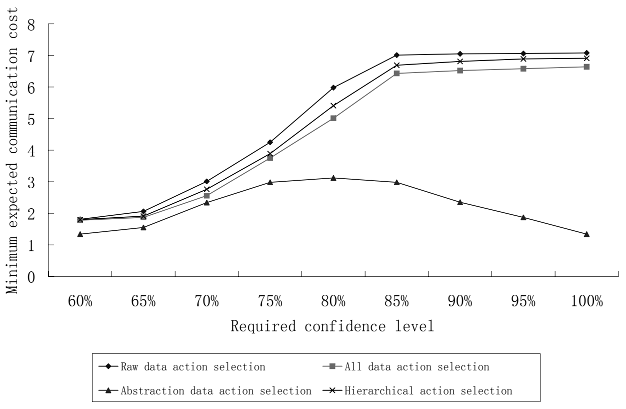


Figure 5: A comparison of the minimum expected communication cost given different action selections

required confidence	(a)	(b)	(c)	(d)
60%	1.66%	0.55%	1.25	0.89
65%	9.22%	7.28%	1.53	0.77
70%	14.95%	8.31%	1.49	0.69
75%	11.76%	8.47%	1.52	0.63
80%	16.22%	9.53%	1.61	0.59
85%	8.27%	4.56%	1.41	0.65
90%	7.52%	3.40%	1.21	0.77
95%	6.80%	2.41%	1.10	0.82
100%	6.21%	2.40%	1.09	0.87

Table 1: Performance compared to raw data action selection approach for single abstraction layer in a 2-10 network. (a) Expected communication cost improvement of *all action selection*. (b) Expected communication cost improvement of *hierarchical action selection*. (c) Time needed to solve the DEC-MDP for *all action selection* normalized by that for raw data action selection. (d) Time needed to solve the DEC-MDP for *hierarchical action selection* normalized by that for raw data action selection.

cludes transferring both the abstraction data and the raw data. On average, the *hierarchical action selection* approach outperforms the *raw data action selection* approach. However, there are cases where the *hierarchical action selection* approach requires more communication than the *raw data action selection* does. In those BNs, it is often the case that there is a low likelihood of any of the abstraction data being true. Column (b) in Table 1 shows the amount of improvement in the minimum communication cost the *hierarchical action selection* approach gains over the *raw data action selection* approach. The pattern is similar to that of column (a).

Column (d) in Table 1 shows the average time needed to solve the DEC-MDP for the *hierarchical action selection* approach normalized by the average time needed for the *raw data action selection* approach. It achieves substantial savings. Even though the *hierarchical action selection* approach does not reduce the size of the action space compared to the *all action selection* approach, it does reduce the number of legal actions available to any given state. This also decreases the size of the state space because H, the communication history, has fewer possibilities. These two factors combined together contribute to the time savings, and the larger the network is, the more substantial the savings should be. We plan to run more experiments on larger networks to verify this observation.

All of the experiments we have shown so far set the depth of the **LogicTree** in Algorithm 1 to be the number of remote data minus 1, i.e., 4. A shallower **LogicTree** will generate fewer abstraction data and result in a higher expected communication cost to reach the same confidence level. However, generating the abstraction layer and finding the communication strategy will take less time. Our future experiments will explore this tradeoff between computation time and communication cost as the depth is varied.

5. MULTIPLE LEVELS OF ABSTRACTION

The idea of varying the depth of the **LogicTree** leads to an interesting way of generating a hierarchy of abstraction data. We can modify Algorithm 1 to generate a set of abstraction data of different lengths. The more literals in the abstraction data, the more abstract it is, the more efficiently it carries information, and the less refined the information is.

Just like in the single abstraction layer case, there are different ways of incorporating multiple layers of abstraction data into the action space of the DEC-MDP. We did experiments on the all data action selection approach and the hierarchical action selection approach, and compared their performance to that of the raw data action selection approach. In the all data action selection approach, all the abstraction data of various length are added to the action space to compete with the raw data. In the hierarchical action selection approach, the agents transfer the most abstract data first. If the confidence level is not achieved, then the agents start transferring the next level of abstract data. This process goes on until the confidence level is reached.

We tested these approaches on networks of different sizes: 4 high level events and 20 low level data (4-20 networks); 6 high level events and 30 low level data (6-30 networks); 8 high level events and 40 low level data (8-40 networks). We generated two abstraction layers for each of the networks. Figure 6 shows the percentage of the networks we tested on

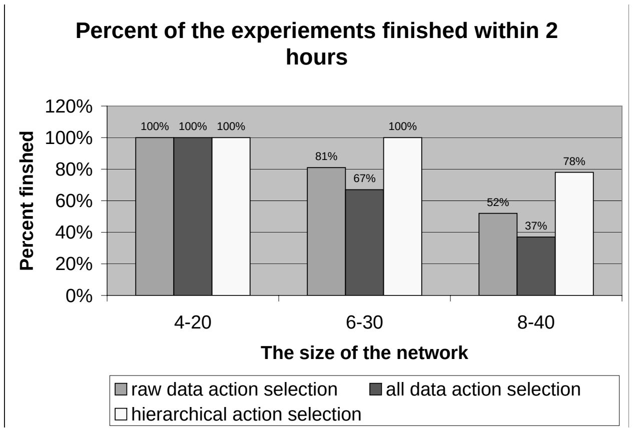


Figure 6: The percentage of the problems tested that each approach was able to finish solving within 2 hours.

required confidence	(a)	(b)	(c)	(d)
60%	3.87%	2.14%	1.33	0.91
65%	10.42%	8.19%	1.49	0.79
70%	16.01%	12.34%	1.52	0.62
75%	19.84%	15.62%	1.59	0.56
80%	18.26%	13.43%	1.71	0.49
85%	12.22%	9.53%	1.59	0.58
90%	8.49%	6.72%	1.39	0.68
95%	5.71%	4.43%	1.19	0.76
100%	4.23%	3.12%	1.10	0.78

Table 2: Performance compared to the raw data action selection approach for multiple abstraction layers in a 6-30 network. (a) Expected communication cost improvement of *all action selection*. (b) Expected communication cost improvement of *hierarchical action selection*. (c) Time needed to solve the DEC-MDP for *all action selection* normalized by that for raw data action selection. (d) Time needed to solve the DEC-MDP for *hierarchical action selection* normalized by that for raw data action selection.

that each of the three approaches was able to finish within two hours. It shows that hierarchical action selection approach was able to finish the policy search better than either of the two other approaches. The larger the network, the bigger an advantage the hierarchical approach has over the other two. Table 2 shows the performance comparison of the all data action selection approach and hierarchical action selection approach over the raw data action selection approach. The data is shown only for the 6-30 networks for which all of the three approaches finished searching for the optimal policy. The first abstraction layer is composed of the abstraction data with no more than 5 literals, and the second layer is composed of the abstraction data with no more than 10 literals and greater than 5 literals. The results show a similar pattern as those of the single abstraction layer experiments. The all data approach is not practical for large networks or more than a few abstraction layers, since it substantially expands both of the state space and action space of the DEC-MDP. However, it does give a lower expected communication cost than the raw data action selection approach when it can solve the problem. On the other hand, the hierarchical approach proves to be more effective than both the raw data action selection approach and the all data action selection approach. It is able to solve more and larger networks, and on average it takes much less time. Furthermore, for the networks that both approaches were able to finish the policy search, the solution generated by the hierarchical action selection approach has a lower expected communication cost on average than that generated by the raw data action selection approach.

6. CONCLUSIONS AND FUTURE WORK

In this paper we investigated the techniques of transferring abstraction data in addition to raw data in Distributed Bayesian Networks to reduce the required communication cost. We introduced an algorithm that automatically generates appropriate abstraction data that facilitates the achievement of the required confidence level and reduces the necessary communication cost. We also discussed approaches to incorporate the new abstraction data into the DEC-MDP framework effectively. Both the improvement in the minimum expected communication cost and the time savings in solving the DEC-MDP make the *hierarchical action selection* an attractive approach, especially for the systems which require a mid-ranged confidence level. We further extended the different action selection approaches to larger networks and multiple abstraction layers. The hierarchical action selection approach was shown to be able to solve problems with larger networks than the other approaches.

This work allows us to look at the use of abstraction to reduce communication cost from a formal perspective. The *hierarchical action selection* approach defines when and how the abstraction data and the raw data should be transferred in the communication process. The addition of transferring abstraction data to the communication actions can be extended to reduce the communication cost of other Cooperative Distributed Problem Solving (CDPS) applications as well. Though the algorithm that we used to generate the abstraction data is specific to the DBN used to represent the DI problems, given a CDPS problem with specific semantics, there often is abstraction data that can be generated and used to transfer information more efficiently. The hierarchical action selection approach then can be used to incorporate this abstraction data to further minimize the communication cost.

We plan to do more experiments with larger networks, and we predict that the savings of the *hierarchical action selection* approach shown in this paper will be more significant for larger networks. The influence of different depths of **LogicTree** in Algorithm 1 on the system performance will also be investigated. In this work, we looked at one particular type of abstraction. There are other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local data. We plan to look into these different types of abstraction in future work. We also intend to investigate other techniques in generating multiple layers of abstraction data, such as different levels of intermediate interpretation results.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-0219606 and IIS-0414711. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, November 2002.

[2] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

[3] N. Carver and V. Lesser. The DRESUN testbed for research in fa/c distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of ICMAS-95*, pages 33–40, 1995.

[4] N. Carver and V. Lesser. Domain monotonicity and the performance of local solutions strategies for CDPS-based distributed sensor interpretation and distributed diagnosis. *International Journal of Autonomous Agents and Multi-Agent Systems*, 6:35–76., 2003.

[5] S. H. Chung and A. Barrett. Distributed real-time model-based diagnosis. In *Proceedings of the 2003 IEEE Aerospace Conference*, 2003.

[6] B. J. Clement and E. H. Durfee. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 495–502, 1999.

[7] L. Erman, F. Hayes-Roth, V. Lesser, and D. Reddy. The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[8] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS03)*, 2003.

[9] D. Heckerman. A tutorial on learning with bayesian networks. In *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

[10] R. Nair, M. Roth, M. Yokoo, and M. Tambe. Communication for improving policy computation in distributed pomdps. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, 2004.

[11] J. Shen, V. Lesser, and N. Carver. Minimizing Communication Cost in a Distributed Bayesian Network using a Decentralized MDP. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)*, volume AAMAS03, pages 678–685, Melbourne, AUS, July 2003. ACM Press.