Preliminary Report on GENERIC NEGOTIATOR

Dajun Zeng
zeng+@cs.cmu.edu
(412)268-8825

Katia P. Sycara
katia+@cs.cmu.edu
(412)268-8815

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

1 Introduction

Coordination among agents (either human or software agent) to resolve conflicting interests is essential in distributed systems. Negotiation, where agents exchange information, reason about other's preferences, and estimate the effect of communication over other's behavior, can resolve the conflicts in a decentralized fashion. The goal of negotiation tasks involves reaching consensus among negotiating participants, while each participant's personal/local preferences should be reflected and respected in the resulting compromises.

Building Artificial Intelligence-oriented systems as negotiation support tools has attracted much attentions recently [1, 2, 3]. Rather than using negotiation as a means to resolve conflicts and foster agreements among multiple agents, we are interested in investigating negotiation as a domain in its own right, characterizing the process and answering research questions, such as "what is and what isn't generic about the negotiation process?" and "what aspects of negotiation are domain independent, and what aspects are domain dependent?". To this end, we are developing a generic negotiator that functions on behalf of human users to perform reasonable (or human-like) negotiating process, potentially for a wide range of application domains. The implemented system, called GENERIC NEGOTIATOR, has been tested in meeting scheduling application. Distributed meeting scheduling has been thoroughly discussed by Sen and Durfee [4]. Our work differs from theirs in that we are more concerned about the *generic* part of negotiation strategies and the *reactiveness* of the system, such as handling cancellations.

2 Negotiation Representation and Strategies

In order to automate negotiation processes, an automated negotiation agent needs to be able to: (1) communicate with other agents to perform negotiation-related information query/answer (such as personal preference information) and to exchange proposals, counter proposals, commitments, etc. (2) generate an initial proposal to start a negotiating process, (3) receive feedback about the quality of its proposal, evaluate it, and use it to modify the proposal or construct a new one, (4) take into consideration the changing negotiation context, (5) change the belief structure of other agents, and (6) determine that a admissible/satisfactory agreement has been achieved (termination criteria).

The following representation (in a frame-like language) is utilized to describe a negotiation scenario in GENERIC NEGOTIATOR.

```
(defframe negotiation ()
    ((interaction-topic) ;; e.g. bargaining-for-wage-increase
     (negotiation-parties)
     (negotiation-issues) ;;  a list of issues under current
             ;; negotiation. e.g., for meeting-scheduling,
             ;; it may take the value
             ;;   (meeting-start-time meeting-duration)
     (negotiation-contexts)
     ))

(defframe negotiation-issue ()
    ((issue-name)
     (values) ;; possible issue values
     (dividable?) ;; if viewing all issues are sort of resources,
             ;; we can categorize these issues into two major classes:
             ;;(1) dividable:  if we split this resource into
             ;; two parts, it won't lose its entity
             ;; and usage. e.g. time interval is dividable
             ;; (2) undividable: this type of resource will lose
             ;; its entity if we separate it in some way.
     (constraint-type) ;; hard-constrained or relaxable
     ))
```

A particular negotiation process will be an instantiation of the negotiation frame which we described above. Communications among agents are performed through exchanging messages. We show below a negotiation protocol which seems to be generic enough to be used to deal with a number of negotiation tasks and serves as the basis of GENERIC NEGOTIATOR: (1) after receiving a negotiation-related message, the negotiator should situate the message first, in other words, find negotiating history related to the interaction topic specified in the in-coming message, (2) based on negotiation history and the new in-coming message, the negotiator will invoke the evaluation module to determine if it is necessary to continue negotiation process (termination criteria), (3) if the negotiation process is deemed to be able to continue, the negotiator will invoke the domain-dependent module to evaluate the in-coming message and the current state of the world (for

instance, the user's calendar in meeting-scheduling domain) to see if all the hard constraints have been satisfied. If so, it means that *feasible* solutions exist and the negotiator will choose among them the best in terms of negotiation participants' preferences, (4) in the case of absence of any solution which satisfies all constraints, the negotiator will try to relax some of the constraints to work out some feasible solution(s), unless the negotiator finds that it is worthless relaxing any constraints and decides to withdraw from the negotiation (e.g., in meeting-scheduling, it might be useless to try to schedule a not-so-important meeting in a heavily constrained time-table). After the relaxation strategy is determined, the domain-dependent module will be invoked to solve the problem locally and come up with the counter-solution (counter-proposal).

Although in particular negotiation processes, negotiation strategies seem to be quite varied, domain-specific and situation-dependent, still there exists negotiation strategies which are domain independent and are able to provide valuable advice to direct the negotiating process towards efficient convergence. There are mainly two types of negotiation strategies considered in GENERIC NEGOTIATOR associated with the following decision making tasks:

1. When multiple feasible solutions (all hard constraints are satisfied) exist, how to choose one or some (or maybe all of them) to propose to other negotiation parties? Specifically, the negotiator could: (1) only propose the most preferable one from the viewpoint of the user/patron of itself, (2) propose all the feasible solutions associated with user's preferences, (3) propose all the feasible solutions without revealing user preferences, (4) propose a subset of the available solutions

2. When no feasible solution exists, how to choose some constraints to relax? Some of the relaxation strategies for the negotiator could be to: (1) simply do nothing to get other participants into relaxing their own constraints, (2) only relax one of the over-constrained issues, (4) propose a *package* solution through relaxing several negotiation issues at the same time while considering potential trade-offs between them, etc.

3 Message Types for Communication

In the GENERIC NEGOTIATOR system, different types of messages are defined to facilitate communication between distributed agents:

un-commitment cancelling a previously committed event

commitment agreeing on the proposed issue values (e.g., a particular meeting time)

proposal proposing values for issues under negotiation

rejection rejecting the proposed issue values

information-request asking the involved agents about the information relevant to the current negotiation (such as other user's personal calendar)

information-reply replying to information requests

action-request initiating a negotiation process.

4 Scenario: Distributed Meeting Scheduling

Distributed Meeting Scheduling was chosen as one of the application domains to test the GENERIC NEGOTIATOR system. In our distributed meeting scheduling scenario, we assume that every user will have a meeting scheduling agent, which manages his/her calendar, and a negotiator agent, which exchanges information over a computer network via Electronic-mail with other negotiators representing other users to schedule meetings. We view meeting scheduling problem as a resource allocation task. Specifically, each user's time-line is represented as a resource. The capacity of this resource (in terms of a specified time-interval) is explicitly expressed in the following way: 0 means idleness and 1 means the corresponding time interval has been occupied. Each meeting instance is considered as an activity which requires to be scheduled on all related attendees' time-tables. We don't allow that the resource capacity exceeds 1, meaning that the user couldn't show up for two different meetings at the same time.

A simplified description of implemented negotiation protocols is given as follows:

1. When a meeting needs to be scheduled, the meeting initiator's negotiation agent (called host hereafter) tries to find out free time intervals in the initiator's schedule. Then it will announce the meeting to other involved agents by proposing one or more of the feasible (preferable) meeting times, depending on the negotiation strategy it has chosen.

2. Each participant receives the announcement proposal(s), tries to find local solutions based on its own personal calendar and sends them back to the host in the form of counter-proposal. The time intervals suggested in the counter-proposal can simply be the subset of those proposed by the host, or they can be completely different.

3. The host collects and evaluated these counter-proposals. If the counter-proposals suggest a common time interval, the meeting can be scheduled and the host will send out the confirmation message. If a mutual agreement cannot be reached, the host generates new proposals (maybe through relaxing some constraints) and sends them out.

4. Other participants will record the resulting schedule if they get confirmation information. If the time intervals they proposed before become unavailable, they will send back rejections, which may entail another cycle of negotiating process.

The above steps are repeated until a satisfactory schedule is achieved or it is recognized heuristically that the meeting cannot be scheduled (due to the over-constraintedness or missing of the meeting due date).

5 Research Issues and Future Work

Interesting research questions with the architectural design of a society of software agents with negotiation capability is how to balance the workload of problem solvings between the negotiator and other domain-specific software modules in such a way that the task can be done effectively as a whole. Particularly we are exploring these issues including:

- what kind of knowledge should the negotiator know about the domain? e.g., should it be able to evaluate the solutions given by other domain-specific software agents in terms of user preferences?

- what kind of knowledge should domain-specific software agents have to make it possible to participate in the negotiation effectively?

- how to construct an effective interface between a generic negotiator and the domain-specific-software-agent it is associated with?

Currently we are designing a set of experiments to test GENERIC NEGOTIATOR with respect to problem solving efficiency, and solution quality.

We are also planning to test the GENERIC NEGOTIATOR system in other application domains, such as *pursuit game*.

References

[1] J. Rosenschein and G. Zlotkin, *Rules of Encounter*. Cambridge, Mass.: MIT Press, 1994.

[2] K. Sycara-Cyranski, "Arguments of persuasion in labor mediation," in *Proceedings of the Ninth International Conference on Artificial Intelligence (IJCAI-85)*, (Los Angeles, CA), pp. 294–296, 1985.

[3] K. Sycara, "Resolving goal conflicts via negotiation," in *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, (St. Paul, MN), 1988.

[4] S. Sen and E. H. Durfee, "The effects of search bias on flexibility in distributed scheduling," in *Proceedings of the Twelfth International Workshop on DAI*, 1993.