Adaptation Using Cases in Cooperative Groups

Thomas Haynes and Sandip Sen
Department of Mathematical & Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104–3189
e–mail: [haynes,sandip]@euler.mcs.utulsa.edu

Abstract

In order to effectively exploit opportunities presented in the environment agents in a group must be well-adapted to each other and to the environment. Agents that fail to adapt and modify their behavior to suit environmental demands can hinder, rather than aid, in achieving group goals. Adaptability and flexibility are key components of intelligent behavior which allow agent groups to improve performance in a given domain using prior problem solving experience. This paper focuses on a particular incremental learning mechanism by which agents can better adapt to each other using problem solving experience. In particular, we propose a framework in which individual group members learn cases to improve their model of other group members. Using these models agents can choose less greedy and more appropriate actions in the context of the group. We use a testbed problem from the distributed AI literature to show that simultaneous learning by group members can lead to significant improvement in group performance and efficiency over agent groups following static behavioral rules.

1 Introduction

Research in multiagent systems have focused on the problem of coordinating groups of cooperative as well as self-interested agents. Whereas achieving coordination in non-cooperative situations poses a more challenging problem in general, cooperative agent groups must also be responsive to environmental demands and the problem solving state of individual group members to achieve effective group performance. A critical problem in multiagent systems is that the optimal action for an individual agent from its local view of a problem-solving scenario might not be the optimal action for the entire group from the global view of the same problem-solving scenario. Thus an agent can evaluate the utility of its actions at two levels: individual and group. The group level calculations require more information and impose greater cognitive load, whereas the individual level calculations may not always yield desirable results. If agents in a group are likely to interact, utility calculations from even the individual perspective requires reasoning about the possible actions of some or all of

the group members. To reason accurately, each individual in a closely-coupled group should model the behavior of other group members, and use these models to derive expectations of the actions of other group members that can possibly infringe on its own plan of actions. This analysis holds irrespective of whether agents are cooperative, antagonistic, or indifferent to other agents.

If an agent's interactions with other agent are fairly infrequent and the environment is stationary, then a static set of behavioral rules may be sufficient in effectively fulfilling local goals. For a large number of practical and interesting scenarios, however, either agents interact with other agents of unknown composition or all possible agent interactions cannot be foreseen. Adaptation and learning are key mechanisms by which agents can modify their behavior on-line to maintain a viable performance profile in such scenarios.

If agents have accurate and consistent models of other group members, reasoning can expedient and help in achieving group coordination. But often, even in cooperative groups, an individual may not have an up to date model of fellow workers. This may be either because of different agent designers designing different agents that constitute a group and not willing to share the internal details of their agent constitutions, or because of agent behaviors changing because of its learning abilities. In either case, a desirable characteristics of such agent groups is that these agents learn to adapt to each other over time. This is often witnessed in human groups where a collection of individuals go through a phase of adjustments before they can effectively perform as a team. The research question to focus on in this context is: what are the mechanisms available by which individual group members can, over time, better adjust to others and more effectively contribute to the group cause?

One approach for adaptation in a group can be for an agent can start with a very coarse or approximate model of other group members. For example, it can start with the default assumption that every one else is like itself, and modify this model based on experience. Since, in most realistic multiagent system, agents are likely interact in unforeseen ways, a dynamic model of the group must be maintained by an individual. Problems of modeling another agent based on passive observation are many: discrepancy between the expected and actual capabilities, goals, relationships, etc. of the observed agent may lead to an inferred model which is inaccurate and misleading; different agents may perceive different views of the environment and hence the observing agent may not be able to correctly infer the motivations for a given action taken by another agent; actions can take different intervals of time and agents can be acting asynchronously. Even if agents are allowed to communicate, communication delays, improper use of language, different underlying assumptions, etc. can prevent agents from developing a shared common body of knowledge [5].

Given the above assumption about the initial model of other agents, an adaptive agent can possibly use various different learning methods to incrementally improve its model of other group members. It would be preferable to use an incremental, rather than batch learning model, and for most multiagent systems, an anytime learning algorithm that is computationally cheap in both the knowledge acquisition and application phases is desirable. Our goal in this research is to show that given some generic behavioral rules that are effective in achieving local goals in the absence of other agents, but are ineffective when they have to share resources with other group members, agents can adapt their behavior to achieve their goals in the presence of other agents. Some of the assumptions in our work are:

agents are provided with a default set of behavioral rules to follow; repeated interaction with other group members allow agents to modify these behavioral rules in some but not in all cases; agents are motivated to achieve local goals but are cognizant of global goals; agents are autonomous; agent perceptions are accurate; agents do not communicate explicitly; all agents act and adapt concurrently.

The elimination of communication between agents further limits the use of existent mechanisms in multiagent systems literature that are used to achieve group coordination. Though we value the wealth of information that can be shared and utilized by agents to improve group performance, our goal is to push the limits of group performance that can be achieved when agents are intelligently building and using models of others without any explicit help from them. The reason for our choice is to investigate how far we can go without using explicit communication. After having leveraged as much as possible from this mode of group problem solving, we believe we will be better equipped to add and exploit communication skills in agents.

We now introduce our proposed model of adaptation with which agents can use problem-solving experience to both update the model they have of other agents, and to some extent predict what others are going to do in a specific situation. This predictive abilities allow agents to choose actions that are less likely to be mutually conflicting or disruptive. We propose a learning framework in which agents learn cases to override default behavioral rules. When the actual outcome of the action of an agent using its behavioral rules is not consistent with the expected outcome based on the model the agent has of other agents, the agent recognizes that a conflict has occurred and that its behavior is not appropriate in that situation. For those situations, the agent learns exceptions to its behavioral rules that are likely to prevent future conflicts. Agents follow their behavioral rules except when a learned case suggests alternative actions. Through this process, the agents dynamically evolve a behavior that is suited for the group in which it is placed. The multiagent case–based learning (MCBL) algorithm thus utilizes exceptions to a default ruleset, which describes the behavior of an agent. These exceptions form a case library. The agent does not reason with these cases, as in CBR [9], but rather adapts an inaccurate individual model to improve performance. Though researchers have used CBR in multiagent systems [14], little work has been done in learning cases in multiagent systems [3, 12].

2 Case–Based Learning

Case–based reasoning (CBR) [4, 6, 9] is a model of this definition of intelligence and is a reasoning process for information retrieval and modification of solutions to problems. A *case* is typically comprised of a representation of a state of a domain and a corresponding set of actions to take to lead from that state to another desired state. These actions could be either a plan, an algorithm, or a modification of another case's actions. A *case library* is a collection of cases. A CBR algorithm contains a module to determine if there is a case that matches the current state of a domain, and so then it is retrieved and used as is. If there is no such match, then cases that are similar to the current state are retrieved from the case library. The set of actions corresponding to the most relevant case is then adapted to fit the current situation. Cardie [2] defined case–based learning (CBL) as a machine learning

technique used to extend instance–based learning (IBL) [1]. The IBL algorithm retrieves the nearest instance (for our purposes, an instance can be thought of a case) to a state, and performs the suggested actions. There is no case adaptation if the retrieved instance is not a direct match to the current state. With CBL, adaptation can take place.

We view cases as generalizations of sets of instances, and in the context of multiagent systems, we define MCBL as a learning system by which an agent can extend its default rules to allow it to respond to exceptions to those rules. The adaptation lies in translating the general case to specific instances. In our framework, the cases in the MCBL system are used by agents to preferentially order their actions. In a single agent system, the state represents the environment, and in multiagent systems, it represents the environment and the agent's expectations of the actions of other agents. In the following we present our formalization of a CBL system tailored for use in multiagent systems.

What do cases represent? The behavioral rules that an agent has can be thought of as a function which maps the state (s) and the applicable action set (A) of an agent to a preference ordering of those actions:

$$BH(s, A) \Rightarrow A' \ = < a_{x_1} a_{x_2} \ldots a_{x_k} > .$$

The cases an agent learns allows it to modify this preference ordering:

$$CB(s, A') \Rightarrow A'' \ = < a'_{x_1} a'_{x_2} \ldots a'_{x_j} >, j \leq k.$$

A case need not fire every time the agent is to perform an action, i.e., A'' can be the same as A'. Cases can be positive or negative [4, 6]. A positive case informs the agent what to do, i.e. it reorders the set of actions. A negative case can reorder the actions and/or delete actions from the set. The cases used in the system we are presenting in this paper are negative in the sense that they eliminate one or more of the most preferred actions as suggested by behavioral rules.

When do agents learn cases? An agent learns a case when its expectations are not met. If either the behavioral rules or a case predict that given a state s_n and the application of an action a_x, the agent should expect to be in state s'_n, and the agent does not end up in that state, a case is learned by the corresponding agent. This case will then cause the action a_x not to be considered the next time the agent is in state s_n. In multiagent systems, we expect cases will be learned primarily from unexpected interactions with other agents. Cases can be generalized by eliminating irrelevant features from the representation of the state. If another agent is too far away to influence the state of an agent, A_i, then the expectations of its behavior should not be included by A_i as it either indexes or creates a new case.

What happens as models change? If agent A_i learns an exception to agent A_j's default rules and agent A_j does not modify its behavioral rules, then A_i does not have to check to see if that exception has to be modified at some later time. In a system where both agents are modifying their behavioral rules, A_i must check to see if A_j took the action corresponding to the case. If it has not, then A_j's behavioral rules have changed, and A_i must update its model of A_j.

3 Predator-Prey

The predator-prey, or pursuit, domain has been widely used in distributed AI research as a testbed for investigating cooperation and conflict resolution [8, 10, 13]. Four predator try agents to capture a prey agent. In spite of its apparent simplicity, it has been shown that the domain provides for complex interactions between agents and no hand-coded coordination strategy is very effective [8]. Simple greedy strategies for the predators have long been postulated to efficiently capture the prey [10]. The underlying assumption that the prey moves first, then the predators move in order simplifies the domain such that efficient capture is possible. Relaxing the assumption leads to a more natural model in which all agents move at once. This model has been shown to create deadlock situations for simple prey algorithms of moving in a straight line (Linear) or even not moving at all (Still) [8]! Two possible solutions have been identified: allowing communication and adding state information. We investigate a learning system that utilizes past expectations to reduce deadlock situations.

The predator agents have to capture the prey agent by blocking its orthogonal movement. The game is typically played on a 30 by 30 grid world, which is toroidal [13]. The behavioral strategies of the predators use one of two distance metrics: *Manhattan distance* (MD) and *max norm* (MN). The MD metric is the sum of the differences of the x and y coordinates between two agents. The MN metric is the maximum of the differences of the x and y coordinates between two agents. Both algorithms examine the metrics from the set of possible moves, i.e. moving in one of the four orthogonal directions or staying still, and select a move corresponding to the minimal distance metric. All ties are randomly broken.

The MD strategy is more successful than the MN in capturing a Linear prey (22% vs 0%) [8]. Despite the fact that it can often block the forward motion of the prey, its success is still very low. The MD strategy is very susceptible to deadlock situations. How then should the agents manage conflict resolution? An answer can be found in the ways we as humans manage conflict resolution, with cases [9]. In the simplest sense, if **predator 1** senses that if **predator 2** is in its *Northeast* cell, and it has determined to move *North*, then if the other agent moves *West* there will be a conflict with **predator 2. Predator 1** should then learn not to move *North* in the above situation, but rather to its next most preferable direction.

In this research we examine multiagent case–based learning (MCBL) of potential conflicts. The default rule employed by predators is to move closer to the prey, unless an overriding case is present. If a case fires, the next best move is considered. This process continues until a move is found without a corresponding negative case. If all moves fire a negative case, then the best move according to the default behavior should be taken [1]. If the suggested move, either by the default rule or a case firing, does not succeed, then a new case is learned.

Agents need a dynamic learning mechanism to model the actions of other agents. Until the potential for conflicts exist, agents can follow their default behaviors. When a conflict occurs, an agent learns that another agent will act a certain way in a specific situation S_j. Thus agent A_i learns not to employ its default rule in situation S_j; instead it considers its next best action. As these specific situations are encountered by an agent, it is actually forming a case–base library of conflicts to avoid. As an agent learns cases, it begins to model the actions of the group. Each agent starts with a rough model of the group, and improves

[1] No such situation has been observed in any of our experiments.

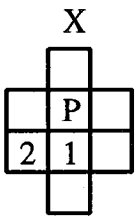

Figure 1: Case window for **predator 1.**

it by incrementally refining the individual models of other agents in the group.

4 The case library

The ideal case representation for the predator–prey domain is to store the entire world and to have each case inform all predators where to move. There are two problems with this setup: the number of cases is too large, and the agents do not act independently. This case window and others are analyzed and rejected in [7]. Unless the entire world is used as a case, any narrowing of the case window is going to suffer from the above points of the "effective" case window presented above. The same case can represent several actual configurations of the domain being modeled. If we accept that the case windows are going to map to more than one physical situation and hence cases are generalized to apply to multiple situations, then clearly the issue is how to find the most relevant general case. If we limit the case window to simply represent the potential conflicts that can occur "after" the agent selects a move based on the default rules or learned case, then we can utilize the case windows shown in Figure 1.

Our cases are negative in the sense they tell the agents what not to do. (A positive case would tell the agent what to do in a certain situation [4].) A crucial piece of information in deciding local action is where does the agent believe the other agents are going to move? This is modeled by storing the orientation of the prey's position with respect to the desired direction of movement of the agent. Specifically, we store whether the prey lies on the agent's line of advance or if it is to the left or right of the line.

An agent has to combine its behavioral rules and learned cases to choose its actions. This is shown algorithmically in Figure 2. When an agent prepares to move, it orders its possible actions by the default rules (the MD distance metric with the additional tie–breaking mechanisms). It then iterates down the ranked list, and checks to see if a negative case advises against that move. To index a case, the agent first determines whether the possible action is for movement or staying still. As discussed above, this decision determines the particular case library to be accessed. Then it examines the contents of each of the four cells in the case whose contents can cause conflicts. The contents can be summed to form an unique integer index in a base number system reflecting the range of contents. The first possible action which does not have a negative case is chosen as the move for that turn.

1 *Preferentially order actions by behavioral rules.*
2 *Choose the first action which does not cause a negative case to fire, i.e., one containing a conflict.*
3 *If the state corresponding to the selected action is not reached, then the agent must learn a case.*

Figure 2: Algorithm for selecting actions based on negative cases.

5 Experimental Setup and Results

The initial configuration consists of the prey in the center of a 30 by 30 grid and the predators placed in random non–overlapping positions. All agents choose their actions simultaneously. The environment is accordingly updated and the agents choose their next action based on the updated environment state. If two agents try to move into the same location simultaneously, they are "bumped back" to their prior positions. One predator, however, can push another predator (but not the prey) if the latter decided not to move. The prey does not move 10% of the time; effectively making the predators travel faster than the prey. The grid is toroidal in nature, and only orthogonal moves are allowed. All agents can sense the positions of all other agents. Furthermore, the predators do not possess any explicit communication skills; two predators cannot communicate to resolve conflicts or negotiate a capture strategy. The case window employed is that depicted in Figure 1. We have also identified two enhancements to break ties caused by the default rules employed in the MD metric: look ahead and least conflict [7]. Look ahead breaks ties in which two moves are equidistant via MD, the one which is potentially closer in two moves is selected. If look ahead also results in a tie, then the move which conflicts with the least number of possible moves by other predators is selected to break the tie.

Initially we were interested in the ability of predator behavioral rules to effectively capture the Still prey. We tested three behavioral strategies: MD – the basic MD algorithm, MD–EDR – the MD modified with the enhancements discussed in [7], and MD–CBL – which is MD–EDR utilizing a case base learned from training on 100 random simulations. The algorithms produced the following captures: MD – 3, MD–EBL – 46, and MD–CBL – 97. While the enhancement of the behavioral rules does increase capture, the addition of learning via negative cases leads to capture in almost every simulation.

We also conducted a set of experiments in which the prey used the Linear algorithm as its behavioral rule. The MD–CBL algorithm was trained on the Still prey. We trained on a Still prey because the Linear prey typically degrades to a Still prey [7]. We also present the results of training the MD–CBL on the Linear prey (MD–CBL*). The algorithms produced the following captures: MD – 2, MD–EDR – 20, MD–CBL – 54, and MD–CBL* – 66.

In Table 1 we present the number of case learned during the training and the number of cases utilized during the testing trials. Note that the learning is evenly distributed across the agents. The increase in learning from Still to Linear can be explained by the interactions caused by the agents chasing the prey: the dynamic movement of the prey forces the predators into configurations not seen in the static case.

With both prey algorithms, the order of increasing effectiveness was MD, MD–EDR, and MD–CBL. Clearly the addition of MCBL to this multiagent system is instrumental in increasing the effectiveness of the behavioral rules. There is some room for improvement, as the results from the Linear prey indicate. A majority of the time spent in capturing the Linear prey is spent chasing it.

	Cases learned		Utilization of cases	
Predator	Still	Linear	Still	Linear
1	27	51	80	118
2	27	40	78	76
3	29	47	80	78
4	33	53	82	92

Table 1: Number of cases learned while trained on the Still and Linear preys. (Of 100 random trials, the Still prey was caught 100 times, and the Linear prey was caught 63). Also, the number of times the learned cases were utilized in the test trials.

6 Conclusions

We have shown that MCBL can be effectively applied to multiagent systems. We have taken a difficult problem of group problem-solving from DAI literature and shown how MCBL can significantly improve on the performance of agent groups utilizing fixed behavioral rules. The individual agents were able to learn cases when its expectations were not met. The individual then utilizes the cases to better adapt to the behavior of the rest of the group.

Some possible future work involves investigating what happens if the different agents involved in a conflict have different views of the the situation. With imperfect agent perception, the agents might all learn individual cases, resulting in inefficient problem solving in the future. Agents need to learn when another agent is not taking the action predicted by the case; the expected action of the agent must be stored, and when it deviates from that action, the relevant case can be forgotten. We have presented a homogeneous agent architecture; there can be situations where agents must adapt in a heterogeneous society of agents. In the context of the pursuit domain, the predators can also learn cases based on the expectations of the prey's movements.

References

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, January 1991.

[2] Claire Cardie. Using decision trees to improve case–based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32. Morgan Kaufmann, 1993.

[3] Andrew Garland and Richard Alterman. Preparation of multi-agent knowledge for reuse. In D. W. Aha and A. Ram, editors, *AAAI Symposium on Adaptation of Knowldege for Reuse*, November 1995.

[4] Andrew R. Golding and Paul S. Rosenbloom. Improving rule–based systems through case–based reasoning. In *AAAI*, pages 22–27, 1991.

[5] Joseph Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[6] Kristian Hammond, Timothy Converse, and Mitchell Marks. Towards a theory of agency. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 354–365, San Diego, November 1990. Morgan Kaufmann.

[7] Thomas Haynes, Kit Lau, and Sandip Sen. Learning cases to compliment rules for conflict resolution in multiagent systems. In Sandip Sen, editor, *AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, March 1996.

[8] Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multiagent Systems*, LNAI. Springer Verlag, Spring 1996.

[9] Janet L. Kolodner, editor. *Proceedings of a Workshop on Case–Based Reasoning (DARPA)*. Morgan Kaufmann, 1988.

[10] Richard E. Korf. A simple solution to pursuit games. In *Working Papers of the 11th International Workshop on Distributed Artificial Intelligence*, pages 183–194, February 1992.

[11] Victor R. Lesser. Multiagent systems: An emerging subdiscipline of AI. *ACM Computing Surveys*, 27(3):340–342, September 1995.

[12] M. V. Nagendra Prasad, Victor R. Lesser, and Susan Lander. Reasoning and retrieval in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 1995.

[13] Larry M. Stephens and Matthias B. Merx. The effect of agent control strategy on the performance of a DAI pursuit problem. In *Distributed AI Workshop*, October 1990.

[14] Katia Sycara. Planning for negotiation: A case-based approach. In *DARPA Knowledge-Based Planning Workshop*, pages 11.1–11.10, December 1987.