An Integrated Multilevel Learning Approach to Multiagent Coalition Formation

Content Areas: multiagent systems, machine learning, case-based reasoning

Abstract

In this paper we describe an integrated multilevel learning approach to multiagent coalition formation in a real-time environment. In our domain, agents negotiate to form teams to solve joint problems. The agent that initiates a coalition shoulders the responsibility of overseeing and managing the formation process. A coalition formation process consists of two stages. During the first stage, the initiating agent identifies the candidates of its coalition, i.e., known neighbors that *could* help. The initiating agent negotiates with these candidates during the finalization stage to determine the neighbors that *are willing* to help. Since our domain is dynamic, noisy, and time-constrained, our coalitions are not optimal. However, our approach employs learning mechanisms at several levels to improve the quality of the coalition formation process. At a tactical level, we use reinforcement learning to identify viable candidates based on their potential utility to the coalition, and case-based learning to refine negotiation strategies. At a strategic level, we use distributed, cooperative case-based learning to improve general negotiation strategies. We have implemented the above three learning components and conducted experiments in multisensor target tracking and CPU re-allocation applications.

1 Introduction

Multiagent coalition formation is important for distributed applications ranging from electronic business to mobile and ubiquitous computing where adaptation to changing resources and environments is crucial. It increases the ability of agents to execute tasks and maximize their payoffs. Moreover, coalitions can dynamically disband when they are no longer needed or effective. Thus the automation of coalition formation will not only save considerable labor time, but also may be more effective at finding beneficial coalitions than human in complex settings [Jennings, 2002].

Although considerable research has been conducted either in coalition formation among self-interested agents (e.g., [Tohme and Sandholm, 1999], [Sandholm *et al.*, 1999], [Sen and Dutta, 2000]), or in coalition formation among cooperative agents (e.g., [Shehory *et al.*, 1997]), little work has been done in coalition formation among both self-interested and cooperative agents. Furthermore, there have been no attempts to study coalition formation among such agents in a dynamic, real-time, uncertain, and noisy environment, which is a typical real-world environment and in which a sub-optimal coalition needs to be formed in a real-time manner.

Here we propose an integrated multilevel learning approach to multiagent coalition formation. In our approach, agents are assumed to be cautiously cooperative—they are willing to help only when they think they benefit from it—and honest. However, due to the noisy, uncertain, dynamic and real-time nature of our domain, not every agent can be correct in its perceptions and assumptions. Thus, to achieve a coalition, an initiating agent has to negotiate with other agents. Through concurrent, multiple 1-to-1 negotiations, the initiating agent identifies the agents that are willing to help. The formation process is successful if the initiating agent successfully persuades enough agents to join the coalition.

Note that in this paper, we focus on improving the quality of the coalition formation process, and not on the quality of the coalition after it is formed and executed.

Note also that our approach is an example of the "good enough, soon enough" design paradigm. In our domain, an agent has incomplete information about the environment, the task execution is time constrained, and the communication between agents is not reliable, so an optimal coalition formed from the deep learning is impractical. Thus, a sub-optimal yet fast coalition formation process is warranted.

2 Coalition Formation

Figure 1 depicts our coalition formation modules that make up the two stages: initialization and finalization. The feasibility study and the ranking of candidates are the initialization stage whereas the negotiations and their management the finalization stage. This two-stage model [Soh and Tsatsoulis, 2001] allows an agent to form an initial coalition *hastily* and *quickly* to react an event and to *rationalize* to arrive at a working final coalition as time progresses. Here we briefly describe each module of the design:

(1) Dynamic profiling: Every agent dynamically profiles each neighbor as a vector in the agent about the negotiation relationship between them, and profiles each negotiation task as a case in the casebase about the negotiation strategy description and negotiation outcome.

(2) Feasibility study: This module analyzes the problem and computes (a) whether the agent has the resources to do something about it, and (b) if yes, the list of agents that the agent thinks could help.

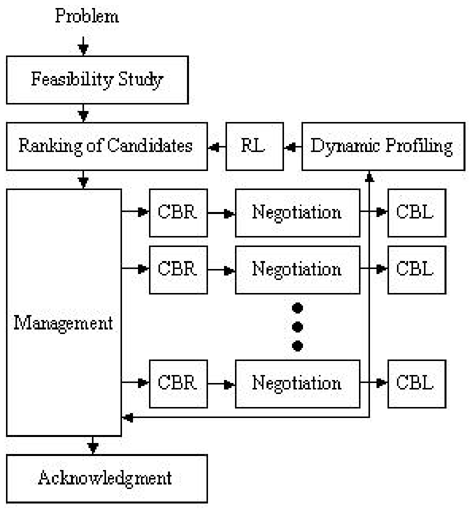

Figure 1. An overview of our coalition formation process

(3) Ranking of Candidates: This module scores and ranks each candidate, and proportionately assigns the requested demand to each candidate, based on its *potential utility* (section 3.1).

(4) Management: This module initiates negotiations with top-ranked candidates. That is, the module manages multiple, concurrent 1-to-1 negotiations. For each negotiation task, it first finds a negotiation strategy through CBR. Then, it spawns a thread to execute that negotiation task. The module oversees the various negotiation threads and modifies the tasks in real-time. For example, the module will terminate all remaining negotiations once it finds out that it no longer can form a viable coalition. The module will reduce its requests or demands once it has secured agreements from successful negotiations. And so on. In effect, this management simulates a 1-to-many negotiation.

(5) CBR: Given the problem description of a task, the CBR module retrieves the best case from the casebase, and adapts the solution of that best case to the current problem. This is based on the work of [Soh and Tsatsoulis, 2001].

(6) Negotiation: Our negotiation protocol is argumentative. The initiating agent provides evidence for its request to persuade the responding agent. The responding agent evaluates these evidence pieces and if they are higher than a dynamic persuasion threshold, then the responding agent will agree to the request. The responding agent also has the ability to counter-offer due to time constraints or poor evidence. This is based on the work of [Soh and Tsatsoulis, 2001].

(7) Acknowledgment: Once all negotiations are completed, if a coalition has been formed, the agent confirms the success of the coalition to all agents who have agreed to help. If the agent has failed to form a coalition, it informs the agents who have agreed to help so they can release themselves from the agreements.

In the next section, we will discuss the learning mechanisms, a critical part of our coalition formation approach.

3 Learning

Our learning approach incorporates reinforcement learning and case-based learning at two levels. At a tactical level, we use reinforcement learning to identify viable candidates based on their potential utility to the coalition, and case-based learning to refine specific negotiation strategies. At a strategic level, we use distributed, cooperative case-based learning to improve general negotiation capabilities.

3.1 Reinforcement Learning

Reinforcement learning is evident at the coalition initialization and finalization stages. During initialization, the initiating agent measures the *potential utility* of a candidate based on a weighted sum of (1) its past cooperation relationship with the initiator such as the candidate's helpfulness, friendliness, and the agent's helpfulness and importance to the candidate, (2) its current cooperation relationship with the initiating agent such as whether the two agents have already been negotiating about other problems, and (3) its ability to help towards the current problem. An initiating agent thus will more likely approach the agents that have been helpful before, thus reinforcing the cooperation relationship among them.

During finalization, an initiating agent also appeals to a candidate about how helpful the initiating agent has been in the past. A candidate is more easily persuaded if it realizes that a particular agent has been helpful in the past, and thus once again reinforcing their cooperation relationship.

3.2 Case-Based Learning

We use CBR to retrieve and adapt negotiation strategies for negotiations during coalition finalization. We also equip our CBR module with both individual and cooperative learning capabilities (Figure 2). Individual learning refers to learning based on an agent's perceptions and actions, without direct communication with other agents. This learning approach allows an agent to build its casebase from its own experience, eventually forming its own *area of specialization*. Cooperative learning refers to learning other agents' experience through interaction among agents. When an agent identifies a problematic case in its casebase, it approaches other agents to obtain a possibly better case.

There has been research in distributed and cooperative CBR. [Prasad and Plaza, 1996] proposed treating corporate memories as distributed case libraries. Resource discovery was achieved through (1) negotiated retrieval that dealt with retrieving and assembling case pieces from different resources in a corporate memory to form a good overall case, and (2) federated peer learning that dealt with distributed and collective CBR [Plaza and Arcos, 1997]. [Martin *et al.*, 1999] extended the model using the notion of competent agents. [Martin and Plaza, 1999] employed an auction-based mechanism that focused on agent-mediated systems where the best case was selected from the bid cases.

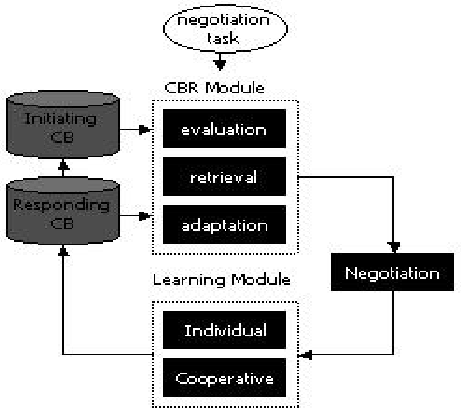

Figure 2. The relationship between case learning and CBR as well as negotiation tasks in an agent

Our methodology employs a cautious utility-based adaptive mechanism to combine the two learning approaches, an interaction protocol for soliciting and exchanging information, and the idea of a chronological casebase. It emphasizes individual learning and only triggers cooperative learning when necessary. Our cooperative learning also differs from collective CBR in that it does not merge case pieces into one as it considers entire cases. In addition, our research focus here is to define a mechanism that combines individual and cooperative learning.

Note that the communication and coordination overhead of cooperative learning may be too high for cooperative learning to be cost-effective or timely. Moreover, since an agent learns from its own experience and its own view of the world, its solution to a problem may not be applicable for another agent facing the same problem. This injection of *foreign* knowledge may also be risky as it may add to the processing cost without improving the solution quality of an agent [Marsella *et al.*, 1999].

3.2.1 Chronological Casebase and Case Utility

We have utilized the notion of a chronological casebase in which each case is stamped with a time-of-birth (when it was created) and a time-of-membership (when it joined the casebase). All initial cases are given the same time-of-birth and time-of-membership. In addition, we profile each case's usage history (Table 1). An agent evaluates the utility of a case based on its usage history. If the case has a low utility, it may be replaced (or forgotten). If the case is deemed problematic, then a cooperative learning will be triggered and the case will be replaced. Table 2 shows the heuristics we use in tandem with the chronological casebase. When a negotiation completes, if the *new* case is useful to add the casebase's diversity, the agent learns it. If the casebase's size has reached a preset limit, then the agent considers replacing one of the existing cases with the new case. For our individual case-based learning, we use heuristics *H1*, *H2*, and *H3*.

Parameters	Description
TU	The number of times the case has been used
TSU	The number of times the case has been used in a successful negotiation
TINC	The number of times the usage of the case has led to a new case getting added to the casebase
TR	The number of times the case has been designated as a problematic case, i.e., with very low utility
TS	The last time that the case was used or the time when the case was added

Table 1. The usage history that an agent profiles of each case

Heuristics	Description
H1 Currency	If a case has not been used in a long time, then this case is more likely to be replaced.
H2 Evolution	With everything else equal, an old case is more likely to be replaced than a young case.
H3 Usefulness	If a case's *TSU* is significantly small, then the case is more likely to be replaced.
H4 Solution Quality I	If a case has a high *TU* but a low *TSU*, then it is a problematic case.
H5 Solution Quality II	If a case has a low *TSU*, and a high *TINC*, then the solution of this case is probably not suitable for the problems encountered by the agent and it is a problematic case.
H6 Persistence	The urgency to trigger a cooperative learning process is directly dependent on the case's *TR*, until a pre-determined threshold is reached.
H7 Ignorance	If a case's *TR* has reached a pre-determined threshold and it is still not replaced, then the problematic case is removed without replacement.

Table 2. Heuristics that support the chronological casebase

3.2.2 Cooperative Learning

Figure 3 depicts our cooperative learning design. We adhere to a cautious approach to cooperative learning:

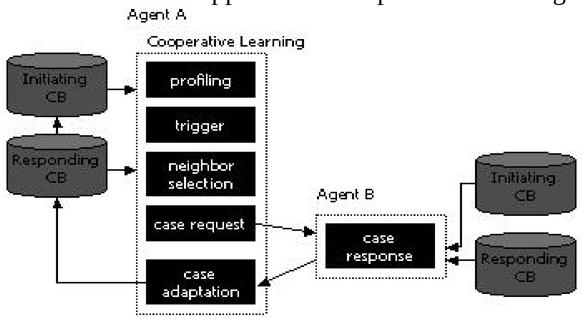

Figure 3. The cooperative learning design

(1) The agent evaluates the case to determine whether it is problematic. To designate a case as problematic, we use heuristics *H4* and *H5*: a (frequently used) case is problematic if it has a low success rate (*TSU/TU*) and a high incurrence rate (*TINC/TU*). The profiling module keeps track of the utility of the cases.

(2) The agent only requests help from a selected agent that it thinks is good at a particular problem. We want to approach neighbors who have initiated successful negotiations with the current agent, with the hope that the agent may be able to learn how those neighbors have been able to be suc-

cessful. This is determined based on the profile of each neighbor that the agent maintains. The exchange protocol is carried out by the case request and response modules.

(3) If the foreign case is similar to the problematic case, the agent adapts the foreign case before adopting it into its casebase. At the same time, the usage history parameters of the new case are reset.

(4) If a problematic case cannot be fixed after *K* times, it will be removed (Heuristics *H6* and *H7*).

Note that this cooperative learning is performed separately from the actual coalition formation process due to real-time constraints — a negotiation task needs immediate attention and cannot afford meddling with cooperative learning.

4 Experiments and Results

We have implemented a multiagent system with multiple agents that perform multi-sensor target tracking and adaptive CPU reallocation in a noisy environment (simulated by a JAVA-based program called RADSIM). Each agent has the same capabilities, but is located at a unique position. Each agent controls a sensor and can activate the sensor to search-and-detect the environment. When an agent detects a moving target, it tries to implement a tracking coalition by cooperating with at least two neighbors. And this is when a CPU shortage may arise: the activity may consume more CPU resource. When an agent detects a CPU shortage, it needs to form a CPU coalition to address the crisis.

The multi-agent system is implemented in C++. In the current design, each agent has $3+N$ threads. The *core* thread is responsible for making decisions, managing tasks, and overseeing negotiations. A *communication* thread is used to interact with the message passing system of the sensor. An *execution* thread actuates the physical sensor: calibration, search-and-detect for a target, etc. Each agent also has N *negotiation* threads to concurrently negotiate with other agents.

We used two simulations for our experiments. We conducted experiments in a simulation called RADSIM where communication may be noisy and unreliable, and one or two targets may appear in the environment. We also designed and implemented our own CPU shortage simulation module. Each task is designated with a CPU usage amount plus a random factor. When an agent detects a CPU shortage, the tasks that it currently performs slow down. Thus, a CPU shortage that goes unresolved will result in failed negotiations since our negotiations are time-constrained.

4.1 Impacts of Learning

We also conducted experiments with four versions of learning: (1) both case-based reasoning and reinforcement learning (CBRRL), (2) only case-based reasoning (NoRL), (3) only reinforcement learning (NoCBR), and (4) no learning at all (NoCBRRL). Figure 4 shows the result in terms of the success rates for negotiations and coalition formations.

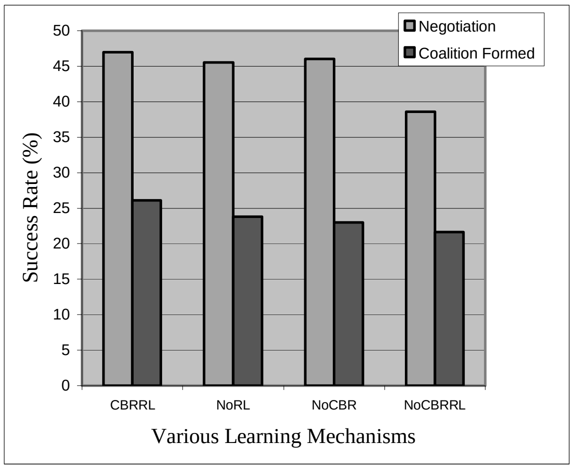

Figure 4. Success rates of negotiations and coalition formations for different learning mechanisms

The agent design with both case-based reasoning/learning and reinforcement learning outperformed others in both negotiation success rate and coalition formation success rate. That means with learning, the agents were able to negotiate more effectively (and perhaps more efficiently as well) that led to more coalitions formed. Without either 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 methods, 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). With no learning, the agents fared noticeably poorly.

4.2 Resource Allocation and System Coherence

We conducted experiments in CPU re-allocation to test the coherence of our system. We refer to the CPU allocation as a sustenance resource since in order for an agent to obtain more CPU, it needs to incur CPU usage while negotiating for the resource. By varying the amount of the initial CPU allocation to each agent, we created mildly-constrained, overly-constrained, and unevenly-constrained scenarios. Tables 3 and 4 compared the agents' behavior in terms of successes in negotiations and coalition formations. In particular, the coalition success rate is the number of successfully formed coalitions over the number of coalitions initiated, where a coalition is successfully formed when the CPU obtained satisfies the agent's need. We observed the following:

(1) In all experiments, the reduction in CPU shortage of each agent and the whole system was obvious. Gradually, the CPU resource was reallocated more evenly among agents. The possibility of a CPU shortage decreases and each agent's shortage amount decreases. This shows a coherent, cooperative behavior among the agents.

(2) In all experiments, after some period of time, each agent’s CPU allocation converged to an average level (14%). After that, each agent fluctuated around that level.

(3) We also observed that the coalition formation was the most successful for the system as a whole when there were roughly the same number of resourceful and resource-starved agents (Experiment #3) and this type of system also required the least number of negotiations and coalitions to converge.

Initiating agent	Initial CPU allocation (%) / Negotiation success rate (%) / # of negotiations											
	Experiment #1			Experiment #2			Experiment #3			Experiment #4		
	Total CPU=28%			Total CPU=56%			Total CPU=56%			Total CPU=56%		
1	7	15	20	7	45.5	11	7	80	5	7	42.9	14
2	7	25	12	7	55.6	9	7	75	8	14	80	5
3	7	35.3	17	7	41.7	12	21	62.5	8	14	44.4	9
4	7	30	10	35	25	4	21	80	5	21	66.7	3
Average	7	25.4	14.75	14	44.4	9	14	73.1	6.5	14	51.6	7.75

Table 3. Comparison between negotiations in experiments

Initiating agent	Initial CPU allocation (%) / Coalition formation rate (%) / Coalition success rate (%) / # of coalitions															
	Experiment #1				Experiment #2				Experiment #3				Experiment #4			
	Total CPU=28%				Total CPU=56%				Total CPU=56%				Total CPU=56%			
1	7	28.6	14.3	7	7	80	20	5	7	100	100	2	7	66.7	16.7	6
2	7	60	40	5	7	75	50	4	7	100	100	3	14	66.7	66.7	3
3	7	57.1	28.6	7	7	80	40	5	21	100	66.7	3	14	75	25	4
4	7	40	20	5	35	33.3	0	3	21	100	100	2	21	50	50	2
Average	7	45.8	25	6	14	70.6	29.4	4.25	14	100	90	2.5	14	66.7	33.3	3.75

Table 4. Comparison between coalitions in experiments

4.3 Individual & Cooperative Case-Based Learning

For our investigation of individual and cooperative case-based learning, we conducted two sets of experiments, Comprehensive Experiment A (CEA) and Comprehensive Experiment B (CEB). We carried out CEA to study the effects of individual learning in subsequent cooperative learning and the roles of cooperative learning in agents of different initial knowledge. We performed CEB to investigate the effects of the environment on the agents’ learning.

4.3.1 Comprehensive Experiment A (CEA)

We conducted four sets of experiments in CEA as shown in Table 5. The goal of these experiment sets was to investigate how learning differed given different casebase sizes, and how learning differed given different types of initial casebases (some had cases collected from different agents from an earlier run, some had only their own cases). Note that for the following experiments we set the limit on the casebase size as 30 where case replacement started to take place after the casebase reached this number. We used two main parameters to evaluate the casebases: utility and diversity. First, we rank the outcome of each case following the utility values of Table 6.

Experiment Set	A1	A2	A3	A4
ES1	16	16	16	16
ES2	2	16	16	16
ES3	16	16	16	28
ES4	2	10	20	28

Table 5. Experiment sets. For example, in ES1, every agent has 16 cases in its casebase; and so on

Outcome	Utility
success	10
channel_jammed	6
Aborted	5
out_of_time	4
out_of_resources	3
rejected	2
others	0

Table 6. Utility of each outcome for a case

The average utility of the case base is the average product of each case’s *TU* value and the utility value of its outcome. The diversity measure of a casebase is computed as the average difference between each pair of cases in the casebase. Three slopes, *sizeSlope*, *diffSlope*, and *utilSlope*, were computed as growth rate between the first learning point and the last learning point, for size, diversity, and utility, respectively. Table 7 shows one example of the results on initiating casebases.

			Size Slope	Utility Slope	Diff. Slope	Ave. Utility Gain	Ave. Diff. Gain
Experiment 1 Combine-first-combine-later	A1	indi	0.1207	0.0832	0.0225	0.0718	0.0938
		coop	0.1143	0.0752	0.0173	0.2242	0.0299
	A2	indi	0.0469	0.1112	0.0108	0.1035	0.0779
		coop	0.0385	0.119	0.0064	0.2152	-0.0005
	A3	indi	0.1642	0.086	0.0322	0.0932	0.0719
		coop	0.1667	0.0579	0.03	0.0084	0.0569
	A4	indi	0.1507	0.0784	0.0224	0.0788	0.072
		coop	0.1556	0.0697	0.0218	0.1025	0.1022
		ave	**0.1197**	**0.0851**	**0.0204**	**0.1122**	**0.0630**
Experiment 2 Individual-first-combine-later	A1	indi	0.1429	0.107	0.0447	0.0957	0.1391
		coop	0.1351	0.1106	0.0402	0.1899	-0.0007
	A2	indi	0.1569	0.062	0.0364	0.0614	0.1314
		coop	0.1795	0.0683	0.038	0.0884	-0.0071
	A3	indi	0.2	0.1317	0.0575	0.1339	0.2887
		coop	One point			-0.0974	0.1831
	A4	indi	0.1136	0.0735	0.0318	0.0598	0.145
		coop	0.1154	0.0648	0.0333	0.1858	0.1005
		ave	**0.1406**	**0.0810**	**0.0374**	**0.1135**	**0.0847**

Table 7. Utility and difference gains for both Sub-Experiments Exp1 and Exp2, after the second stage, for initiating casebases

Looking at all our results, we observed the following:

(1) Cooperative learning results in more utility and diversity per learning occurrence than individual learning,

(2) A small casebase learns more effectively in terms of utility and diversity, but not faster since our learning is

problem-driven. A large casebase learns in a similar manner as an average casebase except when it is greater than the preset limit that triggers case replacement.

(3) The initial casebase affects the effectiveness of learning. Both types of learning bring more utility and diversity to an initial casebase previously grown within an agent than one that has been influenced by other agents.

4.3.1 Comprehensive Experiment B (CEB)

The objective of CEB was to see how the learning results changed in different environments, as shown in Table 8.

	Combination of CPU and Tracking Coalitions
ES1	CPU coalitions more often than tracking
ES2	CPU and tracking coalitions similarly frequent
ES3	Tracking coalitions more often than CPU

Table 8. Sub-Experiments setup in CEB

A tracking coalition is more taxing since it requires at least three agents to be successfully formed. Moreover, a tracking task is durational such that it takes time to actually carry out the tracking task. However, a CPU re-allocation task is carried out at a point in time. In addition, a tracking task is highly time-constrained. A coalition has to be formed in time to *catch* the target before the target moves out of the sensor coverage area. Thus, negotiations related to tracking are more difficult to manage and handle. For these three sets of sub-experiments, they had a few things in common: (1) all of them began with the same set of initial case bases, and (2) every sub-experiment ran with the both individual and cooperative learning. We observed the following:

(1) Different environments affect agents' learning behavior. Depending on the frequency of a task and its characteristics, an agent may rely more on individual learning or cooperative learning. For example, if a type of tasks (tracking) is time consuming and durational, then increasing its frequency actually weakens the potential benefit of individual learning and encourages the agent to perform more cooperative learning.

(2) The environments impact the two initiating and responding roles differently, especially for negotiations associated with tough requirements (such as at least three members of a tracking coalition). Since an initiating agent has to shoulder the coalition management and decision making, it is able to learn more diverse and useful cases. But, negotiating as a responder, an agent's responsibility is less and thus considers fewer issues; thus the learning is less impressive.

5 Conclusions

We have described an integrated multilevel approach to coalition formation, using case-based learning and reinforcement learning to learn better tactics as the agent solves a problem, and distributed, cooperative case-based learning to learn improve the agent's knowledge base strategically. We have conducted several experiments and the results have been promising in proving the feasibility of our approach. With learning, our agents negotiate and form coalitions better. Our future work will focus on tying the outcome of an executed coalition (already formed) to the planning stage to improve our strategic learning.

References

[Jennings, 2002] N. R. Jennings. Coalition formation algorithms for virtual organizations. http://www.iam.ecs.soton.ac.uk/projects/cfvo.

[Marsella *et al.*, 1999] S. Marsella, J. Adibi, Y. Al-Onaizan, G. A. Kaminka, I. Muslea, M. Tallis, and M. Tambe. On being a teammate: experiences acquired in the design of RoboCup teams. In *Proc. 3rd Agents'99*, pages 221-227, Seattle, WA, 1999.

[Martin and Arcos, 1999] F. J. Martin, E. Plaza and J. L. Arcos. Knowledge and experience through communication among competent (peer) agents, *Int. J. Software Engr. & Knowledge Engr.*, 9(3):319-341, 1999.

[Martin and Plaza, 1999] F. J. Martin and E. Plaza Auction-based retrieval, *Proc. 2nd Congres Catala d'Intel.ligencia Artificial*, pages 1-9, 1999.

[Plaza *et al.*, 1997] E. Plaza, J. L. Arcos and F. Martin. Co-operative case-based reasoning, in G Weiss (ed.) *Distributed Artificial Intelligence Meets Machine Learning*, Lecture Notes in Artificial Intelligence, Springer Verlag, pages 1-21, 1997.

[Prasad and Plaza, 1996] M. V. N. Prasad, and E. Plaza. Corporate memories as distributed case libraries, *Proc. 10th KAW'96*, pages 1-19, 1996.

[Sandholm *et al.*, 1999] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2): 209-238, 1999.

[Sen and Dutta, 2000] S. Sen and P. S. Dutta. Searching for optimal coalition structures. In *Proc. 4th Int. Conf. on Multiagent Systems*, pages 286-292, Boston, MA, July 7-12, 2000.

[Shehory *et al.*, 1997] O. Shehory, K. Sycara, and S. Jha. Multi-agent coordination through coalition formation. In *Intelligent Agents IV: Agent Theories, Architectures and Languages*, Lecture Notes in Artificial Intelligence, number 1365, pages 143-154, Springer, 1997.

[Soh and Tsatsoulis, 2001] L.-K. Soh and C. Tsatsoulis. Reflective negotiating agents for real-time multisensor target tracking. In *Proc. of IJCAI'01*, pages 1121-1127, Seattle, WA, 2001.

[Tohme and Sandholm, 1999] F. Tohme and T. Sandholm. Coalition formation processes with belief revision among bounded rational self-interested agents. *Journal of Logic and Computation*, 9(6):793-815, December 1999.