Rating the Usefulness of Information to Communicate

Elise H. Turner and Collette M. Matthias

UNH CS Technical Report #94–22

Unfortunately, the original TR is not available locally, and this was reconstructed from an HTML version that lacked the figures. For the complete version, please contact the Computer Science Department at UNH.

1 Introduction

By communicating, agents in distributed artificial intelligence (DAI) systems can share information. However, agents working together to solve complex problems cannot communicate everything that they know. Due to bandwidth limitations and processing constraints, agents are forced to choose the small portion of their information which they will communicate. For communication to be most effective, an agent should select the information that will be most useful to other agents in helping them to achieve the collaborative task.

Unfortunately, agents also have limited resources which can be used to select information to communicate. Here again, processing constraints are important. Communication is in service of the primary collaborative tasks which the agents work together to perform. Computational effort needed to process communication, including effort needed to support communication, must be added to the cost of the collaborative task. Agents also have only limited knowledge of each other and the world. Consequently, any method for selecting information should rely only on easily accessible knowledge that we are sure will be available to the agent.

Our research addresses the problem of selecting information to communicate given the limited resources of agents in distributed systems. We focus on an often-overlooked resource - the agent's knowledge of its own reasoning. When agents are collaborating to achieve a goal using similar reasoning capabilities and strategies, information that is useful to one will be useful to others. So, if an agent can determine the role of some information in its own reasoning about the collaborative task, it gains insight into that information's usefulness to others.

We are developing an approach to selecting information to communicate based on this idea (10). *Usefulness heuristics* translate problem solving into a *usefulness rating* for information items that are accessed while reasoning about the collaborative task. These heuristics may draw only from knowledge which is easily accessed during problem solving. Specifically, heuristics may reflect: the *knowledge structure* which contains the information, the *type and difficulty* of the reasoning used to access the information, or the *effect* that information has on performing the task. Since the agent simply monitors its own reasoning, determining a usefulness rating requires *no additional knowledge* and *little computational effort* beyond that required to reason about the collaborative task. Our approach does not preclude making use of additional knowledge or computational effort when the resources are available. In fact, it is our intention that usefulness ratings will be integrated with other methods for selecting information.

In earlier papers we describe the general approach in more detail (10,11). In this paper, we present an initial implementation of the approach and our early results. We describe our implementation in Section

1. In Section 3,

we discuss preliminary results which suggest that our approach is promising. In Section 4, we discuss the more general approach, how our approach can be integrated with others, and the types of systems which may benefit from our approach. We also present our plans for extending the approach in this section.

2 Implementing Usefulness Ratings for NONLIN

In this initial implementation of our approach, two NONLIN planners (8) work together to plan a night out. We chose NONLIN because it implements a widely-used technique in planning. Heuristics developed for NONLIN should be transferable to other non-linear, hierarchical planners. NONLIN was also chosen because it is available over the Internet (4).

2.1 An Overview of NONLIN

NONLIN uses *schemas* to describe the steps that may be taken to achieve a goal. Each schema has a *todo* slot which specifies the effect that the schema is intended to achieve. Schemas also contain a list of variables which are bound during planning.

NONLIN plans hierarchically by using schemas at different levels of detail. A higher-level schema specifies steps that are the *todos* of lower-level schemas. When the higher-level schemas are expanded, these lower-level schemas are added to the plans. Schemas are added to NONLIN's plan in the form of *nodes*. Nodes contain the schema as well as additional information about how the schema fits into the plan. When a node is added to the plan, its effects are added to the world model. At this point, the plan is also checked for harmful interactions between plan steps. If a harmful interaction is found, NONLIN looks for ways to prevent the interaction. Interactions can be prevented by binding variables so steps do not interact or by ordering steps. After NONLIN has found all ways of preventing the interaction, it chooses one and continues planning. The plan is complete when all steps are specified as primitive actions.

2.2 Creating Collaborating NONLINs

NONLIN was designed to work as an independent planner, not as part of a multi-agent system. This suited our purposes because we were assured that the planning strategy was not developed to support some specific communication strategy. However, we had to make two extensions to NONLIN to allow it to collaborate. These extensions were made solely to allow the NONLIN s to exchange information so that we could begin to evaluate our approach.

2.2.1 A method for communication was developed.

To allow the NONLIN s to exchange information, we developed a language for them. We first created message types, described in Section 2.3 and shown in Figure

1. The type of the message indicates the

type of information and how it is to be used. Each message type specifies the structure and content of the message. To simplify communication for our initial tests, both NONLIN s have the same

names for schemas and their variables, goals and operators, and entities in their world models. Consequently, these can simply be referred to by name.

There is still some overhead in generating and interpreting messages, however. Within their plans, NONLIN s refer to these items using symbols which have been generated during planning. The meaning of these symbols are unique to each NONLIN , so the common name must be extracted to generate a message. When the message is received, the correct instantiated symbol must be found to understand it. In addition, specific nodes are often referred to internally using node numbers. These node numbers may not be the same in both planners. In order to identify a node, its *todo* and variable bindings must be communicated. The receiver can then match this information with all of the nodes in its current plan in order to find the appropriate node number. If the appropriate node has not yet been created because its parent has not been expanded, the message is handled when the node is created. Of course, the greatest overhead is integrating the information into the receiver's current plan. This differs for each type of message and is described briefly in Section 2.3.

2.2.2 NONLINs can execute communicative actions during planning.

NONLIN was designed to create a complete plan which could then be executed. Clearly, for information to be used during planning, it must be communicated while the plan is being formed. We implement this by calling functions to send or receive messages at specific times during planning. Functions to send messages are called when information is accessed or created which could be communicated. First, the usefulness rating for the information is computed. If the usefulness rating exceeds the *threshold for communication*, the message is created and sent. The function to receive a message is called just before NONLIN chooses a node to expand. This function takes the first message off of the message queue and handles it.

In principle, the usefulness ratings for all information should be maintained and a message should be sent as soon as the usefulness rating exceeds the threshold for communication. For efficiency, however, we may continue to keep usefulness ratings for only some types of information or to handle communication only at certain times. In any case, this simple mechanism for communication allows the NONLIN s to send messages to exchange information and is sufficient for our purposes.

2.3 Message Types

For this implementation we developed message types to communicate information that we believe will be most useful. In earlier papers, we classified useful information in terms of the agent's problem solving (10,11). We have found that information is useful if it helps the agents to do at least one of the following:

2.3.1 Avoid redundant problem solving.

For example, agents should communicate the results of expensive computations.

2.3.2 Prune the search space.

For example, agents should communicate choices made when there are many alternative. This allows both agents to prune parts of the search space that do not lead to or emanate from these choices.

2.3.3 Pool relevant facts.

In (10) we suggest some heuristics for allowing agents to pool relevant facts. In our current implementation, this is not needed since agents have the same plan schemas and the same initial model of the world.

We currently focus on communicating information which decreases the time of problem solving - that is, which avoids redundant problem solving or prunes the search space. To this end, we have implemented two message types: *effect* and *TOME-entry*. These message types communicate partial solutions that may result from lengthy reasoning and may be in competition with many other partial solutions. The form of each message type and an example are given in Figure 1. A third message type, *modify*, is needed for the agents to resolve conflicts between their plans.

2.3.4 Effect messages.

Effect messages help agents to avoid redundant problem solving by communicating how a node was expanded. Node expansion is often a time-consuming process because the agent must find the schemas which can achieve the parent node's *todo* and must ensure that its conditions are satisfied. Effect messages also help agents to prune the search space by specifying which of many possible schemas will be used in the plan.

Effect messages include the schema name, variable bindings and *todo* so the receiver can add the node expansion to its current plan. By adding the node to the plan, its effects are added to the receiver's world model.

2.3.5 TOME-entry messages.

The TOME stores information about the interactions between plan steps. TOME-entry messages communicate the orderings established to avoid harmful interactions between plan steps. Checking for the interactions and finding the possible links to resolve them is another time-consuming process for NONLIN. TOME-entry messages thus help the system to avoid redundant problem solving. There also may be many different links which could be used to resolve an interaction. The TOME-entry message helps to prune the search space of the other agent by specifying the link used.

Links are internally represented by node numbers. TOME-entry messages must specify each node by its *todo*. The receiver creates the proper link by finding the node numbers in its plan with matching *todos*. The link is then used to mark the plan nodes with ordering information.

2.3.6 Modify messages

Both effect and TOME-entry messages help the agents to agree on their plans so that they can continue to collaborate. In some cases, though, an agent might be given information which conflicts with its current plan. For example, an effect message may communicate an expansion for a node that the receiver has already expanded differently. We resolve this conflict by making one agent subordinate to the other. If a message is received from a subordinate which conflicts with the receiver's current plan, the receiver sends the subordinate a message to modify its plan. This modify message is always sent and the conflict is resolved without further communication. We are concerned about the agents agreeing on plans only because it allows them to continue to collaborate. In a fully-implemented system, we would expect agents to have a more sophisticated mechanism for resolving conflicts.

2.4 Heuristics and Usefulness Ratings

Usefulness ratings estimate the value of an information item to other agents. They are calculated from heuristics which monitor specific aspects of problem solving. In keeping with the goals of our research, the heuristics and, in turn, the usefulness ratings, must rely solely on knowledge needed for problem solving. In addition, both must be calculated using very little computational effort beyond that required for problem solving.

We have previously suggested some general-purpose heuristics which might be applied to a wide variety of planners (10,11). These serve as a starting point. For each planner, specific heuristics must be implemented which translate problem solving into usefulness values for a particular DAI system. Initially, we expected one set of heuristics and a single method of calculating usefulness ratings to be appropriate for all information. After developing heuristics for NONLIN, we believe that specific types of information require their own heuristics and calculations for usefulness ratings. For our system with two collaborating NONLIN s and for the message types described above, we have created the following:

2.4.1 Heuristics and Usefulness Ratings for Effect Messages

Heuristics for effect messages include:

1. number of schemas:

   This gives the number of schemas that have the appropriate *todo* and could be used for the expansion. It reflects the value of information in terms of pruning the search space. When NONLIN expands a node, it creates a list of all possible schemas that could serve as the expansion. It selects one as the expansion and places all others on its list of points for backtracking. This heuristic is implemented simply by finding the length of that list.

   (a) number of conditions:

   Before a schema can be used to expand a node, its conditions must be met. NONLIN has three varieties of these conditions, described in (8). Checking the conditions may be computationally expensive, so this heuristic measures redundant problem solving that will be avoided. This heuristic is implemented by taking the length of the list of conditions associated with each schema.

   (b) number of variable bindings:

   By counting the possible variable bindings, this heuristic also measures how many options will be pruned from the search space. The heuristic is implemented by storing the number of bindings available for each variable. The total number of variable bindings is calculated by adding together the number of bindings for each variable with more than one possible binding.

   The usefulness rating for the effect message is the sum of all of the heuristics.

   Heuristics and Usefulness Ratings for TOME-entry Messages

   Heuristics for TOME-entry messages include:

   (c) inferences to find interactions:

   This heuristic is meant to avoid redundant problem solving. Interactions in NONLIN are found by checking the purposes and effects of a new node against those of the existing nodes. This is implemented in a system of embedded loops. Some work is performed within each loop, with the most work performed in the innermost loop which actually creates the possible links. This heuristic is implemented by incrementing a counter each time the innermost loop is executed.

(d) number of possible links:

This is a measure of how much of the search space is pruned. It is calculated by finding the length of the list of links returned by the function described above.

More work is generally done to find an interaction than to reason about a schema. It is also more difficult to pursue alternative links than alternative variable bindings. So, to make usefulness ratings commensurate for TOME-entry and effect messages, we weight the sum of the TOME-entry heuristics.

3 Results of Using Usefulness Ratings to Select Information

The domain used to collect our initial results is planning an evening out. There are 57 plan schemas available. These include 12 high-level plans for the evening's activities. For example, "go to dinner and go to a movie" is such a high-level schema. The other schemas add details to the high-level plans. Plans formed have between 11 and 14 primitive actions. The average number of primitive actions in a plan is 12.8.

To ensure that there would be some variation in planning between the two NONLIN s, the NONLIN s choose different schemas from the same list of possibilities. One planner always chooses the first schema on the list; the other always chooses the last. The plan libraries were varied by modifying the order of the schemas so that each of the six test scenarios would create a different plan. We assume that the planners will agree on the first completed plan, so both planners stop when the first completes a plan.

Our initial tests were intended to give us preliminary feedback about whether or not usefulness ratings could reflect the actual value of information. The value of the information communicated is reflected in both the quality of the plan and in the time spent planning. Since our agents did not have the ability to negotiate about plans and since our initial heuristics and message types did not focus on plan quality, we use only planning time as a measure of the value of the communicated information.

If our usefulness ratings reflect the value of information, messages should become more useful as their ratings increase. When a too-low threshold for communication is used, messages which are not very useful will be communicated. These messages may not save the system the time spent to communicate them, so, the benefit per message will be low. With a too-high threshold for communication, some valuable messages would not be communicated. Although each message sent would have a large benefit, the total benefit would not be as great because not all useful messages would be communicated. Therefore, the average total benefit per message would be low. If our usefulness ratings accurately reflect the value of messages, we would expect the benefit per message to increase as the threshold of communication is increased, up to some point. After that point, we would expect the benefit per message to begin to decline.

Figure 2 shows the benefit per message for collaborative plans for an evening out at several thresholds for communication. The benefit per message is calculated for all six scenarios. The benefit is calculated by subtracting the time spent to form the collaborative plan from the time spent to form a single plan. Because the collaborative planners use different strategies for planning, the time for a single planner is the average planning time of two single planners, each using one of the planning strategies. We also take the average of the two collaborating planners to find the time to form the collaborative plan. The benefit is then divided by the number of messages sent in the six scenarios. Times were found using the Common LISP *time* function. As we can see, our results follow the expected pattern above a threshold for communication of four.

The benefit per message for very low thresholds does not follow this pattern. At a threshold

of zero, all choices are communicated. This allows agents to keep there plans "in sync," adding appropriate choices to their plans immediately or modifying plans before a great deal of work has been done when there are discrepancies. At much higher thresholds, mostly more abstract schemas are communicated because they have more potential variable bindings and because there are more alternatives at this level of abstraction. The agents agree on the plan at the level at which communication takes place, but may not agree at more detailed levels of the plan. Here, again, communication pays off because it can be easily incorporated into the receiver's plan. At thresholds of two and four, a great many details are sent. However, enough are not sent to keep the plans completely in sync. Since the plans may become quite different at the level of communication, agents have difficulty integrating messages into their existing plans. This raises the overhead of communication and lowers the benefit per message. We are investigating this problem to determine if it is simply an artifact of eliminating some messages or if we can design heuristics to help agents synchronize their plans and communicate at the proper level of detail.

It is also important that beneficial heuristics and usefulness calculations can be developed which do not require a great deal of computational effort beyond that already needed to perform the collaborative task. We timed the calculations performed solely to calculate heuristics and usefulness ratings. Some of these calculations consumed so little time that the time function reported 0 msecs. The total reported time for calculating heuristic values and usefulness ratings was 0.2%of the total time and 4.7%of the time spent for communication.

4 Discussion

4.1 Integrating Our Approach with Existing Methods

Many mechanisms for selecting information to communicate already exist. However, most of these rely on knowledge or computational resources which may not be available to agents in a DAI system. Some require an agent to reason extensively about the value of a particular information item (e.g., 1,5). These techniques give the agents the flexibility to determine the value of the information in the current problem solving context. However, this flexibility comes at the price of extensive reasoning. Other techniques provide schemas which specify information to communicate (e.g., 3,6). These reduce processing time but require knowledge of specific scenarios for communication which may arise during collaborative problem solving. To overcome the lack of complete, *a priori* knowledge of the communication scenarios, and to maintain flexibility, schemas may include choice points or optional steps. However, to choose the option which should be taken, an agent is once again faced with the question of what to communicate.

Although the knowledge and resources to support existing techniques may not always be available, we should take advantage of them when they are. Sometimes the resources required to select information are needed to support the paradigm for communication used by the system. Therefore, we do not increase the time to perform the collaborative task if we take advantage of it to select information.

We envision usefulness ratings filling gaps left when other techniques do not have the resources needed to select information. For this reason, we plan to integrate usefulness ratings into our architecture for communication (9). Our architecture consists of a discourse manager which uses schemas to form a coherent conversation. The conversation is motivated by communication goals from the problem solver. When the discourse manager receives a goal, it fits a schema to achieve the goal into its current plan for the discourse.

Usefulness ratings will be used in our architecture in two ways. First, a goal to communicate the information will be generated when the usefulness rating of an information item passes the threshold

for communication. Usefulness ratings could be added to select information in any system which uses goal-motivated communication, including systems based on *speech act theory* (2,7). Once a schema has been added to the plan for discourse, regardless of the motivating goal, the system must decide which of its optional steps will be included in the dialogue. The dialogue manager can find the usefulness ratings of information which might be included and add information items to the dialogue if their usefulness rating exceeds a lower threshold. This threshold can be lower than the threshold for creating a goal to communicate information since less work is required to process additional information in a schema which is already included in the discourse. This method could be used to choose optional steps in any schema-based approach for communication.

4.2 Systems Which Can Benefit from Our Approach

We would expect our approach to be most beneficial for *homogeneous systems* because the reasoning of the sender would be similar to that of the receiver. The system can be expanded to heterogeneous systems, however, by creating heuristics which reflect the difference in the systems. For example, consider a system with one computer and one human. The computer may have a heuristic to report the result of lengthy calculations since, even though these calculations were not difficult for the computer, they will be difficult for the human. Such heuristics will require a model of the receiver, and, in heterogeneous systems with many types of agents, may require several ways of calculating usefulness ratings so that each type of agent is represented. This will require knowledge of other agents' reasoning, which may not be available. It will also require additional computational effort since the individual heuristics may take longer to compute and more values may be needed per information item. Consequently, the approach may not be appropriate for heterogeneous systems when agents (or system designers) know very little about the reasoning processes of others or when an agent must communicate with many different types of agents.

For now, we believe our approach is best suited for homogeneous systems which are in situations where their knowledge or computational resources are limited. Our approach will allow agents to select appropriate information in situations where little context is shared between agents. It also can fill gaps in established paradigms for selecting information.

4.3 Conclusions and Future Work

The results from our preliminary work are promising enough to warrant further work on this approach. In the near future, we will be evaluating the specific heuristics and usefulness ratings suggested for NONLIN and adding new heuristics and message types. We are currently developing heuristics to help agents agree on their plans and to allow agents to plan together even if they have different knowledge of the world. In the more distant future, we hope to extend our work to other kinds of planners in homogeneous systems. This work should allow us to find general guidelines for creating heuristics. Ultimately, we hope to implement and test our approach in systems of heterogeneous agents.

5 Acknowledgements

Thanks to Laura Donia and the CDPS Group at the University of New Hampshire for many helpful discussions. Thanks also to Martin Dempsey for the software to support communication through UNIX ports, to Laura Donia for help in implementing the heuristics in NONLIN, and to Scott Shaw for help in creating the original plans for a night out. Also, thanks to Roy Turner for helpful

comments on earlier drafts of this paper. We would also like to thank the developers of the Common LISP implementation of NONLIN for making it available over the Internet.

6 References

(1) J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143-178, 1980.

(2) Philip R. Cohen and C. Raymond Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177-212, 1979.

(3) Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63-109, 1983.

(4) Subrata Ghosh, James Hendler, Subbarao Kambhampati, and Brian Kettler. NONLIN Common Lisp implementation (v1.2): User manual. Computer Science Dept., University of Maryland, February 1992.

(5) Piotr J. Gmytrasiewicz and Jeffrey S. Rosenschein. The utility of embedded knowledge-oriented actions. In *Proceedings of the 1993 Distributed AI Workshop*, pages 155-169, Hidden Valley, Pennsylvania,

1.

(6) Kathleen R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, New York, 1985.

(7) J. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge, UK, 1969.

(8) A. Tate. Generating project networks. In *Proceedings of IJCAI-77*, pages 888-893, 1977.

(9) Elise H. Turner. Organizing dialogue from an incoherent stream of goals. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING-92)*, pages 338-344, Nantes, France,

1.

(10) Elise H. Turner. Exploiting problem solving to select information to include in dialogues between cooperating agents. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 882-886, Atlanta, Georgia, 1994.

(11) Elise H. Turner. Selecting information to communicate. In *Working notes for the AAAI Workshop on Planning for Interagent Communication*, pages 47-52, Seattle, Washington, 1994.