Chapter 1

A SATISFICING, NEGOTIATED, AND LEARNING COALITION FORMATION ARCHITECTURE

Leen-Kiat Soh[1], Costas Tsatsoulis[2], Hüseyin Sevay[2]

[1] *Computer Science and Engineering Department,*
University of Nebraska,
115 Ferguson Hall,
Lincoln, NE 68588 USA
lksoh@cse.unl.edu

[2] *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
{tsatsoul,hsevay}@ittc.ku.edu

Abstract

In this chapter, we present a multiagent system architecture for dynamic coalition formation and coalition strategy learning in a realtime multisensor target tracking environment. Agents operate autonomously, and they have incomplete information about their potential collaborators. In addition, accurate target tracking requires that multiple agents recognize and synchronize their actions– collecting measurements on the same target within the same time frame. Therefore some form of cooperation is necessary. In our system, agents form coalitions via multiple 1-to-1 negotiations. However, due to the noisy and uncertain properties of the environment, coalitions formed can be only suboptimal and satisficing. To better adapt to changing requirements and environment dynamics, each agent is capable of multiple levels of learning. Each learns about how to negotiate better (case-based learning) and how to form a coalition better (reinforcement learning). To increase the chance of reaching a high-quality negotiated deal, our work also addresses issues in task allocation and dynamic utility-based profiling.

Keywords: coalition formation, argumentative negotiation, learning, case-based reasoning

1. Introduction

In this chapter, we describe a satisficing, negotiated, and learning coalition formation architecture. Our focus is on forming dynamic multiagent coalitions for tracking targets in a noisy and uncertain environment. A coalition is a group of agents that collaborate to perform a task that is generated as a response to an event that has occurred in the environment. A dynamic coalition is one that is formed in response to an event and dissolved when that event no longer exists or when the tasks required to respond to that event are completed. A coalition is necessary when an agent cannot respond to an event by itself due to lack of information, knowledge, functional capabilities, or other resources. Therefore, an agent forms a coalition with other agents that it believes can be of help in solving a problem.

In general, an agent prefers to form a coalition that is optimal to maximize the yield of the system as a whole. To facilitate such optimal rationalization, the coalition-initiating agent needs to have complete information about its world and its neighboring agents. In our problem domain, agents have incomplete information, maintain different information bases, and must react in realtime to events that they encounter, and cannot afford optimality. How resources are used and shared motivated us to design methodologies that allow each agent to be conscious of its own local resources and also those it shares with other agents so that coalitions can be formed in the most efficient manner possible under the circumstances, and targets can be tracked as well as possible. Hence, our coalition formation architecture deals with coalitions that are suboptimal and satisficing, since noise and uncertainty in the environment preclude the possibility of an optimal, fully rational coalition design.

In our work we are interested in improving the quality of a coalition formation process and the quality of a coalition in terms of its future tracking performance. Our coalition formation model is adaptive and takes realtime issues into consideration [Soh and Tsatsoulis 2002a, Soh and Tsatsoulis 2002b]. An initiator agent starts the coalition formation process first by selecting members in the that agent's neighborhood that are qualified to be part of an *initial coalition,* and that the coalition initiator believes (based on experience) that are most likely to accept its request for sharing of resources and tasks. A neighborhood of an agent consists of all other agents that the agent knows about *a priori.* After selecting its potential coalition candidates, the agent evaluates the status of these candidates to rank them in terms of their respective potential utility values to the intended coalition. Following this ranking step, we apply

a number of methods for assigning tasks to the initial coalition members so as to maximize the possibility of all tasks being performed. The task allocation process takes into account the abilities of agents, their previous behavior, and the dynamics of the environment.

Next the agent initiates negotiation requests to the top-ranked candidates in order to attempt to convince them to join the coalition. In argumentative negotiation agents request the sharing of resources and justify their request by providing arguments, such as their own work load, their capabilities to provide the requested resource or service, and domain-specific information such as target location and velocity. In our work an agent may accept the arguments supplied by its negotiation partner and part with some of the resources it controls, or it may counter-offer by offering a part of what is being requested, or it may outright refuse the request. In the two latter cases, the argumentation between agents continues until a deal is reached, a deal is considered impossible, or time lapses. The negotiation process is guided by a protocol that indicates what arguments may be presented first, which counter-offers may be made, what is the willingness of an agent to agree to a deal, and so on. The protocol is situated and is derived from the state of the agent (e.g. number of tasks being performed or already committed to), the state of the target (e.g. speed and location), and the state of the world (e.g. how many other agents can provide the required resource). Since these are all dynamic, we have chosen to use Case-Based Reasoning (CBR) to retrieve the most promising negotiation protocol from a case base of previously used ones. CBR allows us to retrieve, adapt, and then apply a negotiation protocol that is best suited to the specific situation in the environment, making our negotiation technique adaptive and dynamic. Cases of negotiations are also saved and learned, allowing the system to improve its negotiation performance, leading to better deals.

In the end, the coalition may fail to form because the candidates may refuse to cooperate, or the coalition may form successfully when enough members reach a deal with the initiating agent. Finally, the initiating agent sends an acknowledgment message to the coalition members to announce the success or failure of the proposed coalition. If the coalition formation attempt resulted in success, then all coalition members that have agreed to join will carry out their respective tasks at the appropriate time. This approach is opportunistic as the goal is to obtain a satisficing coalition and the success of the formation is not guaranteed. This is the risk that each agent is willing to take: the utility of responding timely to a problem is dominating the utility gained from the quality of the solution, since the domain is time-critical and dynamic. In a way, an agent has no choice but to attempt and accept failures in coalition formation several times as long as the failures are quickly reached.

Our negotiating agents behave based on a set of social characteristicsthat are shared by all of them. Specifically, the sensor-controlling agents can be described as:

1 **Autonomous:** Each agent runs without intervention from human users. It maintains its own knowledge base, makes its own decisions, and interacts with the sensor it controls, neighbors and environment.

2 **Rational:** Each agent is rational in that it knows what its goals are, and it can reason and choose from a set of options and make an advantageous decision to achieve its goal [Wooldridge and Jennings 1995].

3 **Communicative:** Each agent is able to communicate with others, by initiating and responding to messages, and carrying out conversations.

4 **Reflective (or Aware):** According to [Brazier and Treur 1996], a reflective agent reasons based on its own observations, its own information state and assumptions, its communication with another agent and another agent's reasoning, and its own control or reasoning and actions. By being reflective, each agent is time-aware and situationally aware. When an agent is time-aware it observes time in its decision making and actions. When an agent is situationally aware it observes the resources that it shares with other agents, its current tasks, messages, profiles and actions of its neighbors, and the external changes in the environment. However, we require a stronger level of situational awareness. An agent also observes its own resources that *sustain the well-being* of the agent. For a hardware agent, these resources may be the battery power, the radio frequency links, and so on. For a software agent, these resources may be CPU, RAM, disk space, communication channels, and so on.

5 **Adaptive:** Each agent is able to adapt to changes in the environment and learns to perform a task better, not only reactively but also from its past experience.

6 **Cooperative:** Each agent is motivated to cooperate, if possible, with its neighbors to achieve global goals while satisfying local constraints, and does not knowingly lie or intentionally give false information.

We require that all agents be capable of negotiation in which they share a common vocabulary that enables message understanding, and that each agent knows what resources may be used or controlled by a non-empty subset of the other agents in the environment so that it can determine whom to negotiate with. In our particular domain of application, each agent controls the same set of resources, since each one controls the same type of sensor. Also, each agent

uses the same negotiation methodology based on case-based reasoning, but the individual case bases differ due to the experiences of individual agents.

Formally, our multi-agent system architecture is defined as follows. Suppose that we denote a multiagent system as Ω. Suppose that we define a neighborhood of an agent α_i, Ψ_{α_i}, such that $\Psi_{\alpha_i} \subseteq \Omega$, $\Psi_{\alpha_i} \neq \emptyset$, and that the agent α_i knows about all other agents in the neighborhood. Thus, we have $\alpha_i, \alpha_j \in \Psi_{\alpha_i}, \forall j \lambda(\alpha_i, \alpha_j)$ where $\lambda(a, b)$ means agent a knows about the existence of agent b and can communicate with agent b. A *neighborhood* is different from a *team* as defined in [Tambe 1997], since a team is task-driven and is formed among a set of agents to accomplish a task. A neighborhood, on the other hand, is a subset of agents of the multiagent system that could form a team. In our particular domain of application, a neighborhood consists of a set of agents that control sensors that are physically close to each other and whose sensing coverage overlap. So, in our multi-agent system Ω, there is a set of neighborhoods, $\Omega = \{\Psi_{\alpha_1}, \Psi_{\alpha_2}, \Psi_{\alpha_N}\}$, and each neighborhood can form any number of teams. Neighborhoods do not necessarily have the same number of members, and neighborhoods may even share members.

In Section 1.2 we present the formation of the initial coalition based on the agents' previous experiences with their coalition partners. In Section 1.3 we discuss how an agent can allocate tasks to the coalition members. Section Section 1.4 addresses how a resource-sharing coalition is finalized using our argumentative negotiation model, the application of case-based reasoning (CBR), satisficing allocation algorithms, and also addresses the role of learning in our framework. In Section 1.5 we discuss how the final allocation of tasks and resources is announced to the coalition members and how commitments are made. We present and discuss our simulation and experimental results in Section 1.6. In Section 1.7 we briefly discuss related work to our research, and, finally, in Section 1.8 we present our conclusions.

2. Initial Coalition Formation

When prompted by the changes in the environment to solve a new problem an agent takes on the role of a negotiation initiating or initiator agent. The initiator then describes the problem parametrically based on what it observes in the environment and also on its own current state. This parametric description helps guide the identification of coalition candidates from among the neighbors of the initiating agent. To establish who can provide useful resources or perform certain tasks, the initiator uses its knowledge profile of its neighborhood. This knowledge profile consists of the name and the communication channel of each neighboring agent, as well as the list of resources that agent has and the tasks that agent can perform. By matching the neighbor profile with the profile of the problem, the initiator selects coalition candidates and forms an

initial coalition. Since computational resources are limited, and negotiation consumes CPU and communication bandwidth, the initiator does not start to negotiate with all members of the coalition at one time. Instead it ranks its potential collaborators and then initiates negotiation selectively with the top-ranked agents. Ranking of the coalition membersis done using a multi-criterion utility-theoretic evaluation technique.

The potential utility of a candidateis a weighted sum of its ability to help towards the problem at hand, its past relationship with the initiator, and its current relationship with the initiator. Formally, we express potential utility, U^*,as

$$U^*_{\alpha_k a_i} = W(a_i, e_j)\ [rel_{past,a_i}(\alpha_k, t),\ rel_{now,a_i}(\alpha_k, t),\ ability_{a_i}(\alpha_k, e_j, t)]$$

where

$$W(a_i, e_j) = \begin{bmatrix} w_{past,a_i,e_j} \\ w_{now,a_i,e_j} \\ w_{ability,a_i,e_j} \end{bmatrix}$$

and $w_{past,a_i,e_j} + w_{now,a_i,e_j} + w_{ability,a_i,e_j} = 1$.

The terms w_{past,a_i,e_j}, w_{now,a_i,e_j}, and $w_{ability,a_i,e_j}$ are weights used by the agent a_i to factor three groups of attributes: past relationship, current relationship, and the abilityof the candidate to perform the requested task. Note that, ultimately, these weights may be dynamically dependent on the current state of a_i and the task e_j. The term, U^* is the potential utility of candidate agent α_k, as seen by agent a_i, and t refers to time. The term, $rel_{past,a_i}(\alpha_k, t)$, is the past relationship value between the candidate α_k and the agent a_i. The term, $rel_{now,a_i}(\alpha_k, t)$, represents the current relationship value between the the initiator and the candidate agents. Finally, $ability_{a_i}(\alpha_k, e_j, t)$, represents ability value of the candidate α_k computed by agent a_i.

The current relationship is based on the interactions between two agents at the time when the coalition is about to be formed. The past relationship, however, is collected over time, and it enables an agent to adapt its behavior in forming coalitions more effectively.

Ability: The ability value is based on a set of general heuristics and domain-specific criteria. The general heuristics are:

1 the uniqueness of a given resource or functionality that a candidate can provide related to the problem at hand,

2 the quality of the resource and functionality that a candidate can offer, including volume, duration, efficiency, and so on,

3 how many resources that are useful in solving the problem the candidate has, or how many different functions can the candidate perform towards solving the problem.

In addition to these general heuristics, evaluation also involves domain-specific criteria with which we examine the quality of the resource and functionality of the candidate driven by their applicability to the problem at hand as determined by domain-specific requirements.

Current Relationship: Suppose that the maximum number of concurrent negotiations that an agent can conduct is N, the number of tasks that the agent is currently executing as requested by the candidate is t, the number of all tasks that the agent is currently executing is T, and the number of ongoing negotiations initiated by the agent to its candidate is n. Then, the current relationship value is a weighted sum of the following attributes:

- negotiation strain between the agent and the candidate: n/N,
- negotiation leverage between the agent and the candidate: t/N, and
- degree of strain imposed on the agent by the candidate: t/T.

The first attribute approximates how demanding the agent is of a particular neighbor. The more negotiations an agent is initiating with a neighbor, the more demanding the agent is and this strains the relationship between the two agents, and negotiation may suffer. The last two attributes are used as leverage that the agent can use against a neighbor that it is negotiating with.

The current relationship can be computed readily from the status profile of the agent of its tasks and the negotiation processes.

Past Relationship: Suppose that

(a) the number of all previous negotiation requests initiated by the agent to the candidate is $NegAtoC_{req}$,

(b) the number of all previous successful negotiations initiated from the agent to the candidate is $NegAtoC_{succ}$,

(c) the number of negotiation requests from the candidate to the agent that the agent agrees to entertain is $NegCtoA_{succ}$,

(d) the total number of all negotiation requests initiated by the agent to all its neighbors is $NegA_{totreq}$,

(e) the total number of all successful negotiations initiated by the agent to all its neighbors is $NegA_{totsucc}$,

(f) the number of all previous negotiation requests initiated by the candidate to the agent is $NegCtoA_{req}$,

(g) the number of all previous successful negotiations initiated by the candidate to all its neighbors is $NegC_{totsucc}$, and

(h) the number of all previous negotiations initiated by the candidate to all its neighbors is $NegC_{totreq}$.

The past relationship value is a weighted sum of the following attributes:

(a) the helpfulness of the candidate to the agent: $\frac{NegAtoC_{succ}}{NegAtoC_{req}}$,

(b) the importance of the candidate to the agent: $\frac{NegAtoC_{req}}{NegA_{totreq}}$,

(c) the reliance of the agent on the candidate: $\frac{NegAtoC_{succ}}{NegA_{totsucc}}$,

(d) the friendliness of the agent to the candidate: $\frac{NegCtoA_{succ}}{NegCtoA_{req}}$,

(e) the helpfulness of the agent to the candidate: $\frac{NegCtoA_{succ}}{NegC_{totsucc}}$,

(f) the relative importance of the agent to the candidate: $\frac{NegCtoA_{req}}{NegC_{totreq}}$, and

(g) the reliance of the candidate on the agent: $\frac{NegCtoA_{succ}}{NegC_{totsucc}}$.

These attributes are based on data readily available whenever the agent initiates a negotiation request to one of the candidates or whenever it receives a request from one of its neighbors. The higher the value of each of the attributes, the higher the potential utility (U^*)the candidate may contribute to the coalition.

Attributes (a)-(c) tell the agent how helpful and important a particular neighbor has been and are used to estimate the chance of having a successful negotiation with the candidate based on how the candidate has behaved in its past interactions with the agent. The more helpful and important that neighbor is, the better it is to include that neighbor in the coalition. Attributes d-g implement a sense of social reciprocityin the agent interactions in a coalition. Agents are programmed to be more willing to assist other agents that have been useful to them in the past. So, the agent expects a particular candidate that is has helped in the past to be grateful and more willing to agree to a request based on the agent's friendliness, helpfulness, and relative importance to that candidate. Finally, the initiator makes use of the potential utilities to carry out task allocations and assignments. Based on the overall potential utilityof the initial coalition, the initiator may want to lower its demands to improve the chance of forming a coalition. By the same token, if the potential utility of a candidate is high, then the initiator may want to ask more from that candidate.

When an agent approaches a candidate, the candidate examines its current and planned activities against the requested task. If the candidate realizes that the requested task is doable, then it agrees to negotiate. Otherwise, it refuses.

3. Allocation Algorithms

After an initial coalition containing agents that is believed will be willing to help in the task is formed, the initiating agent has to decide how to properly request tasks or resources from each initial coalition member. Previously Soh and Tsatsoulis proposed several resource allocation algorithms [Soh and Tsatsoulis 2002c], which we summarize here. The goals for the design of our allocation algorithms include improving the chance of a final coalition formation on time and with incomplete information, the robustness of the final coalition formation,and the ability of the final coalition to change dynamically.

After the initial coalition is determined, the initiating agent needs to design a task allocation plan.Based on the potential utilityof a candidate agent, the initiating agent matches a particular task in the plan to a candidate. If there is at most one task assigned to a candidate, we call the assignment *1-to-1*, and *many-to-1* otherwise. We identified four (non-exclusive) types of allocation algorithms,*priority-based*, *flexibility-bounded*, *greedy*, and *worried*.

Priority-Based: In this scheme, we rank each subtask based on the maximum potential utility of each candidate performing that subtask. Given the ranked list, the agent first assigns the top-ranked subtasks to their respective candidates. As a result, it is possible that a candidate may be overloaded with all subtasks and another candidate does not receive any assignment.

Flexibility-Bounded: The number of coalition members that an initiating agent can approach is bounded by its available resource. For example, the number of coalition members to be approached is determined by the number of negotiation tasks that can be run currently, and the availability of computational resource that the agent currently has to support the eventual negotiations. As a result, we have to perform lazy task shuffling that tries to dump all the extra assignments into the first (and top-prioritized) task-candidate pair. This may be rational, as the first candidate associated with the top-prioritized task-candidate pair is more likely to become a useful coalition member.

Imperfect Coalition and Greedy Algorithms: When an initiating agent is faced with a dilemma where it has more negotiations to perform than it has available resources to conduct negotiations with its coalition members, it either quits or continues with as many negotiations as possible to recruit as many coalition members as possible. This is feasible since each agent is capable of forming a coalition dynamically on its own, and, if the initiating agent can get the message out, then the hope is that the coalition members will pass the message along to their own coalition members. So, an initiating agent does not necessarily have to plan for a perfect coalition solution for an event. Moreover, it is unlikely to build a perfect coalition solution even with a perfect plan, since the coalition formation processis subjected to dynamic changes in the environment, noise, message loss, refusals to negotiate, and failed negotiations.

The greedy algorithm makes use of a modified prioritized utility score called the *focused utility* that emphasizes in all the subtasks that the candidate knows how to perform, from the viewpoint of the initiating agent. If a candidate has C functional capabilities that suit the subtasks that the initiating agent wants done, then it has C such focused utility values. For example, if the candidate does not do subtask T well, but does other tasks well, then the agent may assign subtask T to the candidate. An initiating agent becomes greedy when practicing the above algorithm because

1 it tries to minimize its own rationalization and computing process,

2 it selects the candidate with the higher overall utility values to approach hoping for a successful negotiation,

3 it cares mostly about high- priority tasks,

4 it tries to maximize its chance of getting a particular task done by including sub-utilities in the focused utility evaluation, and

5 it hopes to shift its responsibility (partially) to the candidates via successful negotiations, expecting the candidates to spawn their own coalitions to help respond to the event.

Insurance and Worried Algorithms: Since negotiations cannot be guaranteed to be always successful, some initial candidates may be dropped from the final coalition. This also implies that, if an initiating agent relies too heavily on one particular candidate, the initiating agent may lose a large portion of the coalition's utility. So, in the task allocation and assignment process, we can build in some insurance policies (i.e., some alternative plans) at least to absorb the impact of such scenarios. Of course, an initiating agent considers these plans only when it has enough computational resources to do so.

A worried algorithm assigns a subtask to multiple candidates. First, it assigns each subtask to the candidate with the highest ability performing that subtask. Then, if there are candidates that have not been assigned a subtask, the algorithm goes through another round of insurance assignments. The algorithm stops when all the candidates have been reached. Note that the insurance assignments as a result of the worried algorithms will be aborted once the initiating agent has achieved a satisfactory coalition (e.g., as different negotiations complete with successes).

Over-Demanding and Caps It is possible that the lazy and greedy algorithms may end up assigning all tasks to a single agent. This becomes an over-demanding scenario that complicates the negotiation, and, as a result, the coalition may suffer. Hence, the number of assignments for a candidate has to be bounded by a cap that can be determined dynamically. For example, if

the primary task that the initiating agent wants the candidate to perform is extremely important, or highly unique, then it is better for the initiating agent not to be over-demanding in its approach to the candidate. On the other hand, if the candidate has been very helpful and friendly, then the initiating agent may be able to take advantage of that relationship by over-demanding.

Suppose we denote the cap for an assignment ρ for candidate α_k as $\lceil f_\rho \rceil_{\alpha_k}$, then $\lceil f_\rho \rceil_{\alpha_k} \propto rel_{past,a_i}(\alpha_k, t)$, $\lceil f_\rho \rceil_{\alpha_k} \propto rel_{now,a_i}(\alpha_k, t)$, and $\lceil f_\rho \rceil_{\alpha_k} \propto 1/ability_{a_i}(\alpha_k, e_j, t)$.

These caps can be inserted into all the algorithms above to prevent too many assignments to a single agent.

4. Coalition Finalization

After obtaining the initial coalition and the coalition candidates ranked according to their respective potential utility values,and after assigning tasks to each one, the initiator invokes the coalition finalization step that is performed using argumentative negotiation.The task allocation step creates a negotiation request for each member of the initial coalition; the coalition initiating agent will approach the agent and negotiate with it for the resource/task allocated to it. Our agents use a variation of the argumentative negotiation model [Jennings et al. 1998] in which it is not necessary for them to exchange their inference model with their negotiation partners, since they are assumed to share the same reasoning mechanism. In addition, we assume that an agent reasons rationally and in good faith, and is cooperative. Note that after the initial coalition formation, the initiator knows who can and might help. Hence the goal of negotiation is to find out who is *willing* to help. Before describing our negotiation model, 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, and a responding agent (or responder) is the one that is contacted by the initiator (See also [Soh and Tsatsoulis 2001a]).

First, the initiator contacts a coalition candidate to start a negotiating session. When the candidate or responder agrees to negotiate, it computes a *persuasion threshold*that indicates the degree to which it needs to be convinced in order to free or share a resource or perform a task. Alternatively, one can view the persuasion threshold as the degree to which an agent tries to hold on to a resource. Subsequently, the initiator attempts to convince the responder by sharing parts of its local information. The responder, in turn, 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 frees the named resource. If the responder is not convinced by the evidential support provided by the initiator, it requests more information that is then provided by the initiator. The negotiation continues based on the established strategy and

eventually either the agents reach an agreement, in which case a resource or a percentage of a resource is freed, or the negotiation fails. Note that, motivated to cooperate, the responder also makes a counter-offer when it realizes that the initiator has exhausted its arguments or when time is running out for the particular negotiation. How to negotiate successfully is dictated by a negotiation strategy,which each agent derives using CBR. CBR greatly limits the time needed to decide on a negotiation strategy, which is necessary in our realtime domain since the agent does not have to compute its negotiation strategy from scratch.

4.1 Negotiation Strategy

We define a *negotiation strategy*as 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 follow 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, 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 use the same description language, but they involve different strategies. Each case also contains the negotiation strategy that was used in the past together with the outcome of that negotiation that may indicate whether the offer was accepted or rejected, and whether the negotiation ran out of time or ran out of resources to continue further. The strategy tells the agent how to conduct the negotiation. For the initiator, the negotiation strategy includes 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 the length of messages it sends, 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 of such messages would require less time and less bandwidth. Thus, it is important for an initiating agent to decide which information pieces are more important to send to the responding agent

2 **time constraint:** how long (in realtime) the agent should be negotiating, since the target may leave the sensing area controlled by the current agent

3 **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 arguments to be convinced. 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 **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 **power usage:** this defines how much power the responder is willing to use to turn on its sensor

2 **persuasion thresholds for resources:** as we 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 different types of measurements required 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 have a linear or an exponential function associated with them. For example, 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 $N = f(X)$, $M = f(X + Y) - f(X)$, where f is either a linear or an exponential function, chosen appropriate depending on the current scenario.

3 **persuasion functions:** each is either a linear or an exponential function. We have chosen these two types of functions since they are easy to compute and represent two different conceding behaviors. The linear function has a uniform conceding rate, and the exponential function is more willing to concede quickly. Thus, in situations where a deal has to be made quickly, an agent tends to choose an exponential function to guide its negotiation. Each of the above functions is modified by two parameters: conceding rate and willingness factor. The conceding rate is the slope in the linear function and the curvature in the exponential function. The willingness factor is the amount of resource that an agent is willing to give up when there is zero evidence. With a persuasion function, the responding agent also makes counter-offers when it realizes that

it is about to run out of the time that it has allotted for a particular negotiation task, or when the initiating agent has no more arguments to provide.

4.2 Negotiation Protocol

Figure 1.1 shows our negotiation protocolas a state diagram between two agents, a and b. In the figure, squares represent terminal states, and the double-circle is the initial state.

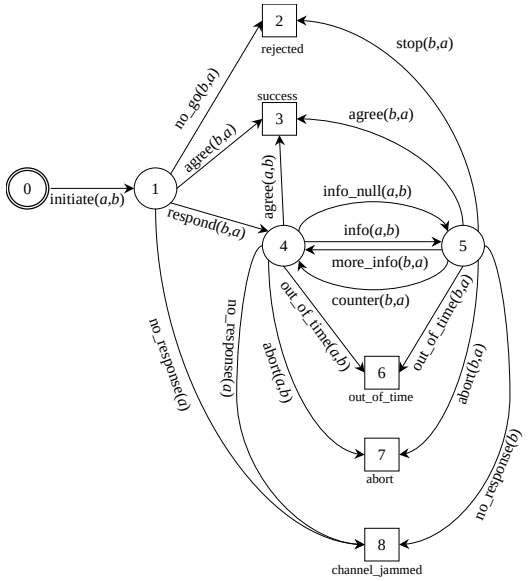

Figure 1.1. The negotiation protocol

State 0 is the initial state. State 1 is the first handshake state, indicating whether the initiated negotiation will be entertained. State 4 is the initiating state while state 5 is the responding state. The initiating state is where the initiating agent, a, returns to, basically the processing loop of the negotiator module. The responding state is where the responding agent, b, returns to, respectively. Agent a initiates a negotiation request to agent b by sending an *INITIATE* message (*initiate(a,b)*), and the state transitions to state 1. At this juncture, there are four possible scenarios. First, agent b may outright refuse to negotiate by sending a *NO_GO message* (*no_go(b,a)*). This results in a final state of failure (state 2, rejected). Second, agent b may outright agree to the requested task by sending an *AGREE* message (agree(b,a)). This results in a final state of success (state 3). Third, agent b may decide to entertain the

negotiation request and thus sends back a *RESPOND* message (respond(b,a)). This transitions the state to 4. Fourth, there may be no response from agent b. Thus agent a, after waiting for some time, has no choice but to declare a no response (no_response(a)) and moves to a state of failure (state 8, channel_jammed). When the agents move to state 4, the argumentative negotiation begins and iterates between states 4 and 5 until one side opts out or both sides opt out or both sides agree. During the negotiation, agent a provides information or arguments to b by sending *INFO messages* (*info(a,b)*). Then agent b demands information or arguments from a by sending *MORE_INFO* messages (*more_info(b,a)*). If agent a runs out of arguments, it sends a *INFO_NULL* message to b (*info_null(a,b)*). If agent b runs out of patience, it makes a counter-proposal by sending a *COUNTER* message to a (*counter(b,a)*). Then agent a can agree to the counter-offer (*agree(a,b)*), and the agent transitions to the state of success (state 3), or provide more information (*info(a,b)*) as requested, or provide no information (*info_null(a,b)*) if it has time to do so, hoping that agent b might come up with a better offer, or simply disagrees (*abort(a,b)*).

Thus, an initiating agent will always negotiate until it has run out of time or when the responding agent opts out. However, an initiating agent may abort a negotiation, and this is where the conditions come into play. If the agent realizes that it has already obtained what it wants from other negotiations happening in parallel, then it aborts the current negotiation. If the agent realizes that it no longer cares about the current negotiation, then it aborts. These conditions are based on desires and intentions, which, in turn, are based on beliefs of the agent. When an agent runs out of time, it issues an *OUT_OF_TIME* message to the other agent and quits the negotiation with a failure (state 6, *out_of_time*). When an agent aborts, it issues an *ABORT* message to the other agent and quits the negotiation with a failure (state 7, *abort*). Finally, whenever an agent does not hear from the other agent within an allowed time period, it assumes that the communication channel has been jammed or congested, and then it quits with a failure (state 8, *channel_jammed*).

Note that we distinguish *NO_GO*, *STOP*, *OUT_OF_TIME*, and *ABORT* in the above protocol. With the above different end states, agent a can determine whether the negotiation has failed, because it has exhausted all its arguments (*STOP*). Otherwise, it subsequently learns from the failure.

4.3 Case-Based Reasoning (CBR)

A case contains a problem description, a solution, and an outcome. The problem description 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, target type, etc.), and the model of the potential coalition members (how many, the number that actually were used in negotiation, their

capabilities, etc.) at the time before negotiation is to take place. The solution is a negotiation strategy (Section 1.4.1).

Case Selection and Retrieval: The CBR component of each agent evaluates cases using weighted matching and different matching functions for different attributes. After evaluation, the most similar cases will be selected. However, if there is more than one case with the same similarity score, the CBR component then compares the outcome of the negotiations and selects the case with the best outcome.

Case Adaptation: Given the set of negotiation strategies from the best case, the CBR component adapts the parameters based on the difference between the new case and the best case and also based on the outcome of the best case. Since each case is situated, the set of negotiation parameters learned from the best case might not be applicable in the current case. Hence, the CBR component modifies the parameters based on the perceived 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 in the retrieved case, then the agents would want to use less CPU resources.

Furthermore, the CBR component modifies the parameters based on the outcome of the best case. If the negotiation of the best case failed, and this failure 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 resources, then the agent asks for more CPU. In this manner, agents are able to learn from their experiences a ”good-enough, soon-enough” set of negotiation strategies, and they are able to learn how to avoid repeating past failures.

Case Storage and Learning: After a negotiation is completed (successfully or otherwise), the agent delegates the case to the CBR component for storage and learning. See Section 1.4.4.2 for more on this topic. The heuristics we use to evaluate the similarity of two cases during retrieval are very similar to those we use to evaluate the difference between two cases when determining whether a new case should be learned or not. During retrieval, we use weighted normalized rules to score each case in the casebase and to select the most similar case based only on the situated parameters. If there are two or more such cases, we select the case with the best outcome to increase the utility of the retrieval. During the learning phase, in addition to situation parameters, we evaluate the solution parameters (i.e.,, the negotiation strategy parameters) and the outcome. The heuristics are weighted and normalized rules as well.

4.4 Learning

Learning is critical in our coalition formation architecture.In our system, the coalition solutions are only satisficing and not optimal. Without learning, our

agents will not be able to improve on their coalition formation skills, and they will not be able to produce coalitions that are closer to the optimal in terms of the resources used.

In our model, there are two levels of learning. First, every agent evaluates the utility of its coalition candidates via reinforcement learning of past negotiation outcomes and behaviors. As a result of this learning, an agent assigns its task requirements differently and approaches candidate agents in different orders, based on what it has learned from its interactions with its neighbors in the past. Second, each agent uses a CBR mechanism to learn useful negotiation strategies that guide how negotiations should be executed in the future. Furthermore, an agent also learns from its past relationship with a particular neighbor when conducting a negotiation with that neighbor. The collaborative learning behavior allows two negotiation partners to reach a deal more effectively, and agents to form better coalitions faster.

4.4.1 Learning to Form Coalitions Better. Our coalition formation strategy is opportunistic, because it tries to form satisficing solutions with no guarantees. Indeed, a coalition formation cannot be guaranteed since the communication medium that is available to the agents does not guarantee delivery of each outgoing message. In particular, messages sent in this environment may be lost, corrupted, or jammed. To improve the chance of successfully forming coalitions, our agents employ a learning mechanism aimed at enabling them to learn to form better coalitions faster in such uncertain environments. When a coalition is initially formed, an agent is motivated to go back to the same neighbor (for a particular task) that the agent has had productive relationship in the past. Hence the reinforcement learning, which strengthens past good relationships with other agents and weakens those that have not been as productive. When an initiating agent sends over different classes of information to argue with the responding agent, one of the information classes includes a profile of the neighbor including its past relationship with the responding agent. As a result, the responding agent may have a reinforced motivation to agree to a negotiation based on its previous interactions with the initiating agent. The more successes the initiator has had with a particular responder, the more effectively it can argue with that neighbor due to its reinforcement-learned attributes.

We also use CBR to retrieve and adapt negotiation strategies for negotiations. When a negotiation is completed, the agent also determines whether to learn the new case. The case-based learning is performed in two modes, namely, in incremental and in refinement mode. During incremental learning, an agent matches the new case to all cases in the casebase, and if that new case is significantly different from all other stored cases, it stores that new case in its casebase. When we want to keep the size of the casebase under control, we

use refinement learning, replacing the most similar old case with the new case, if the replacement will increase the *diversity* of the case base.

4.4.2 Learning to Negotiate Better. Before a negotiation can take place, an agent has to define its negotiation strategy, which it does using the retrieved most similar case. Note that a better negotiation does not mean one that always results in an accepted deal; a better negotiation is one that is effective and efficient, meaning that a quick successful negotiation is always preferred, but if a negotiation is to fail, then a quickly failed negotiation is also preferred. To formulate a new task, an agent composes a new problem case with a situation space. Then it retrieves the most similar case from its case-base using a weighted parametric matching on the situation spaces. Given the most similar case, the agent adapts that solution or negotiation strategy to more closely reflect the current situation. Equipped with this modified negotiation strategy, the agent proceeds with negotiation. Finally, when the negotiation completes, the agent updates the case depending on the outcome.

Refinement and Incremental Learning: After a negotiation is completed (successfully or otherwise), the agent updates its casebase using an incremental and a refinement learning step. During incremental learning, the agent matches the new case to all cases in the casebase, and, if the new case is significantly different from all other stored cases, then it stores the new case. When the agent computes the difference between a pair of cases, it emphasizes more the case description than the negotiation parameters since its objective is to learn cases that cover as much of the problem domain as possible. This kind of learning improves the agent's future case retrieval and case adaptation. So, an agent learns good and unique cases incrementally.

We define the case difference measure as

$$Diff(a,b) = \frac{w_{sit}\sum_{i=1}^{M} w_{sit,i}\left|f^{i}_{sit,a} - f^{i}_{sit,b}\right| + w_{sol}\sum_{i=1}^{N} w_{sol,j}\left|f^{j}_{sol,a} - f^{j}_{sol,b}\right|}{w_{sit}\sum_{i=1}^{M} w_{sit,i} + w_{sol}\sum_{i=1}^{N} w_{sol,j}}$$

where $f_{sit,k}i$ is the ith situation feature for case k, and $f_{sol,k}j$ is the jth solution feature for case k. The term $w_{sit,i}$ is the weight for $f_{sol,k}i$, and $w_{sol,j}$ is the weight for $f_{sol,k}j$. The term w_{sit} is the weight for the situation space, and is the weight for the solution space. Also, $w_{sit} + w_{sol} = 1$, and each w_{sit} or w_{sol} is between 0 and 1. Finally, the agent computes

$$\min_{b=1\rightarrow K} Diff(a,b)$$

for the new case a, and if this number is greater than a pre-determined threshold, then the case is learned.

Since we want to keep the size of the casebase in check, especially to enable quick case retrieval and low-cost maintenance , the agent performs refinement learning. If the new case is found to be very similar to one of the existing cases, then it computes the sum of differences (e.g., $\sum Diff(a, b)$) between that old case and the entire casebase minus the old case. It also computes the sum of differences between the new case and the entire casebase minus the old case. These computations establish the *utility* of the new case and the old case (that the agent considers to replace with the new case). The agent chooses to keep the case that will increase the *diversity* of the case base. Thus, if the second sum is greater than the first one, then the agent replaces the old case with the new one. In this manner, we gradually refine the cases in the casebase while keeping its size under control.

5. Coalition Acknowledgment

After a coalition has been agreed upon, meaning that all negotiation tasks have now terminated, the agent has to *acknowledge* the coalition. If the coalition is a failure, then the agent has to send out a *discard* message to each coalition member that has agreed to a deal, to cancel the deal. If the coalition is a success, then the agent must send out a *confirm* message to the same coalition members. This acknowledgment is necessary for effective task planning. Coalition acknowledgment enables responsible coalition formation: First, an initiating agent is responsible, in that it informs coalition members of the success or failure of the coalition and releases the coalition members from their agreements in case of a coalition failure caused by other negotiations. Second, a responding agent may unilaterally release itself from its agreement if it does not receive a confirmation of an agreed task. This coalition acknowledgment step is a feature that allows the initiator to conduct concurrent negotiations asynchronously.

6. Experimental Results

In this section, we present results from three sets of experiments that we have conducted with our multiagent system applied to the domain of multisensor tracking. In the first experiment, we studied the role of the case-based negotiation strategy in negotiation. In the second experiment we investigated the effectiveness of our coalition formation model. In the third experiment we examined the impact of learning in coalition formation and negotiation effectiveness.

6.1 Case-Based Negotiation Strategy

The experiments we ran concentrated on evaluating whether negotiation would improve target tracking performance, and whether CBR resulted in bet-

ter negotiation strategies. One of our hypotheses was that negotiating agents would track targets better since they are able to coordinate radar measurements among multiple agents and achieve improved triangulation as a result of this coordination that results from negotiation. Another hypothesis was that negotiation using CBR would result in better tracking than using a static negotiation protocol, since CBR would allow dynamic adaptation of the strategy to the current situation. Our experiments confirm these hypotheses. In addition to the accuracy of tracking, we used aspects of intra-agent communication as a measure of quality of performance, since communication is a potential bottleneck in scaling-up a multiagent system. For the contribution of communication to the overall performance of our system, we used the length of messages, the frequency of the type of messages, and a metric called *message cost*, which is defined as message length times message frequency.

We compared our system to a multiagent sensor controlling network where there is no communication among sensor-controlling agents for negotiation or other purposes, except for the communication between each sensor-controlling agent and a special agent, the tracker, to which sensing data is sent for triangulation. Therefore, in this setup, all that each agent does is to make measurements of a target it detects in its sensor's coverage area and send these measurements to the tracker agent. 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 1.2–1.5, were very encouraging.

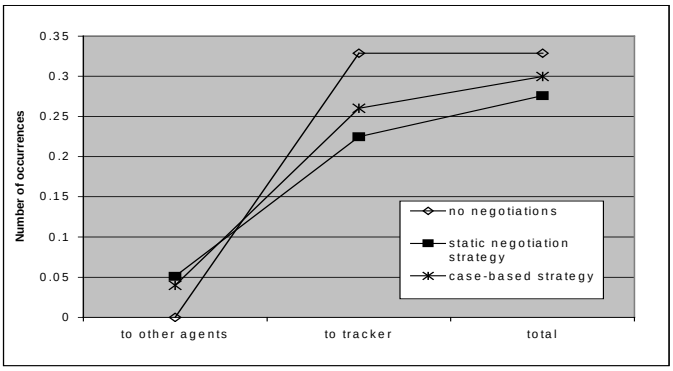

Figure 1.2. Tracking accuracy vs. agent behavior

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 measurements to the tracking software. Since there was no coordination of measurements

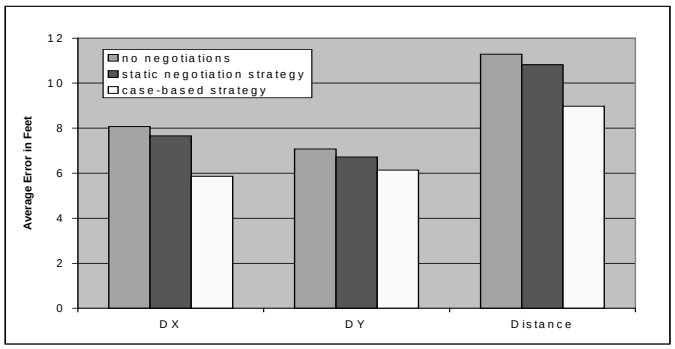

Figure 1.3. Number of messages to agents and to tracking software vs. agent behavior

among measuring agents, 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 a lower message cost (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 tracking accuracy was poor. 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 ranking leads to more effective communication acts overall. The accuracy is due to the fact that case-based agents adapt their negotiation strategy to the current situation and have a higher chance of achieving agreement for resource allocation. On the other hand, agents using the static strategy failed to agree more often, and this led to failure to perform simultaneous measurements that are required for most accurate tracking.

Overall, our agents exhibit all of the behavior described. They use CBR to select and adapt a negotiation strategy, they have time and system awareness, negotiate for sensor 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 multiple sensors.

6.2 Coalition Formation

In this experiment, the total number of attempts to form a coalition was 150. The total number of coalitions successfully formed (after coalition finalization) was 30, or 20%. The total number of coalitions confirmed by all

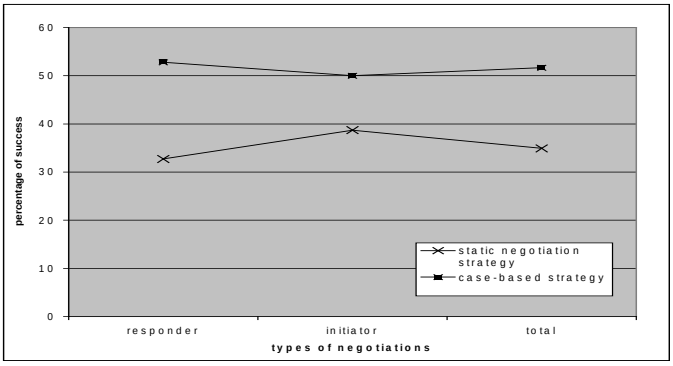

Figure 1.4. Message statistics vs. agent behavior. Message cost is the product of the average length and the total number of messages sent per second

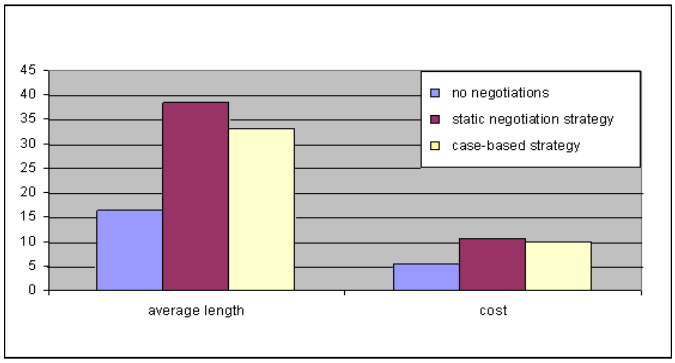

Figure 1.5. Percentage of successful negotiations vs. negotiation strategy type. A successful negotiation is one that completes with a deal between the two negotiating agents

coalition members was 26, or 86.7% of all successfully formed coalitions. Finally, the total number of coalitions executed on time was 18, or 61.5% out of all successfully confirmed coalitions.

First, the percentage of successfully formed coalitions was only 20.0%. Out of the 120 failed attempts, 86 (71.7%) of them were caused by one of the coalition members outright refusing to negotiate, 17 (14.2%) were caused by the communication channels being jammed, and 17 (14.2%) were caused by busy negotiation threads. When an agent initiates a negotiation request to a candidate and that candidate immediately refuses to entertain the negotiation, it can be due to the responding agent not having idle negotiation threads, or being unable to place the requested task into its job queue. Thus, we expect this failure rate to decrease once we increase the number of negotiation threads allocated per agent. When an agent fails to send a message to another agent, or fails

to receive an expected message, we label this as a communication "channel-jammed" problem. When an initiating agent fails to approach at least two candidates, it immediately aborts the other negotiation process that it has invoked for the same coalition. This causes the coalition to fail. Second, the probability of a successfully formed coalition getting confirmed completely was 86.7%. For each coalition successfully formed, three confirmations were required. Out of 30 coalitions, 4 coalitions were confirmed only by two of the members. There were two reasons for this:

- The acknowledgment message sent out by the initiating agent was never received by the responding agent expecting a confirmation.
- The agreed task had been removed from the job queue before the confirmation arrived.

The failure of message reception could be due to jammed communication channels or due to random message loss due to the unreliability of the communication channel The second failure occurred because of a contention for a slot in the job queue by two separate tasks. Now, suppose that both negotiations are successful. The negotiation between A and B hence ends first and then that between A and C. When the first negotiation ends, agent A adds the task requested by B to the job queue. Immediately after, when the second negotiation also ends successfully, agent A adds the second task, requested by C to the job queue, and this causes the second task to replace the first task. This is a problem with over-commitment.

Third, the probability of a confirmed coalition getting executed was 61.5%. Out of 26 coalitions confirmed, only 16 of them were executed completely. Of the 10 failures, there were two cases where none of the members executed its planned task. There was one case where only one of the members executed, and seven cases where only two members executed. Based on the above results, we are investigating solutions to address the following problems in our multiagent system:

- *Channel Jammed:* To prevent negotiation messages from getting lost and holding up negotiation threads, a better use of the available communication channels is needed. This will increase significantly the chance for response and acknowledgment messages to be received on time, and, in turn, the success rate of coalition formation.
- *Task Contention and Over-Commitment:* Currently, if an agent is approached by two other agents for two separate tasks around the same time slot, the agent entertains both requests and may run into task contention in its job queue and over-commitment.

- *Time Modeling:* Sensor-related tasks must be modeled more closely to help plan and schedule a compact and sensible job queue. We have performed time profiling on various calls and have found that sensor-related tasks have a high variance in their execution duration. We need to determine the bounds such that a tracking task can be safely scheduled and expected to be executed.

6.3 Experiments with Learning

We ran two basic experiments to examine the impact of learning in our agents. In the first, we investigated the effect of reinforcement learning in the quality of the resulting coalition. In the second, we analyzed the quality of the negotiation as a function of learning new cases of negotiation strategies. We concluded that the quality was based on the number of successful negotiations, since when the agents reach a negotiated deal to jointly track a target, the overall system utility increases. Initial results indicate that the agents form coalitions with partners who are more willing to accommodate them in negotiation, and that the cases learned are being used in future negotiations. Agents that use learning of negotiation cases have between 40% and 20% fewer failed negotiations. Agents that use reinforcement learning to determine future coalition partners tend to prefer neighbors who are more conceding.

We also conducted experiments with four versions of learning: (1) both case-based learning and reinforcement learning (CBLRL), (2) only reinforcement learning (NoCBL), (3) only case-based learning (NoRL), and (4) no learning at all (NoCBLRL). Figure 1.6 shows the result in terms of the success rates for negotiations and coalition formations. As the graph indicates, the agent design with both case-based learning and reinforcement learning outperformed others in both its negotiation success rate and coalition formation success rate. That means that, with learning, the agents were able to negotiate more effectively (and perhaps more efficiently as well) that led to more coalitions formed. Without either case-based learning or reinforcement learning (but not both), the negotiation success rates remained about the same but the coalition formation rate tended to deteriorate. This indicates that, without one of the learning mechanisms, the agents were still able to negotiate effectively, but may be not efficiently (resulting in less processing time for the initiating agent to post-process an agreement). Without both learning mechanisms, there was significant drop in the negotiation success rate. This indicates that the learning mechanisms helped improve negotiation performance. Unfortunately, the improvement achieved by learning, although present, is small and does not seem as significant as we had initially hypothesized.

The results reported in this chapter need to be scrutinized further to isolate learning, monitoring, detection, reasoning, communication, and execution

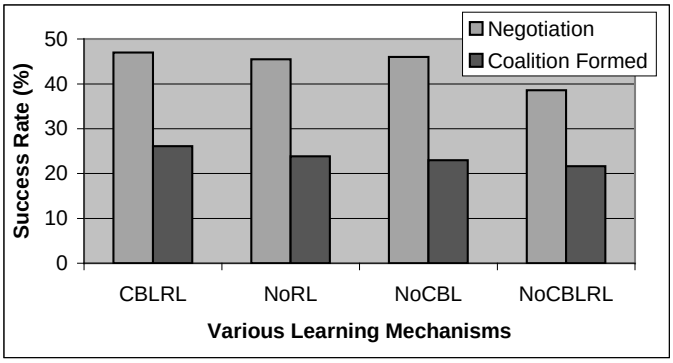

Figure 1.6. Success rates of negotiations and coalition formations for different learning mechanisms

components in the evaluation: An agent that is busy tracking will not entertain a negotiation request that requires it to give up the resources that it is currently using to track. That refusal leads to a failure on the initiator's side. Another point worth mentioning is the realtime nature of our system and experiments. Added learning steps may cause an agent to lose valuable processing time to handle a coalition formation problem.More frequent coalition formations may prevent other negotiations to proceed as more agents will be tied up in their scheduled tasks. Our current work is examining how case maintenance strategies can be used to *prune* the learned case base from deleterious cases, and thus improve the performance of learning.

7. Related Work

7.1 Coalition Formation

A definition for rational coalition is given by Kahan and Rapoport. This definition states that a coalition game is based on the total utility that the member of the coalition can achieve by coordinating and acting together, assuming that information is complete [Kahan and Rapoport 1984]. Our problem domain is not superadditive in which a merged coalition of any pair of sub-coalitions is better than any pair of sub-coalitions operating individually as we have to consider coalition formation costs such as communication and computation costs. Furthermore, sub-additivity does not apply to our model.

Shehory et al. relax some of the restrictive assumptions of theoretical coalition formation algorithms for a real-world system [Shehory et al. 1997]. In their model, each agent has a vector of real non-negative capabilities. Each capability is a property of an agent that quantifies its ability to perform a specific type of action and is associated with an evaluation function. The authors'

model assumes that all agents know about all of the tasks and the other agents. In our model, an initiating agent knows only the agents in its neighborhood and knows partially about the updated status of a selective subset of its neighbors after negotiation. The details of intra-coalitional activity are not necessary for agents outside of the coalition in the author's model [Shehory et al. 1997]. On the contrary, in our model, an agent performs a task contributing to that coalition and the execution of this is reflected in the agent's commitments, constraints, and perceptions. Shehory and Kraus further extend the work by Shehory et al. [Shehory et al. 1997] by incorporating negotiations, computational and communication costs [Shehory and Kraus 1998]. This model is similar to ours. However, our model allows an agent to conduct multiple concurrent negotiations, and adjusts its negotiation strategies to redesign its coalition.

Sandholm and Lesser introduce a bounded rationality in which agents are guided by performance profiles and computation costs in their coalition formation process [Sandholm and Lesser 1995]. In traditional coalition formation, a rational agent can solve the combinatorial problem optimally without paying a penalty for deliberation. In our model, the agents do not pay a penalty per se. Instead, the agents will feel the impact of poor coalition formation and negotiation processes. If a coalition is poorly designed, it may conflict with the number of available negotiation threads that an agent has. If a negotiation strategy is poor, then the agent may have to abort the negotiation due to the realtime hard limit on the particular negotiation process. Indeed, the design of our model is driven by bounded rationality of time and resource constraints.

Tohme and Sandholm study coalition formation among self-interested agents that cannot make side-payments, that is, agents reward each other with payments for agreement to join some coalition, making the evaluation of a coalition solely on its utility [Tohme and Sandholm 1999].

Sen and Dutta propose an order-based genetic algorithm as a stochastic search process to identify the optimal coalition structure [Sen and Dutta 2000]. A significant difference between this work and ours is the scope of coalition formation. The authors' algorithm searches for an optimal coalition structure, which consists of all the agents in the environment grouped into one or more coalitions. Our model, however, focuses on the formation of a single coalition for a particular event while allowing multiple coalitions to be formed concurrently.

Other work in coalition formation include [Zlotkin and Rosenschein 1994, Ketchpel 1994, Klusch and Shehory 1996, Sandholm et al. 1999, Moon and Stirling 2001].

7.2 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: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. There has been work in off-line learning of negotiation strategies using genetic algorithms [Matos et al. 1998] in a service-oriented environment.

8. Conclusions

In this chapter, we described a coalition formation architecture that aims at obtaining satisficing solution for time-critical, noisy, and incomplete resource or task allocation problem. Because of the nature of the strategy, a coalition is not guaranteed to form successfully especially when message passing among agents is unreliable. To offset this unreliability, our architecture incorporates learning.

In our approach, our coalition formation process is divided into three stages: initial coalition formation, coalition finalization, and coalition acknowledgment. Initially, coalition candidates are selected from an agent's neighborhood and subsequently ranked according to their respective potential utilities. Next, during the finalization phase, the coalition is refined and verified through negotiations, where information is exchanged between two agents to clarify commitments and constraints. The agent is able to coordinate directly and indirectly through a coalition awareness link with its negotiation threads. Finally, the coalition acknowledgment step confirms or discards already-agreed requests. This releases an agent from needlessly honoring a lost-cause coalition commitment. We have incorporated utility theory, case-based reasoning, argumentative negotiation, and realtime execution in the above methodology and design.

We have built a multiagent system complete with end-to-end agent behavior. Our preliminary results are promising in that an initiator was able to form satisficing coalitions quickly. Our results also show that we need to improve the management of communication channels, handle task contention and

over-commitment, and model domain-related time constraints better. We also demonstrated 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 are reflective of the system-level resources they use and time-aware.

Finally, we demonstrated 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 agents to schedule the precise time of measurement and also exchange computational resources, leading to faster and more efficient processing. Overall, our results show that our agents are able to form coalitions quickly and in time to track a moving target in the environments. The agents were able to negotiate, plan synchronized tracking tasks and execute them accordingly.

There are also several areas that we are investigating actively:

1 Inter-coalition and intra-coalition competitions – task distribution, priorities, ”health” of coalitions, etc.

2 Coalition awareness and the effects of coalition monitoring on speed (how much should a negotiation process monitor about the coalition when negotiating, and how reflective we want the negotiations to be of the coalition)

3 Online learning of better coalition formation strategies through distributed cooperative case-based learning and by case base maintenance

Acknowledgments

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

Brazier, F. and J. Treur (1996). Compositional Modeling of Reflective Agents, in Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW’96).

Chavez, A. and P. Maes (1996). 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.

Jennings, N. R., S. Parsons, P. Noriega, and C. Sierra (1998). On Argumentation-Based Negotiation, in Proceedings of International Workshop on Multi-Agent Systems (Boston, MA).

Kahan, J. P. and A. Rapoport (1984). Theories of Coalition Formation, Lawrence Erlbaum.

Ketchpel, S. (1994). Forming Coalitions in the Face of Uncertain Rewards, in Proceedings of the AAAI'94, Seattle, WA, July, 414–419.

Klusch, M., and O. Shehory (1996). Coalition Formation among Rational Information Agents, in Proceedings of the MAAMAW'96, Eindhoven, Netherlands, 22–25.

Kraus, S. (1997). Beliefs, Time, and Incomplete Information in Multiple Encounter Negotiations among Autonomous Agents, Annals of Mathematics and Artificial Intelligence, 20(1–4):111–159.

Kraus, S., K. Sycara, K., and A. Evenchik (1998). Reaching Agreements through Argumentation: a Logical Model and Implementation, AI Journal, 104(1–2):1–69.

Kraus, S., J. Wilkenfeld, and G. Zlotkin, G. (1995). Multiagent Negotiation under Time Constraints, Artificial Intelligence, 75:297–345.

Laasri, B., H. Laasri, S. Lander, and V. Lesser (1992). A Generic Model for Intelligent Negotiating Agents, International Journal of Intelligent & Cooperative Information Systems, 1:291–317.

Matos, N., C. Sierra, and N. R. Jennings (1998). Negotiation Strategies: an Evolutionary Approach, in Proceedings of International Conference on Multiagent Systems (ICMAS'98), Paris, France, July 4–7, 182–189.

Moon, T. K. and W. C. Stirling (2001). Satisficing Negotiation for Resource Allocation with Disputed Resources, in Working Notes of 2001 Fall Symposium Series on Negotiation Methods for Autonomous Cooperative Systems, North Falmouth, MA, November, 106–115.

Rosenschein, J. S., and G. Zlotkin (1994). Designing Conventions for Automated Negotiation, AI Magazine, 15(3):29–46.

Sandholm, T. W., K. Larson, M. Andersson, O. Shehory, and F. Tohme (1999). Coalition Structure Generation with Worst Case Guarantees. Artificial Intelligence, 111(1–2):209–238.

Sandholm, T. W. and V. R. Lesser (1995) Coalition Formation amongst Bounded Rational Agents, in Proceedings of IJCAI 1995, Montreal, Canada, 662–669.

Sen, S. and P. S. Dutta (2000). Searching for Optimal Coalition Structures, in Proceedings of the ICMAS'00, Boston, MA, July, 286–292.

Shehory, O. and S. Kraus (1998). Methods for task allocation via agent coalition formation, Artificial Intelligence, 101(1–2):165–200.

Shehory, O. M., K. Sycara, and S. Jha (1997). Multi-Agent Coordination through Coalition Formation, in Proceedings of the ATAL'97, Providence, RI.

Soh, L.-K. and C. Tsatsoulis (2001a). Reflective Negotiating Agents for Real-Time Multisensor Target Tracking, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01), August 6–11, Seattle, WA, 1121–1127.

Soh, L.-K. and C. Tsatsoulis (2002a). Real-Time Satisficing Multiagent Coalition Formation, in Working Notes of the AAAI Workshop on Coalition Formation in Dynamic Multiagent Environments, Edmonton, Alberta, Canada, July 28–August 1, 7–15.

Soh, L.-K. and C. Tsatsoulis (2002b). Satisficing Coalition Formation among Agents, in Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, July 15–19.

Soh, L.-K. and C. Tsatsoulis (2002c). Allocation Algorithms in Dynamic Negotiation-Based Coalition Formation, in Working Notes of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems Workshop on Teamwork and Coalition Formation, Bologna, Italy, July 15–19.

Tambe, M. (1997). Towards Flexible Teamwork, Journal of Artificial Intelligence Research, 7:83–124.

Tohme, F. and T. Sandholm (1999). Coalition Formation Process with Belief Revision among Bounded-Rational Self-Interested Agents, Journal of Logic and Computation, 9(6): 793–815.

Wooldridge, M. and N. Jennings (1995). Intelligent Agents: Theory and Practice, Knowledge Engineering Review, 10(2):114–152.

Zeng, D. and K. Sycara (1998). Bayesian Learning in Negotiation, International Journal of Human-Computer Studies, 48:125–141.

Zlotkin, G. and J. S. Rosenschein (1994). Coalition, Cryptography and Stability: Mechanisms for Coalition Formation in Task Oriented Domains, in Proceedings of the AAAI'94, Seattle, WA, July, 432–437.

Zlotkin, G. and J. S. Rosenschein (1996). Mechanism Design for Automated Negotiation, and Its Application to Task Oriented Domains, Artificial Intelligence, 86(2):195–244.

Index