AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems, 2001

Agent-Based Argumentative Negotiations with Case-Based Reasoning

Leen-Kiat Soh and Costas Tsatsoulis
Information and Telecommunication Technology Center (ITTC)
Department of Electrical Engineering and Computer Science
University of Kansas
2335 Irving Hill Road
Lawrence, KS 66045
{lksoh, tsatsoul}@ittc.ukans.edu

ABSTRACT

In domains of limited resources, problem solving and execution of solutions may require the satisfaction of resource use constraints among a group of collaborating agents. One way for the agents to agree to the distribution of limited resources is through *negotiation.* In this paper we present how an agent that decides it must negotiate for the use of resources that it needs to reason or execute a task can use case-based reasoning (CBR) and utility to learn, select, and apply negotiation strategies. The negotiation process is situated in the current world description, self state, and also dynamically changing evaluation criteria and constraints. Consequently, the negotiation strategies that an agent uses vary greatly. To determine the negotiation strategy an agent uses CBR to compare the new situation to old cases (from its situated case base) and learns from its previous experiences how it should negotiate. This unique synergy allows us to address real-time resource allocation and efficient knowledge management: (1) the use of negotiation reduces communication traffic since knowledge updates and exchanges are performed only when necessary, and (2) the use of CBR and utility streamlines the decision process so that an agent can obtain a "good-enough, soon-enough" negotiation strategy effectively.

1. INTRODUCTION

We have developed a multiagent system that uses negotiation between agents to reason in a domain of limited resources. In our work we use case-based reasoning (CBR) to select, apply, and learn the negotiation strategies that the agents use. The agents control sensors in an environment with multiple targets, and the goal of the agents is to coordinate their activities and collaborate to achieve errorless tracking of as many targets as possible. Since there are more targets than sensors, and since a sensor cannot track a target through the whole physical space, the agents need to request use of the sensing resources (such as battery power, sensor mode, length of sensing, type of sensing beam, etc.) of other agents. At the same time, the target tracking needs to be performed in real time, and agents must negotiate for the use of system resources of the physical computing platform on which they are situated. We are assuming agents that are collaborative, but also require that their own tasks be completed if they are—in the agent's opinion—of higher-priority. The agents negotiate to convince other agents to assist in target tracking or to surrender resources. Given the task description and the current status of the world and the agent itself, how an agent goes about its negotiation may differ. A negotiation process involves knowledge dissemination and transfer towards conflict resolution. To improve its efficiency, an agent has to be aware of the information it passes to its counterpart, including the information's volume and relevance. Moreover, an agent has to know when to abort a negotiation, when to accept a task, and most importantly, how to behave during a negotiation. Our approach is to use case-based reasoning to learn agent negotiation strategies. A case is a semantically rich representation of a negotiation previously experienced. It contains an agent's view of the world, its negotiation partner(s), and itself. Based on this knowledge, an agent derives a set of negotiation parameters that determine the negotiation behavior. Finally, a case contains the outcome of the negotiation, including various aspects of the negotiation such as the elapsed time and the number of interactions. The lesson learned from each negotiation is thus encoded in each case.

Approaching our task and resource allocation problem with negotiation has several advantages. We do away with a centralized knowledge (or information) facility that requires constant updates and polling from agents. Instead, each agent maintains its own situated knowledge of the world, increasing its autonomy. Since knowledge is shared when necessary, there is less communication traffic. In addition, knowledge inconsistencies are resolved in a task-driven manner, making the knowledge management easier. Similarly, the use of CBR benefits our design. It allows an agent to learn a set of "good-enough, soon-enough" negotiation strategies by referring to old, similar cases. CBR also greatly limits the time needed to decide on a negotiation strategy, which is necessary in our real-time domain. Thus, it is essential for the negotiator to be able to constantly maintain a usable strategy while refining it with case adaptation.

In this paper, we first discuss some related work in agent-based negotiations. Then we describe our agent architecture and behavior. In the fourth section, we present our case-based reasoning system for learning, selecting, and applying agent negotiation strategies. In Section 5, we show some results of our design. Finally, we conclude the paper.

2. BACKGROUND

Negotiation can be used by agents to perform problem solving and to achieve coherent behavior in a multiagent system. An agent can negotiate in a non-cooperative environment to compete with each other for rewards [5, 7]. Agents can also negotiate in fully prescribed manner where the negotiating parties know exactly what each other's cost and utility functions are, or when such knowledge is learned during the first step of interaction in a negotiation [4, 8]. There are agents that negotiate using the unified negotiation protocol (UNP) in worth-, state-, and task-driven domains where agents look for mutually beneficial deals to perform task distribution [13, 15, 16]. There are agents that negotiate via a blackboard architecture in which agents post their demands and service on a common site. Depending on the responses an agent receives, agents can relax their demands or enhance their services to attract a partner in cooperative distributed problem solving [2, 10]. There are agents that conduct argumentation-based negotiation in which an agent sends over its inference rules to its neighbor to demonstrate the soundness of its arguments [3, 11]. A closely related work uses negotiation among agents for case-based retrieval [12] in which agents exchange constraints to help retrieve a partial case that satisfies the requirements of different agents; the partial case is then used to solve diagnostic problems, for example. Finally, there are agents that incorporate AI techniques [1, 9, 14] and logical models [6] into negotiation.

Our negotiation model is argumentative. The initiating agent sends over evidential support to the responding agent when they argue. The responding agent computes (when it performs conflict resolution between its own perception and that of its counterpart) the utility and certainty of the information against a task- and situation-specific acceptance level. The responding agent may ask for further information from the initiating agent. This iterative negotiation process continues until (1) the time frame for the negotiation runs out, (2) a failure occurs (when the responding agent has exhausted all its counter-requests and the initiating agent still does not fulfil the requirement), (3) one of the agents aborts the negotiation (due to resource scarcity since our agents are multi-tasked), or (4) a deal occurs (when the responding agent accepts the evidential support).

3. Agent Architecture and Behavior

In our design, each agent has 2+*N* threads that can process tasks concurrently. Each agent has three basic processing threads: (1) a core thread that carries out functions of the agent such as housekeeping, managing tasks, reasoning, (2) a communication thread that checks the mail box for incoming messages and sends out messages to other agents, and (3) an execution thread that interacts with radar (either the actual physical hardware or the software simulation) to perform target tracking, sensor calibration, and target searching. Each agent has *N* additional negotiation threads. Each of these negotiation threads can initiate or respond to negotiation requests. The negotiation threads are spawned during agent initialization and remain inactive if there are no negotiation tasks. Figure 1 shows the basic thread architecture of our agent design.

Each agent has a set of managers:

(1) The Profile Manager keeps track of the current status of the agent, its neighbors and the target that the agent is tracking.
(2) The Communication Manager handles the communication channel for send and receive messages and maintains a local message mailbox (or queue).
(3) The Task Manager monitors the tasks that the agent is currently performing (such as negotiations, target tracking, etc.).
(4) The Reasoner performs feasibility studies and evaluates potential negotiation partners for dynamic coalition formation.
(5) The Negotiation Manager handles the activation of negotiation threads and monitors the status of the negotiations.
(6) The Execution Manager handles the job queue and monitors the execution of radar-related tasks.
(7) The Real-Time Manager interfaces with a kernel-based Real-Time Scheduling Service (RTSS) to perform timer announcement, CPU allocation, and other real-time activities with the operating system.
(8) The CBR Manager performs case-based reasoning to retrieve most similar cases for negotiation strategies and maintains the case bases.
(9) The Radar Manager models the environment of sensors and interacts with the sensor hardware and communication links (RF-links).

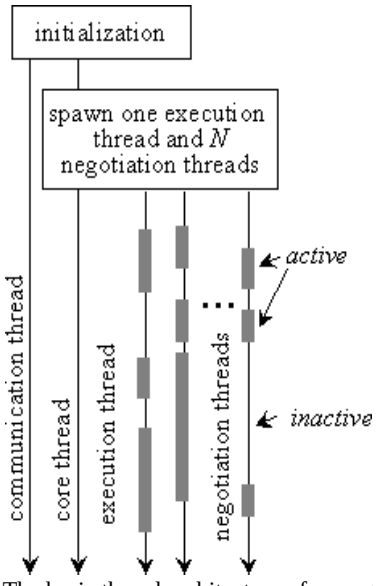

Figure 1 The basic thread architecture of our agent design

When an agent is idle, its core thread performs radar calibration and also a search-and-detect procedure. If a target is detected, the agent has to carry out a series of actions. First, it creates a tracking task and adds it to the Task Manager. Second, it asks the Radar Manager to measure the target to obtain its location and velocity. Third, based on the location and velocity, it obtains a sensor-sector list from the Radar Manager[1]. This list specifies all sensor-sector pairs whose sensing areas are covering the target or will be covering the target. Fourth, the agent then determines the sampling size of the sensor via the Reasoner. Now, if the sensor-sector list is not empty, then, the Reasoner also analyzes the potential partners (or neighbors) that it wants to negotiate with (to ask them to turn on their sensors to help track the target). After that, it sorts the list based on the utility of the partners and approaches them orderly.

The partner analysis is based on the past relationship between the current agent and each of its neighbors. The utility of each neighbor's helpfulness is high when the current agent has had a high number of successful negotiations with it. Also, based on the trajectory of a target, the sensor-sector list may contain entries for a same sensor but with different sectors. This partner analysis also computes which sector of a sensor is more important to have turned on. This is based on the times that the target will be hitting the sectors—sectors with earlier arrival times have higher utilities.

When initiating negotiations, the agent first retrieves the best case from its case base through the CBR Manager. This best case gives the agent a set of negotiation strategies. Then the agent checks to see whether there is at least one idle negotiation thread. If there is, it creates a negotiation task. Then it is ready to activate a negotiation thread via the Negotiation Manager. It subsequently uploads the agent data collected from the Profile Manager (so the negotiation thread waking up can download the current status of the profile), and then signals the negotiation thread. If the activation is successful, the agent adds the negotiation task to the Task Manager. Currently, the agent will activate as many negotiation threads as possible for each target detected—if the sensor-sector list has more than one entry, the agent will try to contact all distinct sensors on that list as long as there are idle negotiation threads.

Our agents are cooperative and share a common global goal, i.e., to track as many targets as possible and as accurately as possible. However, each agent also attempts to conserve its power usage and its resources. Hence, if an agent A asks an agent B to perform a certain task, A must convince B to do so.

[1] Each agent controls a sensor that has three sensing sectors that can be activated separately.

4. CBR and Negotiation

A negotiation process is situated: for the same task, because of the differences in the world scenarios, constraints, evaluation criteria, information certainty and completeness, and agent status, an agent may adopt different negotiation strategies. A strategy established the types of information transmitted (i.e., argued about), the number of communication acts, the computing resources expended, and so on. To represent each situation, we use cases. Each agent maintains its own case base and learns from its own experiences.

Theoretically, CBR offers a set of possible negotiation strategies, all of them valid in the current context, but only one of them adequate to produce a "good enough, soon enough" solution. The negotiation environment defines the evaluation metrics and the real-time constraints. Multivalued utility theory relates satisfaction and preference criteria to the possible negotiation strategies and selects the one that optimizes these criteria while minimizing risky behavior by the autonomous negotiating agent. The result is a qualitative evaluation of all known, valid negotiation strategies so that the system can select the best one. After each negotiation transaction the agent reviews the results of the transaction and continues with the original negotiation plan or replans a new strategy, depending on whether the outcomes matched its predictions, and based on new information about the world that resolves previously uncertain or unknown information. The result is that known negotiation strategies are evaluated using the current criteria and constraints, and the agent effectively adapts its overall reasoning to the current situation.

4.1 Case Description

In our work a case contains the following information: (1) a description of the part of the world relevant to the case, (2) the current agent profile, and (3) the neighbor profiles, where a neighbor is defined as an agent with which one chooses to negotiate. The world description includes the target and sensor information such as orientation, speed, quality of recognition, and list of actions that an agent can perform to improve recognition (e.g. the data collection mode of the sensor, the duration of the collection, the number of measurements, and the quality of calibration). The agent profile keeps track of the current power of the sensor, the sensing sector that is active, the data collection mode, the data quality, the status of the receiver and the transmitter, the communication channels, the list of current tasks that the agent is handling, and the knowledge of other agents. The neighbor profile contains the viewpoint by the agent of its neighbors, including the neighbors' attributes such as capability, location, and responsibility, and perceived attributes such as friendliness, helpfulness, efficiency, etc. Finally, a case records the outcome of a

negotiation; if the negotiation failed, a case contains a coded reason.

4.2 Determining Negotiation Strategies

When an agent decides that it needs help to complete a task, it locates another agent and negotiates with that *neighbor* to use some of its resources. Before a negotiation can take place, an agent has to define its negotiation strategy, which it derives from the retrieved, most similar old case.

4.2.1 Description and Strategies of Initiating and Responding Cases

Each agent maintains two case bases—one for the cases when the agent was the initiator of a negotiation, and one when the agent was the responder to a negotiation. In general, an initiator is more conceding and agreeable and a responder is more demanding and unyielding.

In an initiator case, the case descriptors are the list of current tasks, what the potential negotiation partners are, the task description, and the target speed and location. The negotiation parameters are the classes and descriptions of the information to transfer, the time constraints, the number of negotiation steps planned, the CPU resource usage, the number of steps possible, the CPU resources needed, and the number of agents that can be contacted.

In a responder case, the relevant attribute set consists of the list of current tasks, the ID of the initiating agent, the task description, the power and data quality of its sensor, its CPU resources available, and the status of its sensing sector. The negotiation parameters to be determined are the time allocated, the number of negotiation steps planned, the CPU usage, the power usage, a persuasion threshold for turning a sensing sector on (performing frequency or amplitude measurements), giving up CPU resources, or sharing communication channels.

4.2.2 Arguments and Persuasion Threshold

The agents use an *argumentative* negotiation process, where the initiating agent provides its negotiation partner with arguments that are supposed to convince it to give up use of some of its resources. Depending on the situation, each resource has a *persuasion threshold* associated with it. The negotiating agent must provide enough arguments to convince its negotiation partner beyond its persuasion threshold for a specific resource.

To illustrate, suppose a responding agent is trying to determine its persuasion threshold for turning a sensing sector on—i.e., the strength of the arguments provided by the initiating agent must be higher than this threshold before the responding agent agrees to oblige to the request. Thus, if the agent is currently engaged in many tasks or some top-priority tasks, then the agent sets a high threshold, and vice versa. If, after a review of the task description provided by the initiating agent, the agent finds that the task negotiated for is of low importance, then the agent sets a high threshold, and vice versa. For example, if the battery power of the sensor controlled by the agent is low, then the agent may set a high threshold, and demand to be convinced very strongly to use this limited, consumable resource to satisfy some other agent's needs. If, on the other hand, the agent currently has an active sensing sector covering the zone of the new task, then the agent sets a low persuasion threshold.

In another example, suppose an initiating agent is trying to determine its negotiation strategy. In general, if the agent is currently managing many tasks, then it favors a short time period for the negotiation, and a small number of interaction steps. If the target's speed is high, the agent also favors short and less cumbersome negotiations. If the task is complicated, the agent prefers a neighbor that has a lot of available CPU resources.

The negotiation strategies are implicitly embedded in the cases and adapted to the current situation in real-time so that agents can argue effectively in a negotiation.

4.2.3 Discrete and Continuous Requests

We deal with two types of requests in our negotiations: discrete and continuous.

Discrete requests are such as turning on a sector, performing amplitude or frequency measurements, and giving up communication channels. The responding agent either agrees to the request to perform such a task or simply rejects.

Continuous requests are such as giving up CPU resources. The responding agent has the flexibility of counter-offering the initiating agent how much CPU resource that it is willing to give up. To do so, the responding agent first computes the maximum CPU resource that it is willing to give up via the Reasoner. Then, from the best case, it obtains the following: (1) linear or exponential persuasion function, (2) beta – a conceding factor for giving up CPU resource, affecting the persuasion function, and (3) kappa – a willingness factor for giving up CPU resource, affecting the persuasion function. The y-axis of the persuasion function is the amount of CPU resource and the x-axis is the evidence support (which will be discussed in Section 4.2.4). If the responding agent is busy, it prefers an exponential to a linear persuasion function since it can expect a short negotiation with the former. If the responding agent is operating close to its allocated CPU resource, then it prefers a linear function to not upset its CPU usage in the short term. The willingness factor determines the y-axis crossing point of the persuasion function—specifying how much CPU resource that the agent is readily willing to give up. The conceding factor, on the other hand, determines the slope of the persuasion function. The larger the slope, the less resolution the counter-offers are that the responding agent plans to make. At each negotiation step, the responding agent refers to the evidence support that the initiating agent has provided, and

uses the persuasion function to make a counter-offer. If, for example, the initiating agent already obtains a large portion of the CPU it needs from another agent, then it may want to accept the counter-offer and complete the negotiation. If the initiating agent realizes that it is running out of time and is in need of CPU immediately, then it may want to accept the counter-offer and move on. Otherwise, the initiating agent continues to argue to convince the responding agent to give up more.

4.2.4 Ranking of Information Classes and Evidence Support

During a negotiation, the initiating agent sends over information pieces to the responding agent. The responding agent computes the evidence support of the information pieces. If the support is greater than the persuasion threshold (derived from the best case), then the responding agent agrees to perform the requested task. If, however, the initiating agent has exhausted all its information pieces and still fails to garner enough evidence support to convince the responding agent, then the responding agent either rejects the requested task (as in a discrete-resource case) or makes a final counter-offer (as in a continuous-resource case).

During a negotiation, each agent attempts to minimize the number of messages sent and the length of the messages sent. With fewer messages sent, the agents can avoid message loss due to communication failures, and reduce traffic among the agents. The agents want to send short messages as well since the transfer is faster and the bandwidth is constrained. Thus, it is important for an initiating agent to decide which information pieces are more important to send to the responding agent.

In our current design, there are three pieces of information: (1) *self* – the current status of the agent and its sensor, (2) *target* – the current status of the target, and (3) *world* – the current status of the world and the view of the neighbors. The ranking of these pieces of information is derived from the best case, retrieved from the case base. In general, if the target has a high speed, the *target* information is given more priority. If the initiating agent is busy (performing many tasks), then the *self* information is ranked higher. If the initiating agent has only one neighbor to negotiate with (thus making the negotiation more critical), then the *world* information is more important. After case adaptation, the initiating agent has the ranking of the information pieces. Subsequently, at each negotiation step, the initiating agent sends the information pieces over orderly to the responding agent until a deal is met or there is not any more information left.

On the other hand, the responding agent uses a *relaxed* constraint-satisfaction approach to compute the evidence support of the initiating agent's arguments (or information pieces). The constraints are (1) bilateral beliefs that need to be reconciled, (2) uni-lateral beliefs that are single-value or multi-valued with conjunctive or disjunctive relations. The satisfaction is relaxed such that partial satisfaction is allowed and the support is a continuous value. For example, if the initiating agent informs the responding agent that it has been helpful in the past to the responding agent's requests and the responding agent checks its record and finds out it is true, then the responding agent is more ready to agree to the negotiation.

We are also looking into an evidential reasoning approach in which the responding agent is able to pinpoint the exact pieces of information that it needs from the initiating agent. Each piece of information is also scored with the points that it would add to the evidence support. Then, the responding agent can inform the initiating agent to supply which pieces of information first. This way, the initiating and the responding agents can cooperate closely towards achieving a deal more efficiently and effectively.

4.3 Case Selection and Retrieval

The CBR Manager evaluates cases using weighted matching and different matching functions for different attributes. After evaluation, the most similar cases will be selected. However, if there are more than one case with the same similarity score, the CBR Manager then compares the outcome of the negotiations and selects the case with the best outcome. We are currently working on using a utility-based evaluation, where (1) the elapsed time of the negotiation, including the number of steps and the volume of messages communicated, (2) the quality of the negotiation, and (3) the number of strategy changes during the negotiation are also used

4.4 Case Adaptation

Given the set of negotiation strategies from the best case, the CBR Manager adapts the parameters based on the difference between the new case and the best case, and also on the outcome of the best case. Since each case is situated, the set of negotiation parameters learned from the best case might no be applicable in the current case. Hence, the CBR Manager modifies the parameters based on the differences. For example, if the current target has a higher speed than the old target of the best case, then we allocate less time to conduct the negotiation. If the agent is performing more tasks currently than it was when the best case happened, then we want to use less CPU resources. Furthermore, the CBR Manager modifies the parameters based on the outcome of the best case. If the negotiation of the best case failed and it was because of the negotiation running out of the allocated time, then we plan for more time. If the negotiation failed due to lack of CPU resource, then we ask for more CPU. In this manner, we are able to learn from our previous experiences a "good-enough, soon-enough" set of negotiation strategies and how to avoid repeating past failures.

4.5 Case Storage and Learning

After a negotiation is completed (successfully or otherwise), the agent delegates the case to the CBR Manager for storage and learning. In our case storage design, we perform two types of learning: incremental and refinement, as shown in Figure 2.

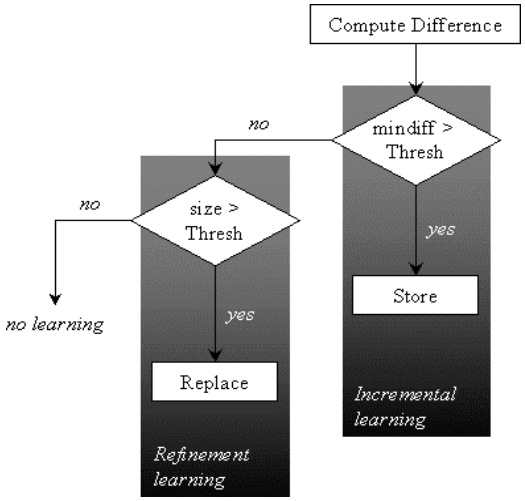

Figure 2 Case-based learning in our agents. First we compute the difference between the new case and the existing casebase. If the minimum difference between the new case and any case in the casebase is greater than a pre-determined threshold, then we store the case. Otherwise, if the size of the casebase is greater than another pre-determined threshold, then we replace the least different case already in the casebase with the new case.

First, we perform incremental learning. We match the new case to all cases in the case base and if it is significantly different from all other existing cases in the case base, then we store the case in the case base. When we compute the difference between a pair of cases, we emphasize more on the case description than the negotiation parameters since our objective here is to learn a wide coverage of the problem domain. This will improve our case retrieval and case adaptation. So, we learn good, unique cases incrementally.

Second, since we want to keep the size of the case base under control, especially for speed in retrieval and maintenance (since our problem domain deals with real-time target tracking), we also perform refinement learning. If the new case is found to be very similar to one of the existing cases, then we compute (1) the sum of differences between that old case and the entire case base (minus the old case) and (2) the sum of differences between the new case and the entire case base (minus the old case). If the second sum is greater than the first sum, then we replace the old case with the new case. In this manner, we are able to increase the diversity of the case base while keeping the size of the case base under control. As a result, we gradually refine the cases in the case base.

5. RESULTS

We have built a fully-integrated multiagent system with agents performing end-to-end behavior. In our simulation, we have four autonomous agents and a Tracker module. The Tracker module is tasked to accept target measurements from the agents and predict the location of the target. The agents track targets, and negotiate with each other. We have tested our agents in a simulated environment and also with actual hardware package of 4 sensors and 1 target. For the following experiments, there was only one target moving about 50 feet between two points.

5.1 Different Negotiation Behaviors

One of the experiments that we conducted was on the effects of different agent behaviors on negotiations: (1) Behavior I – agents only track but do not negotiate, (2) Behavior II – agents negotiate using a reduced set of cases, and (3) Behavior III – agents negotiate using all cases.

Figure 3 graphs the number of messages sent per agent cycle for each type of agent behavior. Behavior I did not have any sendMessage events since agents could not negotiate. And as a result, it actually had more target measurements as it had the highest number of sendTracker (sending a target measurement to the Tracker module). Agents using the reduced case bases negotiated a bit more than those using the full case bases. This is probably because the negotiation strategies derived from the reduced case bases were less efficient than those derived from the full case base, since the latter covered a wider domain space. Incidentally, as a result of more efficient negotiations, the agents performing behavior III were able to conduct more target measurements than those performing behavior II. Thus, this indirectly shows that agents are able to utilize the case-based reasoning to learn negotiation strategies to negotiate more efficiently, and to track a target more frequently.

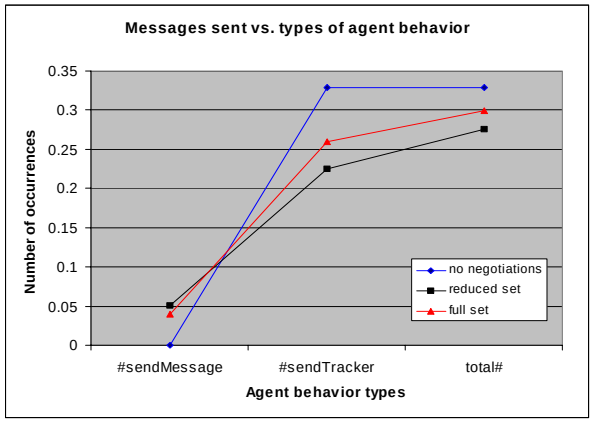

Figure 3 Number of messages sent per agent cycle vs. types of agent behavior

Figures 4 and 5 illustrate that the agents are able to learn negotiation strategy using case-based reasoning to negotiate more successfully and to track targets more accurately. Figure 4 shows the percentage of successful negotiations vs. different types of case bases. The 'to' column denotes the success rate of an initiating agent; the 'from' column denotes that of a responding agent. As we can see from the graph, since the negotiation strategies derived from the full set of case bases are more efficient, the agents were able to conduct more successful negotiations. Figure 5 shows the tracking accuracy of the different agent behaviors. The 'DX' column denotes the errors in foot along the x-axis; the 'DY' column denotes those along the y-axis. As we can see from the graph, without negotiations, the tracking accuracy was the worst. Agents negotiating using the full set of case bases gave the best tracking accuracy.

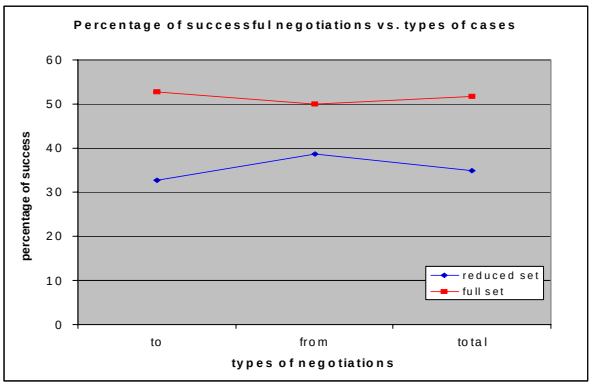

Figure 4 Percentage of successful negotiations vs. types of case bases

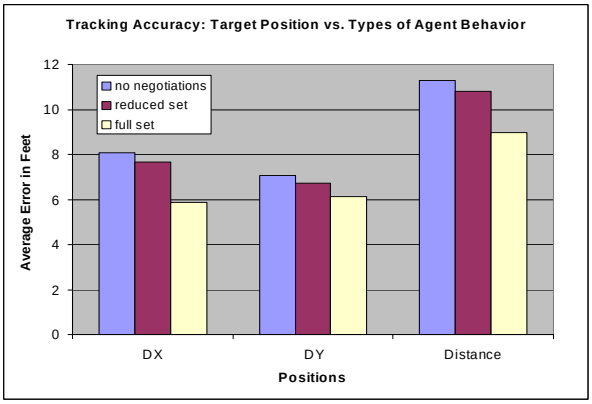

Figure 5 Tracking accuracy vs. different agent behaviors

Figure 6 shows the dynamic relationships among our agents. For example, agent 1 negotiated with agent2 about 17% of the time, with agent about 3 52%, and with agent4 about 31%.

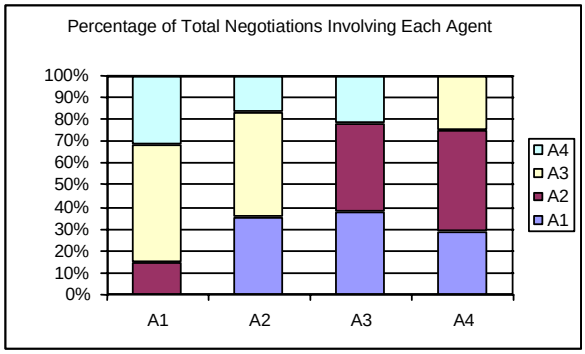

Figure 6 Percentage of total negotiations among agents

5.2 Negotiation Outcomes

Table 1 shows the relationships between outcome types and the average number of iterations, the average message volume processed, and the message length. This allows us to analyze how a negotiation succeeded or failed. Some interesting observations are:

(a) When an agent outright agrees to perform a requested task, or rejects to do so, or suffer a channel jammed or an out-of-time failure, the number of iterations is 1 or 2. This follows directly from the way our agents negotiate.

(b) When a rejection occurs (initiatingReject and respondingReject), the average number of iterations is very high (6 or 7.76, respectively). This is because before a responding agent rejects a request, it has to go through argumentative-negotiation with the initiating agent. The corresponding average message volume sent is also very high (850.8 or 1041.4, respectively).

(c) A channel jam occurs usually after 3.79 iterations for the initiating agent and after 4.41 iterations for the responding agent. This indicates that maybe we may want to complete our negotiations before 4 iterations. A channel jam could be due to message loss, channel jammed, or busy agent not responding.

(d) An out-of-time failure occurs usually after 3.17 iterations for the initiating agent and after 4.52 iterations for the responding agent. These two numbers are similar to those regarding channel jam failures.

(e) When a negotiation is successful, the average message volume is 553.7 for initiating and 577.9 for responding. So, in general, it is about 560 characters per successful negotiation. If it is an outright agreement, then it is around 250 characters. When a negotiation fails because of channel jam or out-of-time, the average message volume is between 454 and 629 characters. When a negotiation fails because of the initiating agent's failure to convince the responding agent, the average message is about 850 or 1041, for initiating and responding agents, respectively.

	count	ave. iter.	ave. msg. volume	ave. msg. length
Total	543	3.99	550.8	138.0
OutrightAgree	54	2	248.9	124.4
OutrightReject	41	2	217.7	108.9
OutrightJammed	60	1	117.7	117.7
OutrightOutOfTime	12	1	111.0	111.0
initiatingSuccess	33	3.73	553.7	148.4
initiatingReject	74	6	850.8	141.8
initiatingJammed	68	3.79	566.1	149.4
initiatingOutOfTime	35	3.17	454.6	143.4
respondingSuccess	34	4.09	577.9	141.3
respondingReject	74	7.76	1041.4	134.2
respondingJammed	37	4.41	619.9	140.6
respondingOutOfTime	21	4.52	629.0	139.2

Table 1 Outcomes and messages. outrightAgree means the responding agent agreed without negotiation; initiatingSuccess means the initiating agent came to an agreement with the responding agent after negotiation; and so on.

From the above experiment, we also conclude the following:

(a) A rejection after negotiation was expensive: it took an average number of iterations of about 6 and 7.76, for initiating and responding agents, correspondingly. It also took an average message volume of about 850 and 1040, for initiating and responding agents, respectively. The significance here is that we had many such occurrences—rejection after full negotiations.

(b) Overall, the reasons of failure were: rejected after negotiations (39.3%), out of time (18.4%), and channel jammed (42.3%) for responding agents; and rejected after negotiations (49.6%), out of time (11.0%), and channel jammed (38.8%) for initiating agents. This calls for a better design of the communication infrastructure and we are currently looking into building a socket-based communication object that better speed and latency.

(c) Successful negotiations were mostly achieved via outright agreement (65.06%).

(d) The percentage of success decreased as more arguments were sent. For example, when no information was sent, the success rate was 54/116 = 46.6%; when one piece of argument was sent, the rate was 19/67 = 28.4%; when there are two pieces, the rate was 4/30 = 13.3%; when there are three pieces, the rate was 6/109 = 5.5%. This means that the first argument carries the most weight in our domain problem where communication channel jam is common.

(e) As discussed in Section 4.2.4, arguments are divided into the current status of the world, the current status of the agent, and the current status of the target. In this experiment, we also measured the distribution of rules used in computing the evidence support for a responding agent. World-related heuristic rules were used most (90.5%), self-related heuristics came in second (77.7%), and target related heuristics came in last (54.0%). This information allows us to better design our heuristics to improve the negotiations.

6. FUTURE WORK

There are other CBR and negotiations issues that we plan to investigate. Of immediate concerns to us are the following.

(1) We plan to study the learned cases and the overall 'health' of the case bases. We plan to measure the diversity growth of each case base (initiating and responding) for each agent. We also plan to examine the relationships between an agent's behavior (such as number of tracking tasks, the number of negotiation tasks, the CPU resources allocated, etc.) with the agent's learning rate. For example, if an agent is always busy, does it learn more or learn less, or learn faster or more slowly? We also want to investigate when agents stop learning where no new cases are stored and the learned case bases of the agents are similar.

(2) Since we also negotiate for CPU resources, we plan to investigate whether dynamic resource scheduling (at the operating system level) as a result of multiagent negotiations leads to convergence or oscillations in CPU allocations among the agents sharing the same operating platform.

(3) We plan to expand our real-time profiling during negotiation. We want to reflect more changes in an agent at each negotiation step. One critical issue here is whether we describe each case with its real-time adjustments so that these adjustments can be applied to the current negotiation, or we build simple cases that the negotiation thread can retrieve to help deal with real-time situational changes. Tradeoffs are the complexity of the cases and speed in case-based reasoning. In general, we would like to have learning capabilities of real-time agent negotiation strategies.

(4) We plan to investigate the benefits of the 'good-enough, soon-enough' solutions, as opposed to a baseline multiagent approach in which all negotiation strategies are derived from scratch based on the information collected from the different managers of the agent. Then, we plan to evaluate the results (in tracking accuracy, in speed, in the number of messages exchanged, etc.) of the baseline approach against the CBR approach. The key question is, within how many microseconds, for example, can the CBR approach produce a set of usable negotiation strategies in time-critical situations as opposed to the baseline approach?

7. CONCLUSIONS

We have proposed a multiagent approach to distributed resource allocation and task allocation problems,

particularly to multi-sensor tracking of targets in a real-time environment. The approach integrates case-based reasoning to derive and learn negotiation strategies that are applicable to situated, real-time problems. Our negotiation protocol is argumentation-based in which the initiating agent attempts to persuade the responding agent to perform a task or give up a resource by iteratively supplying useful arguments.

We have described our agent architecture and behavior briefly and our CBR approach thoroughly. Our approach allows the agents to learn agent negotiation strategies based on previous experiences, adapt to the current situations, and avoid repeating past failures. Our CBR can also learn new cases to improve the diversity while maintaining the size of the case bases. More importantly, we have shown that CBR with good cases helped our agents to negotiate more efficiently and more successfully, and that indirectly helped our agents track their targets more frequently and more accurately. The CBR and negotiation synergy allows us to address real-time resource allocation and efficient knowledge management as we aim at (1) reducing communication traffic so that knowledge updates and exchanges are performed only when necessary and (2) improving the response time to time-critical events using "good-enough, soon-enough" negotiation strategies. We have also presented comprehensive analyses on the negotiation outcomes and the arguments used based on our experimental results.

8. ACKNOWLEDGMENTS

The authors would like to thank Kelly Corn, Will Dinkel, Jim Emery, Arun Gautam, Douglas Niehaus, Pete Prasad, and Huseyin Sevay for their work on the ANTS Project at the University of Kansas. The work described in this paper is sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-99-2-0502. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

9. REFERENCES

[1] Chavez, A., and Maes, P. Kasbah: An agent marketplace for buying and selling goods, in Proc. of 1st Int. Conf. on Practical Application of Intelligent Agents & Multi-Agent Technology (1996), 75-90.

[2] Durfee, E. H., and Lesser, V. R. Partial global planning: A coordination framework for distributed hypothesis formation, IEEE Trans. on Systems, Man, and Cybernetics 21, 5 (1991), 1167-1183.

[3] Jennings, N. R., Parsons, S., Noriega, P., and Sierra, C. On argumentation-based negotiation, in Proc. of Int. Workshop on Multi-Agent Systems (Boston, MA, 1998).

[4] Kraus, S. Beliefs, time, and incomplete information in multiple encounter negotiations among autonomous agents, Annals of Mathematics and Artificial Intelligence 20, 1-4 (1997), 111-159.

[5] Kraus, S., Ephrati, E., and Lehmann, D. Negotiation in a non-cooperative environment, J. of Experimental and Theoretical AI 3, 4 (1991), 255-282.

[6] Kraus, S., Sycara, K., and Evenchik, A. Reaching agreements through argumentation: a logical model and implementation, AI 104, 1-2 (1998), 1-69.

[7] Kraus, S., and Wilkenfeld, J. A strategic negotiations model with applications to an international crisis, IEEE Trans. on Systems, Man, and Cybernetics 23, 1 (1993), 313-323.

[8] Kraus, S., Wilkenfeld, J., and Zlotkin, G. Multiagent negotiation under time constraints, AI 75 (1995), 297-345.

[9] Laasri, B., Laasri, H., Lander, S., and Lesser, V. A generic model for intelligent negotiating agents, Int. J. of Intelligent & Cooperative Information Systems 1 (1992), 291-317.

[10] Lander, S. E., and Lesser, V. R. Customizing distributed search among agents with heterogeneous knowledge, in Proc. of the 1st Int. Conf. on Information and Knowledge Management (Baltimore, MD, 1992), 335-344.

[11] Parsons, S., Sierra, C., and Jennings, N. R. Agents that reason and negotiate by arguing, J. of Logic and Computation 8, 3 (1998), 261-291.

[12] Prasad, M. V. N., Lesser, V. R., and Lander, S. E. Retrieval and reasoning in distributed case bases, J. of Visual Communication and Image Representation, Special Issue on Digital Libraries 7, 1 (1996), 74-87.

[13] Rosenschein, J. S., and Zlotkin, G. Designing conventions for automated negotiation, AI Magazine 15, 3 (1994), 29-46.

[14] Zeng, D., and Sycara, K. Bayesian learning in negotiation, International Journal of Human-Computer Studies 48 (1998), 125-141.

[15] Zlotkin, G., and Rosenschein, J. S. Compromise in negotiation: exploiting worth functions over states, AI 84, 1-2 (1996), 151-176.

[16] Zlotkin, G., and Rosenschein, J. S. Mechanism design for automated negotiation, and its application to task oriented domains, AI 86, 2 (1996), 195-244.