Int. J. Conf. On Artificial Intelligence (IJCAI-01), Seattle, WA, 2001

Reflective Negotiating Agents for Real-Time Multisensor Target Tracking

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

Abstract

In this paper we describe a multiagent system in which agents negotiate to allocate resources and satisfy constraints in a real-time environment of multisensor target tracking. The agents attempt to optimize the use of their own consumable resources while adhering to the global goal, i.e., accurate and effective multisensor target tracking. Agents negotiate based on different strategies which are selected and instantiated using case-based reasoning (CBR). Agents are also fully reflective in that they are aware of all their resources including system-level ones such as CPU allocation, and this allows them to achieve real-time behavior. We focus our discussion on multisensor target racking, case-based negotiation, and real-time behavior, and present experimental results comparing our methodology to ones using either no negotiation or using a static negotiation protocol.

1 Introduction

We describe a negotiating agent approach to multisensor target tracking and distributed resource allocation in a real-time environment. Each agent controls a set of resources, and is motivated to use these resources to track targets appearing in its coverage area, and also to make the resources available to other agents in an effort to satisfy the global tracking goals. The act of balancing the local use of tracking resources and the global goal satisfaction increases the complexity of the problem. The agents have to incorporate real-time issues into their decision making process since global tasks and resources are bounded by time. Each agent in the system is autonomous, monitors its environment through a sensor, reacts to changes that it observes, and maintains its own knowledge bases. There is no hierarchical organization among the agents allowing the system as a whole to react to world events more quickly. Since there is also no centralized information shared by the agents, information can only be exchanged directly during negotiation sessions and only what is considered relevant and useful information is communicated. This increases the autonomy of each agent and consequently strengthens the system's robustness. Since there is no top-down coordination, our agents dynamically form temporary coalitions to perform a task, with each agent in the coalition using and also making available its resources.

The driving application for our system is multisensor target tracking, a distributed resource allocation and constraint satisfaction problem. The objective is to track as many targets as possible and as accurately as possible using a network of sensors. Each sensor has a set of consumable resources, such as beam-seconds (the amount of time a sensor is active), battery power, and communication channels, that each sensor desires to utilize efficiently. Each sensor is at a fixed physical location and, as a target passes through its coverage area, it has to collaborate with neighboring sensors to triangulate their measurements to obtain an accurate estimate of the position and velocity of the target. As more targets appear in the environment, the sensors need to decide which ones to track, when to track them, and when not to track them, always being aware of the status and usage of sensor resources.

The problem is further complicated by the real-time constraints of the environment and the fact that agents have to share physical resources such as communication channels and disk storage. For example, for a target moving at one foot per second, accurate tracking requires one measurement each from at least three different sensors within a time interval of less than 2 seconds. The real-time constraints force our agents to deal with issues such as CPU allocation (since speed of execution depends on it), disk space allocation, communication latency, and processing times. Finally, the environment is noisy and subject to uncertainty and error: messages may be lost, a sensor may fail to operate, or a communication channel could be jammed. Thus, in addition to improving autonomy, one is required to promote noise-resistance in agent reasoning, sensor control, and communications.

The sensors are 9.35 GHz Doppler MTI radars that communicate using a 900 MHz wireless, radio-frequency (RF) transmitter with a total of eight available channels. Each sensor can at any time scan one of three sectors, each covering a 120-degree swath. Sensors are connected to a network of CPU platforms on which the agents controlling each sensor reside. The agents (and sensors) must communicate over the eight-channel RF link, leading to potential channel jamming and lost messages. Finally, there is software (the "tracker") that, given a set of radar measurements, produces a possible location and velocity for a target; the accuracy of the location and velocity estimates depend on the quality and frequency of the radar measurements: as we mentioned, the target must be sensed by at least three radars within a two second interval for accurate tracking.

Our solution to the problem is to use *reflective, case-based, negotiating agents.* The agents are *reflective* since they are aware of their resources (including computational ones) and of how their actions and commitments affect these resources. They also integrate case-based reasoning (CBR) and negotiation to dynamically form target-tracking coalitions and to determine how resources should be shared and used. CBR allows the negotiation to adapt to the dynamically changing environment. Negotiation allows a bottom-up generation of an any-time solution. All agents are peers, each responsible for initiating and responding to negotiations. When an agent senses an event that it cannot solve on its own, it dynamically forms a coalition from a subset of its known neighbors. It then initiates negotiation requests to the members of the coalition and conducts 1-to-1 negotiations. This way, the common goal of target tracking is divided into subgoals by the initiating agent.

The integration of real-time, CBR, and negotiation is a unique and innovative approach to the solution of a general class of dynamic, distributed, time-bound, over-constrained resource allocation problems represented by our domain of real-time multisensor target tracking.

2 Agent Negotiation

In our system negotiation is used to allocate sensor and computational resources and to allow the agents to reach an agreement on tracking a target. An agent is connected to and controls a sensor, and is aware of its state and the status of the resources it controls. Each agent operates in one of three different modes: tracking, negotiating, or both.

In this paper we will not discuss in detail how an agent tracks a target, since the topic is only tangentially related to negotiating agents. In a few words, an agent polls its sensor at predefined time intervals and if a radar return is considered "interesting," it then follows the potential target for one second sending all radar measurements to the tracker (the software component that computes target location and velocity given a set of radar measurements).

Since accurate target tracking requires triangulation, an agent that finds a potential target must contact other sensor-controlling agents to ask for their help. Illuminating a target by a radar implies the use of consumable resources: first, the radar will have to abandon its own target tracking (if any) to accommodate the request; second, using the radar consumes battery power; third, sending the measurements to the tracker occupies one of the eight globally available communication channels; fourth, the simple act of communicating between agents and handling the cognitive cost of this communication requires CPU resources. Agents are *cooperative* and desire the completion of the high-level goal of target tracking. At the same time, they are *individualistic*, in that they want to preserve their resources for their own goal satisfaction, and they are also *reliable*, in that they do not easily break a resource commitment made to another agent. Consequently, when an agent requests the use of the resources of another agent, it needs to convince that agent that it has priority in the use of these resources. To do so our agents use *negotiation.*

Negotiation can be used by agents to perform problem solving and to achieve coherent behavior in a multiagent system. Agents can negotiate in a 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 [Kraus, 1997; Kraus *et al.*, 1995]. There are agents that negotiate using the unified negotiation protocol in worth-, state-, and task-driven domains where agents look for mutually beneficial deals to perform task distribution [Rosenschein and Zlotkin, 1994; Zlotkin and Rosenschein, 1996]. Agents can also conduct argumentation-based negotiation in which an agent sends over its inference rules to its neighbor to demonstrate the soundness of its arguments [Jennings *et al.*, 1998]. Finally, there are agents that incorporate AI techniques [Chavez and Maes, 1996; Laasri *et al.*, 1992; Zeng and Sycara, 1998] and logical models [Kraus *et al.*, 1998] into negotiation.

2.1. Negotiation Model

Our agents use a variation of the *argumentative negotiation model.* Traditionally, in argumentative negotiation, an argument is a representation of a sequence of inferences leading to a conclusion [Jennings *et al.*, 1998]. Since our agents are assumed to share the same reasoning mechanism, it is not necessary for them to exchange their inference model with their negotiation partners. In addition, we assume that an agent reasons rationally and in good faith and is cooperative.

Before describing our negotiation model in detail we introduce some terminology: an *initiator* or initiating agent is the agent that requires a resource and contacts another agent to start a negotiating session; a *responder* or responding agent is the one that is contacted by the initiator; a *persuasion threshold* is a value associated with each resource or percentage of a resource that indicates the degree to which an agent needs to be convinced in order to free or share a resource (alternatively, one can view the persuasion threshold as the degree to which an agent tries to hold on to a resource). Finally, a *negotiation strategy* dictates how an agent should behave before the start of a negotiation process; it spells out the time allowed for the agent to complete the negotiation, what type of information to send over as arguments, how agreeable the responding agent should be, and so on.

After the initiator determines that it requires assistance in tracking a target, it establishes a set of negotiation partners (a *coalition,* discussed in the next section), and contacts them to start negotiating. The initiator sends to the responders a message requesting a resource and a time interval it needs this resource (the resource, for example, can be turning on a radar beam at time T). The responder determines if it can satisfy the request immediately (for example, if the radar beam is already on), if it cannot negotiate at all (it is too busy, implying that it has no CPU resources available or no more threads), or if it can negotiate.

If both agents determine that negotiation is possible, they establish a negotiation strategy (see section 2.3) and start negotiating. Each agent has a local view of the world based on its sensor information. The initiator attempts to convince the responder by sharing parts of its local information. For example, it may share with the responder the speed with which the target is traveling or the other tasks it is currently performing; the responder uses this data to "see through the initiator's eyes," and determine if its needs are more pressing than its own. The responder uses a set of domain-specific rules to establish whether the information provided by the initiator pushes it above a resource's persuasion threshold, in which case it would free the resource. For example, a target being tracked by an initiating agent that is already busy tracking another target is a more convincing argument than a target that is already being tracked by multiple sensors.

If the responder is not convinced by the evidential support provided by the initiator, it requests more information from the initiator. The initiator, guided by its negotiation strategy, sends over what it views as its most useful arguments first. The responder evaluates these new arguments and updates the evidence support. This process iterates until either the agents reach an agreement, in which case a resource or a percentage of a resource is freed, or the negotiation fails.

2.2. Coalition Formation

In order to negotiate, an initiating agent must identify a group of other agents that it can talk to. This group is a negotiating coalition and is established dynamically by the initiator. To become a member of a coalition an agent must, first, be known to the initiator, and, second, have the potential to provide useful resources.

An agent knows a subset of the agents in the multiagent system. Usually it knows the agents in its physical neighborhood, since they all control radars that cover a certain area. Since, as mentioned, the radars are sessile, an agent only needs to be told once who its physical neighbors are. To establish who can provide useful resources, the initiator calculates the velocity of the target it is tracking and establishes a potential future path that the target will follow. Next, the initiator finds the radar coverage areas that the path crosses and identifies areas where at least three radars can track the target (remember that tracking requires almost simultaneous measurement from at least three sensors). The agents controlling these radars become members of the negotiating coalition.

Since computational resources are limited, and negotiating consumes CPU and bandwidth, the initiator does not start negotiation with all members of the coalition, but first ranks them and then initiates negotiation with the highest-ranked ones. Ranking of the coalition members is done using a multi-criterion utility-theoretic evaluation technique. The evaluation criteria are:

1. the target's projected time of arrival at the coverage area of a sensor: there has to be a balance between too short arrival times which do not allow enough time to negotiate and too long arrival times which do not allow adequate tracking;
2. the target's projected time of departure from the coverage area of a sensor: the target needs to be in the coverage area long enough to be illuminated by the radar;
3. the number of overlapping radar sectors: the more sectors that overlap the higher the chance that three agents will agree on measurements, thus achieving target triangulation;
4. whether the initiator's coverage overlaps the coverage area of the coalition agent: in this case the initiator needs to convince only two agents to measure (since it is the third one), which may be easier than convincing three;
5. the success rate in previous negotiations between the initiator and the agent in the coalition: previous successes are an indicator that an agent is more willing to be persuaded to free resources (since all agents are collaborative this is an indirect indication that an agent is mostly idle or has more resources than it needs).

At the end of the evaluation all coalition members are ranked and the initiator activates negotiations with as many high-ranked agents as possible (there have to be at least two and the maximum is established by the negotiation threads available to the initiator at the time, since it may be responding to negotiation requests even as it is initiating other negotiations).

2.3. Case-Based Negotiation Strategy

As *negotiation strategy* we define the set of guidelines (or protocol) that govern the behavior of an agent during a particular negotiation. In contrast to other work in negotiation where the negotiating parties followed a predefined, static protocol, our agents dynamically establish a new strategy depending on their current state and the state of the world.

The goal is to situate a negotiation and to improve the chances of its success by taking into account the dynamically changing world state. This is accomplished by using CBR to select, adapt, and eventually learn negotiation strategies.

Since initiating a negotiation and responding to one are fundamentally different tasks, although still governed by the same methodology, each agent has two different case bases: one with strategies for initiating negotiations and one with strategies for responding to negotiation requests. Cases of both initiating and responding negotiation strategies have the same description, but different strategies. In the following we discuss the joint situation description of the two case types and then discuss the two types of strategies separately.

A case contains a description of a situation that allows an agent, using simple weighted matching, to establish similarity between the current situation and the cases in the case base. The situation describes the state of the agent (tasks it is performing, state of the radar, its battery, etc.), the state of the target (current location and speed, projected path, type, etc.), and the model of the potential coalition members (how many, the number that actually were used in negotiation, their capabilities, etc.) Since an agent is always aware of this information, it can match the current situation with the description of the cases in the case base, find the best match, and apply (after adaptation) the negotiation strategy in the case to the current negotiation task.

Each case also contains the negotiation strategy that was used in the past together with the outcome of the negotiation, such as: “offer accepted,” “offer rejected,” “ran out of time,” or “ran out of resources.” The strategy tells the agent how to conduct the negotiation. For the initiator the negotiation strategy consists of the following:

1. a ranking of the classes of information it should use as arguments: during a negotiation each agent attempts to minimize the number and length of messages sent, since with fewer messages 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;
2. the time constraint: how long (in real time) the agent should be negotiating, since the target may leave the area;
3. the number of negotiation steps: a “step” is a complete negotiation communication act where the initiator sends arguments and the responder makes a counter-offer or requests more convincing arguments. Clearly the more steps that are allowed the higher the chance of reaching an agreement, but also the more time and resources are spent;
4. the CPU usage: more CPU resources for a negotiation mean faster negotiation, but also less CPU available for other tasks.

The responder has a slightly different negotiation strategy. It shares some elements of the initiator’s protocol, specifically the time constraint, the number of negotiation steps, and the maximum CPU usage, but it also introduces two more parameters:

1. the power usage: this defines how much power the responder is willing to use to turn on its radar;
2. persuasion thresholds for resources: as already mentioned, each resource has a persuasion threshold associated with it which determines how difficult it will be to convince the responder to free the resource. The resources are radar sectors for performing frequency or amplitude measurements to track a target, CPU allocation, and usage of the RF communication channels. Discrete resources like turning on a radar, have a single valued persuasion threshold. Continuous resources like CPU, where a responder may agree to free a percentage of it, have a linear or an exponential function of evidence support, as persuasion thresholds (so, if an initiator convinces a responder by degree X, then the responder is willing to free N% of its CPU allocation; if it is convinced by degree (X+Y) it will be willing to free (N+M)% of its CPU, where N = f(X), M = f(X+Y) – f(X), and f is either a linear or an exponential function depending on the actual situation). We have chosen these two functions since they are easy to compute and represent two different conceding behaviors—the linear function has a uniform conceding rate whereas the exponential function models agents willing to concede quickly.

After a case has been selected and an old negotiation strategy has been retrieved, the agent adapts the strategy to best fit the current situation. Our adaptation technique uses two sets of rules: one that maps the differences between the case description and the current world state into strategy fixes, and a second one that uses the outcomes of the old negotiation strategy to guide its adaptation into a more potentially successful one. For example, if the current target is faster than the target in the case, then the agent reduces the negotiation time in its strategy. Or, if the old negotiation failed because the agents could not reach an agreement, then the agent may want to use fewer CPU resources and plan to spend less time on the negotiation, since it may fail again. Currently, we have 17 difference-driven and seven outcome-driven domain-specific adaptation rules. On average, each rule has two conditions to facilitate fast token matching and has about three conclusions such as “increase the number of negotiation steps by X,” “decrease the time by Y,” and so on.

Finally, when a negotiation strategy has been created, the agents engage in negotiation, as discussed in section 2.1. After the negotiation is concluded, the agents involved in it decide whether it is worthwhile to learn the strategy they used, and to add it to their respective case base. An agent matches the new case to all cases in the case base and if it is significantly different from all of them it learns the new strategy by storing the case in the case base. By learning cases whose description differs sufficiently from the ones in the case base the agents attempt to improve their negotiation strategies by covering a larger part of the problem domain. If, though, the new case is not sufficiently different from the ones in the case base, the agent determines which one to keep: the new one or the old case which best matches the new one? To do so the agent attempts to increase the diversity of the case base by computing the sum of differences between that old case and the entire case base (minus the best matching old case) and the sum of differences between the new case and the entire case base (minus the best matching old case). If the second sum is greater than the first sum, then it replaces the old case with the new case. The heuristics we use to evaluate whether a new case should be learned or not are similar to the similarity evaluation performed during retrieval, with the additional evaluation of the solution parameters. Since learning is performed by the agent on-line, we have designed it to be of only linearly related to the number of cases in the case base. Consequently, the diversity measurement is between the new case and every case in the case base instead of between all case pairs. This allows us to improve the speed of the learning step when a new case comes in and to reduce the computational requirements.

There has been work in off-line learning of negotiation strategies using genetic algorithms [Matos *et al.*, 1998], but in our work learning is continuous and on-line.

3 Real-Time Reflective Agents

A fundamental concern in multisensor target tracking is the timeliness of the measurements: a radar must be active and illuminating an area when a target is passing through it and when other radars are measuring, too. This introduces real-time constraints to the sensor management by the agents: negotiations must be concluded within sufficient time to allow execution of sensing commands, or must be aborted to allow negotiation with other members of the coalition. To achieve real-time behavior the agents must be fully aware of the status of system-level resources and of the passage of time. This awareness defines a *real-time reflective agent*.

Our agents use the Real-Time Scheduling Services (RTSS) that reside on top of the KU Real-Time system (KURT) [Srinivasan *et al.*, 1998] that adds real-time functionality to Linux. First, the RTSS provides an interface between the agents and the system timers, allowing agents to: (1) query the OS about the current time; (2) ask the RTSS to notify them after the passage of certain length of time; and (3) ask the RTSS to ping them at fixed time intervals. This allows agents to know when to, for example, conclude a negotiation process or turn on a radar sector. Second, the agents may ask the RTSS to notify them when certain system-level events occur, such as process threads being activated, or communication messages going out or coming into the system. Third, the agents can ask the RTSS to allocate them a percentage of the CPU for each one of their threads (such as the ones controlling the radar and tracking or the ones used in negotiations) and to schedule this allocation within an interval of time. This way agents residing on the same computational platform can establish execution priorities and can control how fast an operation can be performed (clearly, more CPU scheduled in consecutive time intervals implies faster execution for a thread, leading to faster reasoning and negotiation).

The RTSS may be unable to perform such a CPU allocation and scheduling, if, for example, all available CPU resources are already occupied. Then, the requesting agent is notified and is also informed of which agents are using the CPU resources. This allows the agent to initiate a negotiation for CPU with the other agents. This is when the fourth function of the RTSS comes into play: an agent needs to know what percentage of its allocated CPU it is using, to be able to determine whether it is willing to give up part of its CPU allocation to a requesting agent. After agents have negotiated a new sharing of CPU resources they request a rescheduling of the allocations, and the RTSS dynamically performs it.

The RTSS allows agents to be full masters of all their resources, including system-level ones. Agents can negotiate about CPU and can simply cede part of their allocation to other agents. Knowledge of the passage of real time, of the occurrence of system-level events, and of CPU usage and load make our agents reflective and allow them to function effectively in a real-time domain.

Previous work in real-time AI fell under two general categories: (1) anytime algorithms [Dean and Boddy, 1988] where a solution to a problem can be incrementally refined and can be applied at anytime during the refinement process, and (2) multiple methods or approximate processing [Lesser *et al.*, 1988] where different approaches to a solution are available and can be combined.

4 Experimental Results

The reflective, real-time, case-based negotiating agents described in the previous sections have been implemented and tested using real sensors and targets moving in a physical environment. The agents exhibit all of the behavior described: they use CBR to select and adapt a negotiation strategy, use the RTSS to request CPU resources and to have time and system awareness, negotiate for radar use, and learn the new negotiation strategies they have developed. Most importantly, the agents achieve the high-level goal of the system: they track targets traversing an area covered by many radars.

Our experiments concentrated on evaluating whether negotiating agents can track targets better, and whether CBR results in better negotiation strategies. Our hypotheses were, first, that negotiating agents can track targets better since they can coordinate radar measurements and achieve better triangulation, and, second, that negotiation using CBR will result in better tracking than using a static negotiation protocol, since CBR will allow adaptation of the strategy to the current situation. Our experiments support these hypotheses. In addition to the accuracy of tracking, we used communication as a measure of quality (length of messages, frequency of messages and message cost, i.e. length times frequency), since communication is an important bottleneck of scale up.

First we compared our system to a multiagent, sensor-controlling network where there is no communication between the agents, and where when a target appears in the coverage area of a sensor it is tracked. Next, we compared our case-based negotiating agents to a system where negotiation uses a predefined, static strategy. We selected the static strategy carefully to make sure it should be adequate for most cases.

In general, the results, summarized in figures 1-4, were very encouraging. The agents which used no negotiation sent almost 20% more messages but had almost 27% worse tracking accuracy than negotiating agents. The non-negotiating agents exchanged no messages, and only sent their radar measurements to the tracking software. Since there was no coordination of the measurements, there were too many messages sent to the tracker. On the other hand, we also found that such messages are short compared to arguments exchanged between agents during negotiation, resulting in lower message costs—the product of the average length and the total number of messages sent per second. Since there was no cooperation to triangulate measurements, the resulting accuracy was poor.

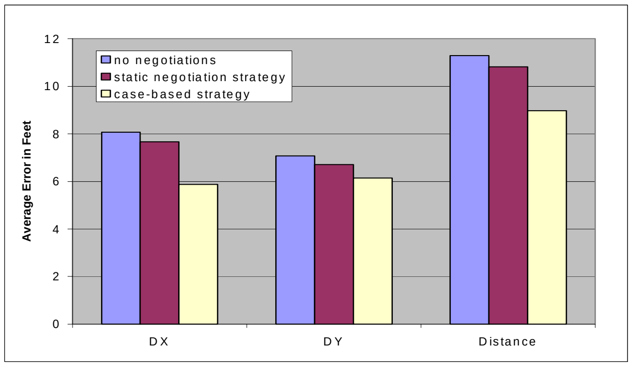

Figure 1: Tracking accuracy vs. agent behavior

The agents that used a static negotiation strategy fared worse than the ones that used a case-based, adaptive strategy. Specifically, the agents using a static protocol sent approximately 10% fewer messages (though with a slightly higher message cost) and had almost 18% worse accuracy than the case-based negotiating agents. The message cost is due to the fact that the case-based agents change the ranking of the arguments they communicate based on the situation; this leads to overall more effective communication acts. The accuracy is due to the fact that case-based agents adapt their negotiation to the current situation and have a higher chance of achieving agreement for resource allocation; on the other hand the static strategy agents failed to agree more often and this led to failure to perform the multiple, simultaneous radar measurements that are required for accurate tracking.

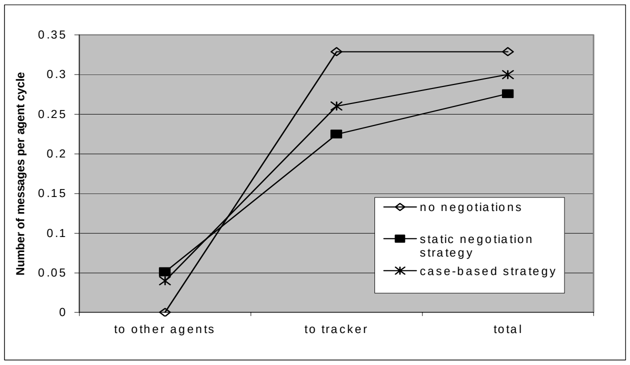

Figure 2: Number of messages to agents and to tracking software vs. agent behavior.

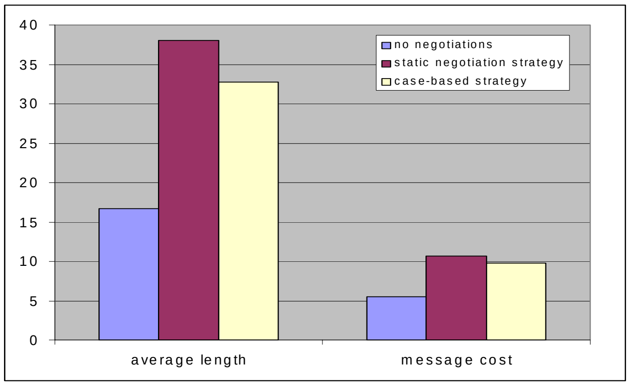

Figure 3: Message statistics vs. agent behavior. Message cost is the product of the average length and the total number of messages sent per second.

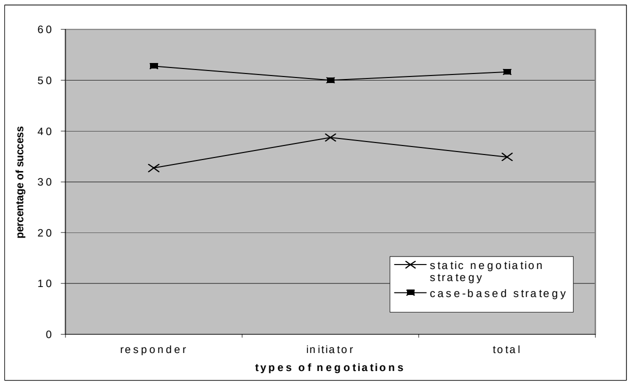

Figure 4: Percentage of successful negotiations vs. negotiation strategy type. A successful negotiation is one that completes with a deal between the two negotiating agents.

5 Conclusions

We have described a multiagent approach to distributed resource allocation problems, particularly to multisensor tracking of targets in a real-time environment. Our approach uses negotiations among agents to exchange information based on strategies retrieved using a case-based reasoning system. This allows the agents to learn negotiation strategies based on previous experiences, adapt to the current situations, and avoid repeating past failures. We have shown experimentally that CBR-based negotiations helped agents to negotiate more efficiently and more successfully, indirectly helping the agents track their targets more accurately. The agents in our system use real-time scheduling services to become reflective of the system-level resources they use and to be time-aware; this allows the agents to work in an environment of real-time constraints. Finally, we showed experimentally that reflective negotiating agents can track targets much better than agents that simply react to the presence of targets in their environment. The reflective nature of the agents allows them to schedule the precise time of measurement and also exchange computational resources, leading to faster and more efficient processing.

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 the 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.

References

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

[Dean and Boddy, 1988] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. of the Seventh National Conf. on Artificial Intelligence* (St. Paul, MN), August, 49-54,

[Jennings *et al.*, 1998] 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).

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

[Kraus *et al.*, 1998] Kraus, S., Sycara, K., and Evenchik, A. Reaching agreements through argumentation: a logical model and implementation, *AI Journal* 104, 1-2, 1-69.

[Kraus *et al.*, 1995] Kraus, S., Wilkenfeld, J., and Zlotkin, G. Multiagent negotiation under time constraints. *Artificial Intelligence* 75, 297-345.

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

[Lesser *et al.*, 1988] Lesser, V. R., Pavlin, J., and Durfee, E. Approximate processing in real-time problem solving. *AI Magazine* 9, 1, 49-61.

[Matos *et al.*, 1998] Matos, N., Sierra, C., and Jennings, N. R. Negotiation strategies: an evolutionary approach. In *Proc. of Int. Conf. on Multiagent Systems (ICMAS'98)* (Paris, France), July 4-7, 182-189.

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

[Srinivasan *et al.*, 1998] Srinivasan, B., Pather, S., Hill, R., Ansari, F., and Niehaus, D. A firm real-time system implementation using commercial off-the shelf hardware and free software. In *Proc. of the Real-Time Technology and Applications Symposium,* (Denver, CO).

[Zeng and Sycara, 1998] Zeng, D., and Sycara, K. Bayesian learning in negotiation, *Int. J. of Human-Computer Studies* 48, 125-141.

[Zlotkin and Rosenschein, 1996] Zlotkin, G., and Rosenschein, J. S. Mechanism design for automated negotiation, and its application to task oriented domains, *Artificial Intelligence* 86, 2, 195-244.