Learning Cases to Resolve Conflicts and Improve Group Behavior

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

Groups of agents following fixed behavioral rules can be limited in performance and efficiency. 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. We motivate the usefulness of individual learning by group members in the context of overall group behavior. In particular, we propose a framework in which individual group members learn cases to improve their model of other group members. 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.

Introduction

An agent is rational if when faced with a choice from a set of actions, it chooses the one that maximizes the expected utilities of those actions. Implicit in this definition is the assumption that the preference of the agent for different actions is based on the utilities resulting from those actions. A problem in multiagent systems is that the best action for Agent A_i might be in conflict with that for another Agent A_j. Agent A_i, then, should try to model the behavior of A_j, and incorporate that into its expected utility calculations (Gmytrasiewicz & Durfee 1995).

The optimal action for an individual agent might not be the optimal action for its group. Thus an agent can evaluate the utility of its actions on 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. Thus, to be effective, each individual in a closely-coupled group should model the behavior of other group members, and use these models while reasoning about its actions. The above analysis holds irrespective of whether agents are cooperative, antagonistic, or indifferent to other agents.

In general, 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. In such a situation, since agents can interact in unforeseen ways, a dynamic model of the group must be maintained by the 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 (Halpern & Moses 1990). For example, even communicating intentions and negotiating to avoid conflict situations may prove to be too time consuming and impractical in some domains (Lesser 1995). These and other problems combine to confound an individual in its attempt to predict the behavior of other members of its group.

We investigate a method for allowing agents to improve their models of other members of the group. Using these evolving models an agent can determine appropriate local actions. Our goal 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 learn to modify 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.

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. A similar argument can also be made for cooperative agent groups for environments that are well understood and for which effective group behaviors can be designed off-line. 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. A number of researchers have recently started investigating learning approaches targeted for multiagent systems (Sen 1995).

Since we eliminate communication between agents, then how is group learning to occur? When the actual outcome of the action of an agent is not consistent with the expected outcome based on the model the agent has of other agents, the agent knows that it has found a case where its model of other agents and its default behavioral rule is inadequate. We believe that case based reasoning (CBR) can be adapted to provide effective learning with such situations. Though researchers have used CBR in multiagent systems (Sycara 1987), little work has been done in learning cases in multiagent systems (Garland & Alterman 1995; Prasad, Lesser, & Lander 1995). We propose a learning framework in which agents learn cases to complement behavioral rules. Agents find out through interacting with other agents that their behavior is not appropriate in certain situations. In those situations, they learn exceptions to their behavioral rules. They still follow their behavioral rules except when a learned case guides them to act otherwise. Through this process, the agents dynamically evolve a behavior that is suited for the group in which it is placed.

A typical multiagent situation in which case learning can be effectively used to adapt local behavior can be seen in the interactions of Adam and his cat Buster: Buster is diabetic, and must receive insulin shots every morning; he must also be given some food with his shot. Adam decides to administer the shot when he wakes up to go to work. He discovered that Buster would react to the sound of the alarm, and go to his food dish. As the alarm clock does not go off on weekends, the cat learned it has to wake up Adam to get its food. The latter is an exception to the routine behavior, and is learned when the cat's expectation of the alarm clock going off in the morning is not met.

We propose to place a learning mechanism on top of the default ruleset, which adapts the individual greedy strategy such that local goal maps to the global goal. The multiagent case-based learning (MCBL) algorithm 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 (Kolodner 1993), but rather modifies an inaccurate individual model to approximate a group model.

Case–Based Learning

Case–based reasoning (CBR) (Golding & Rosenbloom 1991; Hammond, Converse, & Marks 1990; Kolodner 1993) 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 (Cardie 1993) defined case–based learning (CBL) as a machine learning technique used to extend instance–based learning (IBL) (Aha, Kibler, & Albert 1991). 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 (Golding & Rosenbloom 1991; Hammond, Converse, & Marks 1990). 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.

Predator-Prey

We use a concrete problem from DAI literature to illustrate our approach to CBL in multiagent systems. The predator-prey, or pursuit, domain has been widely used in distributed AI research as a testbed for investigating cooperation and conflict resolution (Haynes & Sen 1996; Haynes *et al.* 1995; Korf 1992; Stephens & Merx 1990). The goal is for four predator 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 (Haynes *et al.* 1995). Simple greedy strategies for the predators have long been postulated to efficiently capture the prey (Korf 1992). 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) (Haynes *et al.* 1995)! 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 (Stephens & Merx 1990). 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.

Korf (Korf 1992) claims in his research that a discretization of the continuous world that allows only horizontal and vertical movements is a poor approximation. He calls this the orthogonal game. Korf developed several greedy solutions to problems where eight predators are allowed to move orthogonally as well as diagonally. He calls this the diagonal game. In Korf's solutions, each agent chooses a step that brings it nearest to the predator. The max norm distance metric is used by agents to chose their steps. The predator was captured in each of a thousand random configurations in these games. Korf does not however report the average number of steps until capture. But the max norm metric does not produce stable captures in the orthogonal game; the predators circle the prey, allowing it to escape. Korf replaces the previously used randomly moving prey with a prey that chooses a move that places it at the maximum distance from the nearest predator. Any ties are broken randomly. He claims this addition to the prey movements makes the problem considerably more difficult.

The MD strategy is more successful than the MN in capturing a Linear prey (22% vs 0%) (Haynes *et al.* 1995). Despite the fact that it can often block the forward motion of the prey, its success is still very low. The MD metric algorithms are very susceptible to deadlock situations, such as in Figure 1. The greedy nature of this family of algorithms ensures that in situations similar to Figure 1(c), neither will **predator 2** yield to **predator 3** nor will **predator 3** go around **predator 2**. While the MN metric algorithms can perform either of these two actions, they will not be able to keep the Linear prey from advancing. This analysis explains the surprising lack of captures of the Still prey, and the Linear once it is blocked.

The question that arises from these findings is how should the agents manage conflict resolution? An answer can be found in the ways we as humans manage conflict resolution, with cases (Kolodner 1993). 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.

From one move to the next, the MD algorithm usually suffices in at least keeping a predator agent equidistant from the prey. Since the prey effectively moves 10% slower than the predators and the grid world is toroidal, the prey must occasionally move towards some predators in order to move away from others. Therefore the predators will eventually catch up with it. It is when the predators either get close to the prey, or bunched up on one of the orthogonal axes, that contention for desirable cells starts to come into play. Under certain conditions, i.e., when two or more predator agents vie for a cell, the greedy nature of the above algorithms must be overridden. We could simply order the movements of the predators, allowing **predator 1** to always go first. But it might not always be the fastest way to capture the prey. No static ordering will be effective in all situations. Also this will require communication for synchronization, etc.

What is needed is a dynamic learning mechanism to model the actions of other agents. Until the potential for conflicts exist, agents can follow their default behaviors. It is only when a conflict occurs that 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 it by incrementally refining the individual models of other agents in the group.

Case Representation and Indexing

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 (Haynes, Lau, & Sen 1996). 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 2.

Our cases are negative in the sense they tell the agents what not to do. (A positive case would tell

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

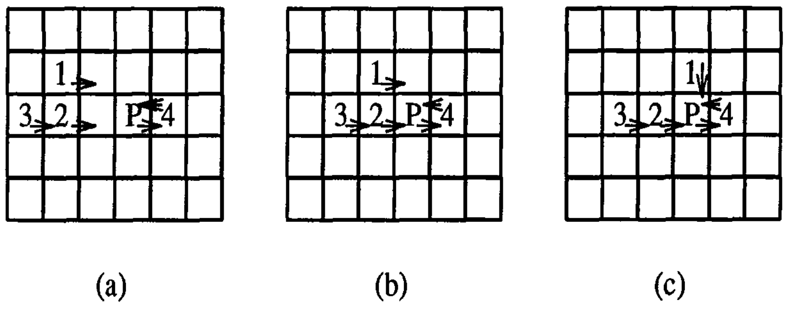

Figure 1: A possible scenario in which a MD metric based predator tries to block the **prey P**. (a) **predator 4** manages to block **P**. **predators 1**, **2**, and **3** move in for the capture. (b) **predator 2** has moved into a capture position. (c) **predator 1** has moved into a capture position. **predator 2** will not yield to **predator 3**. They are in deadlock, and the **prey P** will never be captured.

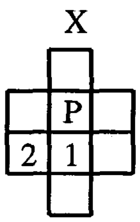

Figure 2: Case window for **predator 1**.

the agent what to do in a certain situation (Golding & Rosenbloom 1991).) 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. In the case window of Figure 2, the prey's relation to the line of advance is marked with a 'X'.

An agent has to combine its behavioral rules and learned cases to choose its actions. 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.

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, then one is "bumped back" to its prior position and learns a case. One predator 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 2. We have also identified two enhancements to break ties caused by the default rules employed in the MD metric: look ahead and least conflict (Haynes, Lau, & Sen 1996). 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 (Haynes, Lau, & Sen 1996), and MD–CBL – which is MD–EDR utilizing a case base learned from training on 100 random simulations. The results of applying these strategies on 100 test cases are shown in Table 1. As discussed earlier, the

MD performs poorly against the Linear prey due to deadlock situations. While the enhancement of the behavioral rules does increase capture, the addition of learning via negative cases leads to capture in almost every simulation.

Algorithm	Captures	Ave. Number of Steps
MD	3	19.00
MD–EDR	46	21.02
MD–CBL	97	23.23

Table 1: Number of captures (out of a 100 test cases) and average number of steps to capture for the Still prey.

We also conducted a set of experiments in which the prey used the Linear algorithm as its behavioral rule. Again we tested the three predator behavioral strategies of MD, MD–EDR, and MD–CBL. The MD–CBL algorithm was trained on the Still prey. We trained on a Still prey because, as shown earlier, the Linear prey typically degrades to a Still prey. We have also presented the results of training the MD–CBL on the Linear prey (MD–CBL*). The results for the Linear prey are presented in Table 2.

Algorithm	Captures	Ave. Number of Steps
MD	2	26.00
MD–EDR	20	24.10
MD–CBL	54	27.89
MD–CBL*	66	26.45

Table 2: Number of captures (out of a 100 test cases) and average number of steps to capture for the Linear prey. MD–CBL* denotes a test of the MD–CBL when trained on a Linear prey.

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. Only after it is blocked do interesting conflict situations occur.

Conclusions

We have shown that case–based learning 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. Our results, however, suggests interesting avenues for future research. Some of the critical aspects of MCBL in agent groups that we plan to further investigate are the following:

Changing agent models : A potential problem with this algorithm is that as Agent A_i is learning to model the group behavior, the other agents in the group are likewise refining their models of the group interactions. This learning is dynamical, and the model Agent A_i constructs of A_j may be invalidated by the model of A_j of A_i. In the environment state E_l, agent A_i learns that A_j will select action a_y. It might be the situation that when the environment is again at E_l, A_j does not select a_y, but instead a_z. Is this an exception to the exception? Or is it just a re-learning of Agent A_i's model of A_j? Note that if z is the expected default behavior without case learning, then A_i might simply need to forget what it had learned earlier.

If we return to our cat example presented earlier, we can see a situation in which group learning occurs when Daylight Savings Time takes effect. The time the alarm clock is set for is pushed back an hour. No one has informed Buster of this change in his environment. Adam's model of the cat is that Buster will try to wake him up "early" on weekday mornings. As predicted, Buster tries to wake up Adam. Adam refuses to get out of bed until the alarm sounds. After a week of not being able to wake Adam, Buster changes his routine by waiting until the new time before he tries to wake Adam.

Diversity of experience : In order for agents to significantly improve performance through learning it is essential that they be exposed to a wide array of situations. In some domains, agents can deliberately experiment to create novel interaction scenarios which will allow them to learn more about other agents in the group.

Forgetting : We believe that in order to further improve the performance of the presented system, it is essential to incorporate a structured mechanism for deleting or overwriting cases that are recognized to be ineffective. This is particularly important in multiagent systems because as multiple agents concurrently adapt their behavior, a particular agent's model of other agents is bound to get outdated. In effect, "the person I knew is not the same person any more!" To modify learned cases, we need to store more information about which agent caused us to learn the case, and what is our expectation of the behavior of that particular agent. We are currently working on developing a representation for the above without exploding the search space.

References

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine Learning* 6(1):37–66.

Cardie, C. 1993. Using decision trees to improve case–based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, 25–32. Morgan Kaufmann Publishers, Inc.

Garland, A., and Alterman, R. 1995. Preparation of multi-agent knowledge for reuse. In Aha, D. W., and Ram, A., eds., *Working Notes for the AAAI Symposium on Adaptation of Knowldege for Reuse.* Cambridge, MA: AAAI.

Gmytrasiewicz, P. J., and Durfee, E. H. 1995. A rigorous, operational formalization of recursive modeling. In Lesser, V., ed., *Proceedings of the First International Conference on Multi–Agent Systems*, 125–132. San Francisco, CA: MIT Press.

Golding, A. R., and Rosenbloom, P. S. 1991. Improving rule–based systems through case–based reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 22–27.

Halpern, J., and Moses, Y. 1990. Knowledge and common knowledge in a distributed environment. *Journal of the ACM* 37(3):549–587. A preliminary version appeared in *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

Hammond, K.; Converse, T.; and Marks, M. 1990. Towards a theory of agency. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, 354–365. San Diego: Morgan Kaufmann.

Haynes, T., and Sen, S. 1996. Evolving behavioral strategies in predators and prey. In Weiß, G., and Sen, S., eds., *Adaptation and Learning in Multiagent Systems*, Lecture Notes in Artificial Intelligence. Berlin: Springer Verlag.

Haynes, T.; Sen, S.; Schoenefeld, D.; and Wainwright, R. 1995. Evolving multiagent coordination strategies with genetic programming. *Artificial Intelligence.* (submitted for review).

Haynes, T.; Lau, K.; and Sen, S. 1996. Learning cases to compliment rules for conflict resolution in multiagent systems. In Sen, S., ed., *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, 51–56.

Kolodner, J. L. 1993. *Case–Based Reasoning.* Morgan Kaufmann Publishers.

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

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

Prasad, M. V. N.; Lesser, V. R.; and Lander, S. 1995. Reasoning and retrieval in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries.* Also as UMASS CS Technical Report 95-27, 1995.

Sen, S., ed. 1995. *Working Notes of the IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems.*

Stephens, L. M., and Merx, M. B. 1990. The effect of agent control strategy on the performance of a DAI pursuit problem. In *Proceedings of the 1990 Distributed AI Workshop.*

Sycara, K. 1987. Planning for negotiation: A casebased approach. In *DARPA Knowledge-Based Planning Workshop*, 11.1–11.10.