6 Learning in Multiagent Systems

Sandip Sen and Gerhard Weiss

6.1 Introduction

Learning and intelligence are intimately related to each other. It is usually agreed that a system capable of learning deserves to be called intelligent; and conversely, a system being considered as intelligent is, among other things, usually expected to be able to learn. Learning always has to do with the self-improvement of future behavior based on past experience. More precisely, according to the standard artificial intelligence (AI) point of view learning can be informally defined as follows:

> *The acquisition of new knowledge and motor and cognitive skills and the incorporation of the acquired knowledge and skills in future system activities, provided that this acquisition and incorporation is conducted by the system itself and leads to an improvement in its performance.*

This definition also serves as a basis for this chapter. Machine learning (ML), as one of the core fields of AI, is concerned with the computational aspects of learning in natural as well as technical systems. It is beyond the scope and intention of this chapter to offer an introduction to the broad and well developed field of ML. Instead, it introduces the reader into learning in multiagent systems and, with that, into a subfield of both ML and distributed AI (DAI). The chapter is written such that it can be understood without requiring familiarity with ML.

The intersection of DAI and ML constitutes a young but important area of research and application. The DAI and the ML communities largely ignored this area for a long time (there are exceptions on both sides, but they just prove the rule). On the one hand, work in DAI was mainly concerned with multiagent systems whose structural organization and functional behavior typically were determined in detail and therefore were more or less fixed. On the other hand, work in ML primarily dealt with learning as a centralized and isolated process that occurs in intelligent stand-alone systems. In the past this mutual ignorance of DAI and ML has disappeared, and today the area of learning in multiagent systems receives broad and steadily increasing attention. This is also reflected by the growing number of publications in this area; see [23, 24, 43, 45, 64, 66, 68] for collections of papers related to learning in multiagent systems. There are two major reasons for this attention, both showing the importance of bringing DAI and ML together:

- there is a strong need to equip multiagent systems with learning abilities; and
- an extended view of ML that captures not only single-agent learning but also multiagent learning can lead to an improved understanding of the general principles underlying learning in both computational and natural systems.

The first reason is grounded in the insight that multiagent systems typically are intended to act in complex—large, open, dynamic, and unpredictable—environments. For such environments it is extremely difficult and sometimes even impossible to correctly and completely specify these systems a priori, that is, at the time of their design and prior to their use. This would require, for instance, that it is known a priori which environmental conditions will emerge in the future, which agents will be available at the time of emergence, and how the available agents will have to react and interact in response to these conditions. The only feasible way to cope with this difficulty is to endow the individual agents with the ability to improve their own and the overall system performance. The second reason reflects the insight that learning in multiagent systems is not just a magnification of learning in stand-alone systems, and not just the sum of isolated learning activities of several agents. Learning in multiagent systems comprises learning in stand-alone systems because an agent may learn in a solitary way and completely independent of other agents. Moreover, learning in multiagent systems extends learning in stand-alone systems. This is because the learning activities of an individual agent may be considerably influenced (e.g., delayed, accelerated, redirected, or made possible at all) by other agents and because several agents may learn in a distributed and interactive way as a single coherent whole. Such an extended view of learning is qualitatively different from the view traditionally taken in ML, and has the capacity to provoke valuable research impulses that lead to novel machine learning techniques and algorithms.

The chapter is organized as follows. First, Section 6.2 presents a general characterization of learning in multiagent systems. Next, Sections 6.3 to 6.5 describe several concrete learning approaches in detail. These sections offer three major, overlapping perspectives of learning in multiagent systems, each reflecting a different focus of attention: learning and activity coordination; learning about and from other agents; and learning and communication. Section 6.6 shows open directions for future research, and gives some further references to related work in ML, economics, and psychology.

6.2 A General Characterization

Learning in multiagent systems is a many-faceted phenomenon, and it is therefore not surprising that many terms can be found in the literature that all refer to this kind of learning while stressing different facets. Examples of such terms are: mutual learning, cooperative learning, collaborative learning, co-learning, team learning, social learning, shared learning, pluralistic learning, and organizational learning. The purpose of this section is to make the different facets more explicit by offering a general characterization of learning in multiagent systems. This is done by describing, from the point of view of multiagent systems, principal categories of learning, basic features in which learning approaches may differ, and the fundamental learning problem known as the credit-assignment problem. The intention of this section is to enable the reader to basically characterize algorithms for learning in multiagent systems, and to get an understanding of what makes this kind of learning different from learning in stand-alone systems. (Further considerations of how to characterize learning in multiagent systems can be found in [63].)

6.2.1 Principal Categories

It is useful to distinguish two principal categories of learning in multiagent systems:

- *centralized learning* (or isolated learning) and
- *decentralized learning* (or interactive learning).

In order to make clear what kinds of learning are covered by these two categories we introduce the notion of a *learning process*:

> *The term learning process refers to all activities (e.g., planning, inference or decision steps) that are executed with the intention to achieve a particular learning goal.*

Learning is said to be centralized if the learning process is executed in all its parts by a single agent and does not require any interaction with other agents. With that, centralized learning takes place through an agent completely independent of other agents—in conducting centralized learning the learner acts as if it were alone. Learning is said to be decentralized if several agents are engaged in the same learning process. This means that in decentralized learning the activities constituting the learning process are executed by different agents. In contrast to centralized learning, decentralized learning relies on, or even requires, the presence of several agents capable of carrying out particular activities.

In a multiagent system several centralized learners that try to obtain different or even the same learning goals may be active at the same time. Similarly, there may be several groups of agents that are involved in different decentralized learning processes. Moreover, the learning goals pursued by such groups may be different or identical. It is also important to see that a single agent may be involved in several centralized and/or distributed learning processes at the same time. Centralized and decentralized learning are best interpreted as two appearances of learning in multiagent systems that span a broad range of possible forms of learning. Learning features that can be applied to structure this broad range are shown in the next subsection.

6.2.2 Differencing Features

The two learning categories described above are of a rather general nature, and they cover a broad variety of forms of learning that can occur in multiagent systems. In the following, several differencing features are described that are useful for structuring this variety. The last two features, which are well known in the field of ML (see, e.g., [6] where several other features are described), are equally well suited for characterizing centralized and decentralized learning approaches. The others are particularly or even exclusively useful for characterizing decentralized learning.

(1) The degree of decentralization. The decentralization of a learning process concerns its

- distributedness and
- parallelism.

One extreme is that a single agent carries out all learning activities sequentially. The other extreme is that the learning activities are distributed over and parallelized through all agents in a multiagent system.

(2) Interaction-specific features. There is a number of features that can be applied to classifying the interactions required for realizing a decentralized learning process. Here are some examples:

- the level of interaction (ranging from pure observation over simple signal passing and sophisticated information exchange to complex dialogues and negotiations);
- the persistence of interaction (ranging from short-term to long-term);
- the frequency of interaction (ranging from low to high);
- the pattern of interaction (ranging from completely unstructured to strictly hierarchical); and
- the variability of interaction (ranging from fixed to changeable).

There may be situations in which learning requires only “minimal interaction” (e.g., the observation of another agent for a short time interval), whereas other learning situations require “maximal interaction” (e.g., iterated negotiation over a long time period).

(3) Involvement-specific features. Examples of features that can be used for characterizing the involvement of an agent into a learning process are

- the relevance of involvement and
- role played during involvement.

With respect to relevance, two extremes can be distinguished: the involvement of an agent is not a condition for goal attainment because its learning activities could be executed by another available agent as well; and to the contrary, the learning goal could not be achieved without the involvement of exactly this agent. With respect to the role an agent plays in learning, an agent may act as a "generalist" in so far as it performs all learning activities (in the case of centralized learning), or it may act as a "specialist" in so far as it is specialized in a particular activity (in the case of decentralized learning).

(4) Goal-specific features. Two examples of features that characterize learning in multiagent systems with respect to the learning goals are

- the type of improvement that is tried to be achieved by learning and
- the compatibility of the learning goals pursued by the agents.

The first feature leads to the important distinction between learning that aims at an improvement with respect to a single agent (e.g., its motor skills or inference abilities) and learning that aims at an improvement with respect to several agents acting as a group (e.g., their communication and negotiation abilities or their degree of coordination and coherence). The second feature leads to the important distinction between conflicting and complementary learning goals.

(5) The learning method. The following learning methods or strategies used by an agent are usually distinguished:

- rote learning (i.e., direct implantation of knowledge and skills without requiring further inference or transformation from the learner);
- learning from instruction and by advice taking (i.e., operationalization—transformation into an internal representation and integration with prior knowledge and skills—of new information like an instruction or advice that is not directly executable by the learner);
- learning from examples and by practice (i.e., extraction and refinement of knowledge and skills like a general concept or a standardized pattern of motion from positive and negative examples or from practical experience);
- learning by analogy (i.e., solution-preserving transformation of knowledge and skills from a solved to a similar but unsolved problem);
- learning by discovery (i.e., gathering new knowledge and skills by making observations, conducting experiments, and generating and testing hypotheses or theories on the basis of the observational and experimental results).

A major difference between these methods lies in the amount of learning efforts required by them (increasing from top to bottom).

(6) The learning feedback. The learning feedback indicates the performance level achieved so far. This feature leads to the following distinction:

- supervised learning (i.e., the feedback specifies the desired activity of the learner and the objective of learning is to match this desired action as closely as possible);
- reinforcement learning (i.e., the feedback only specifies the utility of the actual activity of the learner and the objective is to maximize this utility);

- unsupervised learning (i.e., no explicit feedback is provided and the objective is to find out useful and desired activities on the basis of trial-and-error and self-organization processes).

In all three cases the learning feedback is assumed to be provided by the system environment or the agents themselves. This means that the environment or an agent providing feedback acts as a "teacher" in the case of supervised learning, as a "critic" in the case of reinforcement learning, and just as a passive "observer" in the case of unsupervised learning.

These features characterize learning in multiagent systems from different points of view and at different levels. In particular, they have a significant impact on the requirements on the abilities of the agents involved in learning. Numerous combinations of different values for these features are possible. It is recommended that the reader thinks about concrete learning scenarios (e.g., ones known from everyday life), their characterizing features, and how easy or difficult it would be to implement them.

6.2.3 The Credit-Assignment Problem

The basic problem any learning system is confronted with is the credit-assignment problem (CAP), that is, the problem of properly assigning feedback—credit or blame—for an overall performance change (increase or decrease) to each of the system activities that contributed to that change. This problem has been traditionally considered in the context of stand-alone systems, but it also exists in the context of multiagent systems. Taking the standard AI view according to which the activities of an intelligent system are given by the external actions carried out by it and its internal inferences and decisions implying these actions, the credit-assignment problem for multiagent systems can be usefully decomposed into two subproblems:

- the *inter-agent CAP*, that is, the assignment of credit or blame for an overall performance change to the external actions of the agents; and
- the *intra-agent CAP*, that is, the assignment of credit or blame for a particular external action of an agent to its underlying internal inferences and decisions.

Figures 6.1 and 6.2 illustrate these subproblems. The inter-agent CAP is particularly difficult for multiagent systems, because here an overall performance change may be caused by external actions of different spatial and/or logically distributed agents. Solving this subproblem necessitates to operate on the level of the overall system, and to answer the question of *what action carried out by what agent contributed to what extent to the performance change*. The second subproblem is equally difficult in single-agent and multiagent systems. Solving this sub-problem necessitates to operate on the level of the individual agent, and to answer the question of *what knowledge, what inferences and what decisions led to an action*. How difficult it is to answer these questions and, with that, to solve the CAP, depends on the concrete learning situation.

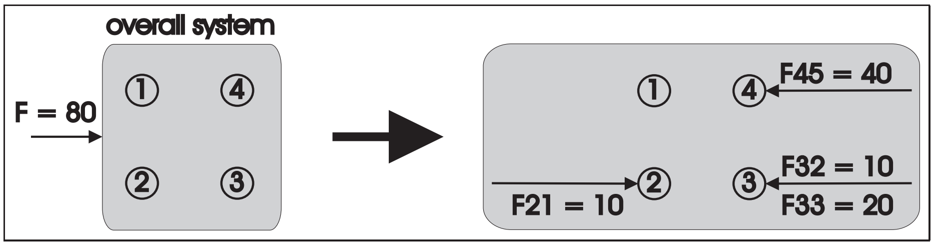

Figure 6.1 Inter-agent CAP. The overall system consists of four agents. The ith agent is represented by (i). A feedback F for an overall performance change is "decomposed" into action-specific portions F_{ij}, where F_{ij} indicates to what degree the jth external action carried out by the ith agent contributes to F.

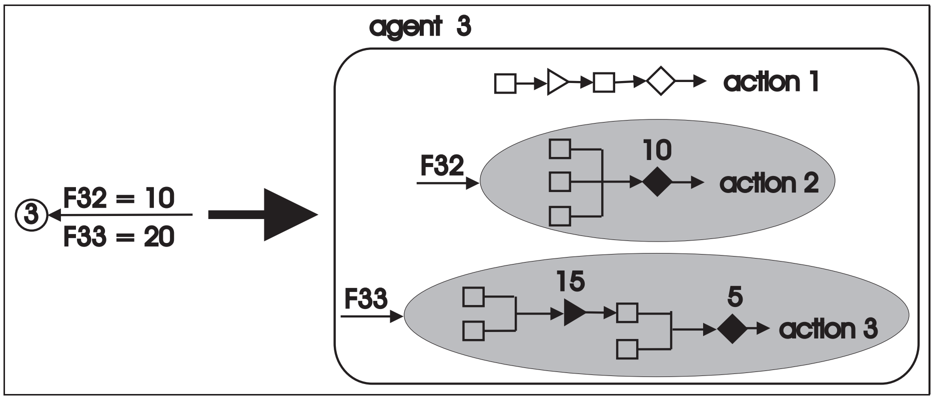

Figure 6.2 Intra-agent CAP. Agent 3 carried out three actions, each based on internal knowledge (□), inferences (▷) and decisions (◇). The feedback F_{33} for action 3, for instance, is divided among an inference and a decision step. Action 1 is assumed to have no influence on the overall performance change.

The above description of the CAP is of a conceptual nature, and aims at a clear distinction between the inter-agent and intra-agent subproblems. In practice this distinction is not always obvious. Moreover, typically the available approaches to learning in multiagent systems do not explicitly differ between the two subproblems, or just focus on one of them while strongly simplifying the other. In any case, it is useful to be aware of both subproblems when attacking a multiagent learning problem.

6.3 Learning and Activity Coordination

This section is centered around the question of how multiple agents can learn to appropriately coordinate their activities (e.g., in order to optimally share resources or to maximize one own's profit). Appropriate activity coordination is much concerned with the development and adaptation of data-flow and control patterns that improve the interactions among multiple agents (see also Chapters 2, 3, and 7). Whereas previous research on developing agent coordination mechanisms focused on off-line design of agent organizations, behavioral rules, negotiation protocols, etc., it was recognized that agents operating in open, dynamic environments must be able to adapt to changing demands and opportunities [29, 44, 68]. In particular, individual agents are forced to engage with other agents that have varying goals, abilities, composition, and lifespan. To effectively utilize opportunities presented and avoid pitfalls, agents need to learn about other agents and adapt local behavior based on group composition and dynamics. To represent the basic problems and approaches used for developing coordination through learning, two of the earliest research efforts in the area of multiagent learning will be described below. The first is work by Sen and his students [47] on the use of reinforcement learning techniques for the purpose of achieving coordination in multiagent situations in which the individual agents are not aware of each another. The second approach is work by Weiss on optimization of environmental reinforcement by a group of cooperating learners [62]. (Both approaches were developed in the first half of the 1990s, and thus at a time of intensified interest in reinforcement learning techniques. It is stressed that several other reinforcement learning methods were described in the literature that could be also used to demonstrate the scope and benefits of learning to coordinate in multiagent settings; we choose the two approaches mentioned above because we are particular familiar with them.) To enable the reader to follow the discussion of the use of reinforcement learning techniques, a brief overview of the reinforcement learning problem and a couple of widely used techniques for this problem class is presented.

6.3.1 Reinforcement Learning

In reinforcement learning problems [3, 26] reactive and adaptive agents are given a description of the current state and have to choose the next action from a set of possible actions so as to maximize a scalar *reinforcement* or *feedback* received after each action. The learner's environment can be modeled by a discrete time, finite state, Markov decision process that can be represented by a 4-tuple $\langle S, A, P, r \rangle$ where S is a set of states, A is a set of actions, $P : S \times S \times A \mapsto [0, 1]$ gives the probability of moving from state s_1 to s_2 on performing action a, and $r : S \times A \mapsto \Re$ is a scalar reward function. Each agent maintains a policy, π, that maps the current state into the desirable action(s) to be performed in that state. The expected value of a discounted sum of future rewards of a policy π at a state x is given by $V_\gamma^\pi \stackrel{\text{def}}{=} E\{\sum_{t=0}^{\infty} \gamma^t r_{s,t}^\pi\}$, where $r_{s,t}^\pi$ is the random variable corresponding to the reward received by the learning agent t time steps after if starts using the policy π in state s, and γ is a discount rate ($0 \leq \gamma < 1$).

Q-Learning

Various reinforcement learning strategies have been proposed that can be used by agents to develop a policy for maximizing rewards accumulated over time. For evaluating the classifier system paradigm for multiagent reinforcement learning described below, it is compared with the Q-learning [59] algorithm, which is designed to find a policy π^* that maximizes $V_\gamma^\pi(s)$ for all states $s \in S$. The decision policy is represented by a function, $Q : S \times A \mapsto \Re$, which estimates long-term discounted rewards for each state-action pair. The Q values are defined as $Q_\gamma^\pi(s,a) = V_\gamma^{a;\pi}(s)$, where $a;\pi$ denotes the event sequence of choosing action a at the current state, followed by choosing actions based on policy π. The action, a, to perform in a state s is chosen such that it is expected to maximize the reward,

$$V_\gamma^{\pi^*}(s) = \max_{a \in A} Q_\gamma^{\pi^*}(s,a) \text{ for all } s \in S.$$

If an action a in state s produces a *reinforcement* of R and a transition to state s', then the corresponding Q value is modified as follows:

$$Q(s,a) \leftarrow (1-\beta)\, Q(s,a) + \beta\, (R + \gamma \max_{a' \in A} Q(s',a')) \quad ,$$

where β is a small constant called *learning rate*.

Learning Classifier Systems

Classifier systems are rule based systems that learn by adjusting rule strengths from environmental feedback and by discovering better rules using genetic algorithms. In the following a simplified classifier system is used where all possible message action pairs are explicitly stored and classifiers have one condition and one action. These assumptions are similar to those made by Dorigo and Bersini [15]. Following their notation, a classifier i is described by (c_i, a_i), where c_i and a_i are respectively the condition and action parts of the classifier. $S_t(c_i, a_i)$ gives the strength of classifier i at time step t.

All classifiers are initialized to some default strength. At each time step of problem solving, an input message is received from the environment and matched with the classifier rules to form a matchset, $\mathcal{M}$. One of these classifiers is chosen to fire and, based on its action, a feedback may be received from the environment. Then the strengths of the classifier rules are adjusted. This cycle is repeated for a given number of time steps. A series of cycles constitute a *trial* of the classifier system. In the bucket brigade algorithm (BBA) for credit allocation, when a classifier is chosen to fire, its strength is increased by the environmental feedback. But before that, a fraction α of its strength is removed and added to the strength of the classifier that fired in the last time cycle. So, if *(i)* the firing of classifier i at time step t results in an external feedback R and *(ii)* classifier j fires at the next time step, the following equation gives the strength update of classifier i:

$$S_{t+1}(c_i, a_i) = (1 - \alpha) * S_t(c_i, a_i) + \alpha * (R + S_{t+1}(c_j, a_j)) \quad .$$

It is instructive to note that the BBA and Q-learning credit allocation schemes are similar in nature.

6.3.2 Isolated, Concurrent Reinforcement Learners

Reinforcement learning techniques can be used by agents to develop action selection policies to optimize environmental feedback by forming a mapping between perceptions and actions. A particular advantage of these techniques is the fact that they can be used in domains in which agents have little or no pre-existing domain expertise, and have little information about the capabilities and goals of other agents. The lack of this useful information makes the coordination problem particularly hard. Almost all currently used coordination mechanisms rely heavily on domain knowledge and shared information between agents. The position espoused here is that reinforcement learning approaches can be used as new coordination techniques for domains where currently available coordination schemes are ineffective.

A related question is: should agents choose not to use communication while learning to coordinate (see 6.5)? Though communication is often helpful and indispensable as an aid to group activity, it does not guarantee coordinated behavior [20], is time-consuming, and can detract from other problem-solving activity if not carefully controlled [16]. Also, agents overly reliant on communication will be severely affected if the quality of communication is compromised (broken communication channels, incorrect or deliberately misleading information, etc.). At other times, communication can be risky or even fatal (as in some combat situations where the adversary can intercept communicated messages). Even when communication is feasible and safe, it is prudent to use it only when absolutely necessary. Such a design philosophy produces systems where agents do not flood communication channels with unwarranted information. As a result, agents do not have to shift through a maze of useless data to locate necessary and time-critical information.

In the isolated, concurrent form of learning discussed here, each agent learns to optimize its reinforcement from the environment. Other agents in the environment are not explicitly modeled. As such, an interesting research question is whether it is feasible for such an agent to use the same learning mechanism in both cooperative and non-cooperative environments.

An underlying assumption of most reinforcement learning techniques is that the dynamics of the environment is not affected by other agencies. This assumption is invalid in domains with multiple, concurrent learners. A valid concern, therefore, is whether standard reinforcement learning techniques will be adequate for concurrent, isolated learning of coordination. More generally, the following dimensions were identified to characterize domains amenable to concurrent, isolated, reinforcement learning (referred to as CIRL henceforth) approach:

Agent coupling: In some domains the actions of one agent strongly and frequently affect the plans of other agents (tightly coupled system), whereas in other domains the actions of one agent only weakly and infrequently affect the plans of other agents (loosely coupled system).

Agent relationships: Agents in a multiagent system can have different kinds of mutual relationships:

- they may act in a group to solve a common problem (cooperative agents),
- they may not have any preset disposition towards each other but interact because they use common resources (indifferent agents),
- they may have opposing interests (adversarial agents).

For the discussions in this chapter, the latter two classes of domains are grouped as non-cooperative domains.

Feedback timing: In some domains, the agents may have immediate knowledge of the effects of their actions, whereas in others they may get the feedback for their actions only after a period of delay.

Optimal behavior combinations: How many behavior combinations of participating agents will optimally solve the task at hand? This value varies from one to infinity for different domains.

To evaluate these questions, both Q-learning and classifier systems were used in three different domains:

Block pushing: Two agents individually learn to push a box from a starting location to a goal location along a given trajectory. Both cooperative (two agents have same goal location) and competitive (two agents have distinct goal locations) situations are studied. Feedback is based on the deviation of box location from desired path. Domain characteristics are: concurrent learning by two agents with immediate environmental feedback; strongly coupled system; multiple optimal behaviors.

Resource sharing: Given individual task loads, two agents have to learn to share a resource over a time period. Domain characteristics are: delayed environmental feedback; strongly coupled system; single optimal behavior.

Robot navigation: Two robots learn to navigate intersecting paths on a grid without colliding. Domain characteristics: immediate environmental feedback; variable coupling; multiple optimal behaviors.

The basic conclusion from these series of experiments is that CIRL provides a novel paradigm for multiagent systems through which both friends and foes can concurrently acquire useful coordination knowledge. Neither prior knowledge about domain characteristics nor an explicit model about capabilities of other agents is required. The limitation of this approach lies in the inability of CIRL to develop effective coordination when agent actions are strongly coupled, feedback is delayed, and there is one or only a few optimal behavior combinations. A possible partial fix to this problem would be to use some form of staggered or lock-step learning. In this approach, each agent can learn for a period of time, then execute its current policy without modification for some time, then switch back to learning, etc. Two agents can synchronize their behavior so that one is learning while the other is following a fixed policy and vice versa. Even if perfect synchronization is infeasible, the staggered learning mode is likely to be more effective than the concurrent learning mode.

Other interesting observations include the following:

- In cooperative situations, agents can learn complimentary policies to solve the problem. This amounts to role specialization rather than developing identical behavior. This phenomenon has been observed by other researchers when global reinforcement is used [1].
- Agents can transfer learning to similar situations, i.e., once agents learn to coordinate for a given problem, they can learn to coordinate quickly for a similar problem.

6.3.3 Interactive Reinforcement Learning of Coordination

In contrast to the above-mentioned work, Weiss [62] investigates agents explicitly communicating to decide on individual and group actions. The learning approach used is a modification of the BBA scheme for classifier systems. In this approach, agents can observe the set of actions being considered by other agents, and accordingly can eliminate incompatible actions from its local choices. Two variants of the BBA algorithm, the Action Estimation (ACE) and Action Group Estimation (AGE) algorithms, are investigated that requires varying degree of involvement and coordination effort on the part of the group members. The underlying assumption of this work is that the agents are working to optimize a group goal. Below simplified versions of the ACE and AGE algorithms are presented. An algorithm called Dissolution and Formation of Groups (DFG), which is based on these two algorithms but explicitly models group development processes, is described in [61].

Action Estimation Algorithm (ACE): Given its perception, S_i, of the current environmental state, S, each agent, a_i, in a group first calculates the set of actions, $A_i(S)$, it can execute in that state. For each such executable action, $A_i^j \in A_i(S)$, an agent calculates the goal relevance, $E_i^j(S)$, of that action. For all actions whose estimated goal relevance is above a threshold, the agent calculates and announces to other agents a bid that is proportional to its goal relevance plus a noise term, β (to prevent convergence to local minima):

$$B_i^j(S) = (\alpha + \beta) E_i^j(S) \quad ,$$

where α is a small constant *risk factor*.

The action with the highest bid is selected for execution, and incompatible actions are eliminated from consideration. This process is repeated until all actions for which bids were submitted are either selected or eliminated. Selected actions form the *activity context*, $\mathcal{A}$. Then a BBA type mechanism is used to reduce the estimates of the selected action, with the total reduced amount being distributed among actions in the previous activity context. If upon the execution of actions in the current activity context the system receives external payoff, the latter is equally distributed among the executed actions. The goal of this estimate reassignment is to enable successful action sequences to increase in estimate over time and to suppress the estimates of ineffective actions. The net estimate update for any action selected for execution is as follows:

$$E_i^j(S) \leftarrow E_i^j(S) - B_i^j(S) + \frac{R}{|\mathcal{A}|} \quad ,$$

where R is the external rewards received. The bid values paid out are then summed up and redistributed equally between all actions A_k^l executed in the immediately previous activity context, $\mathcal{B}$, corresponding the previous state S':

$$E_k^l(S') \leftarrow E_k^l(S') + \frac{\sum_{A_i^j \in \mathcal{A}} B_i^j(S)}{|\mathcal{B}|} \quad .$$

Action Group Estimation Algorithm (AGE): In the AGE algorithm, first the applicable actions from all agents in a given environmental state are collected. From these action sets, the set of all activity contexts, $\mathcal{A}(S)$ is calculated where an activity context, $\mathcal{A}$, consists of any set of mutually compatible actions:

$$\mathcal{A}(S) = \{\mathcal{A} : \forall A_k^l, A_i^j \in \mathcal{A},\ A_k^l \text{ and } A_i^j \text{ are compatible } \} \quad .$$

Then, for each activity context, bids are collected from each agent for all of its actions in that activity context:

$$B_i^j(S, \mathcal{A}) = (\alpha + \beta) E_i^j(S, \mathcal{A}) \quad ,$$

where $E_i^j(S, \mathcal{A})$ is a_i's estimate of goal relevance of action A_i^j given its perception S_i of state S and the activity context $\mathcal{A}$. The activity context with the highest sum of bids for the actions contained is selected, and all the actions contained in it are executed by respective agents.

Let $\mathcal{A}$ be the activity context selected as above. Then for each $A_i^j \in \mathcal{A}$ agent a_i modifies its estimate as follows:

$$E_i^j(S, \mathcal{A}) \leftarrow E_i^j(S, \mathcal{A}) - B_i^j(S, \mathcal{A}) + \frac{R}{|\mathcal{A}|} \quad .$$

The total bid paid out in the current activity activity context is distributed among actions executed in the previous activity context in a manner analogous to the ACE algorithm:

$$E_k^l(S', \mathcal{B}) \leftarrow E_k^l(S', \mathcal{B}) + \frac{\sum_{A_i^j \in \mathcal{A}} B_i^j(S, \mathcal{A})}{|\mathcal{B}|} \quad .$$

From the above descriptions, it is clear that the AGE algorithm requires more computational effort. The possible gain is the use of a global view in selecting the activity context. The conjecture is that this will lead to better system performance. To test this conjecture, a multiagent blocks world domain is used, where each agent is capable of performing only some of the necessary operations in the environment.

Experiments demonstrated that both the ACE and AGE algorithms enabled agents to learn coordinated behavior in the sense that the agents were able to much more effectively solve problems compared to random action selection. AGE produced more effective coordination compared to ACE but at the cost of increased higher space and computation costs. Globally optimal performance, however, was not attained because of the limited local perception and the inability to distinguish some distinct global states. Though fairly simple in design, ACE and AGE represent potent designs that can be extended and augmented to enable the use of additional agent knowledge and reasoning abilities.

Recent work on theoretical and experimental issues in multiagent reinforcement learning promises new frameworks for isolated and interactive learning of coordination (e.g., [1, 11, 19, 42, 64]).

6.4 Learning about and from Other Agents

In the last section, scenarios are discussed where agents learned to coordinate their actions. The primary emphasis there was on learning to better cooperate to achieve common tasks. In this section scenarios are considered where agents learn to improve their individual performance. At times such improvement in performance or increase in environmental reward has to come at the expense of other agents in the environment. The emphasis in the learning scenarios presented in this section is on agents trying to learn about other agents in order to better capitalize on available opportunities, and on the question of how learning conducted by an agent can be influenced by other agents. This focus is much concerned with the prediction of the behavior of other agents (including their preferences, strategies, intentions, etc.), with the improvement and refinement of an agent's behavior by interacting with and observing other agents, and with the development of a common view of the world.

Since space restrictions preclude the possibility of discussing all published research in this area, a few representative samples from literature were chosen for illustration:

Learning organizational roles: Agents in groups need to learn role assignments to effectively complement each other. Adapting group structure and individual member activities in a situation-dependent manner enables a group to enhance system performance and meet unforeseen challenges. Nagendra Prasad, Lesser, and Lander [35] present a formalism that combines memory-based reasoning and reinforcement learning to enable group members to adaptively select organizational roles.

Learning to benefit from market conditions: Information agents selling and buying information units in an electronic marketplace need to be adaptive to their environmental conditions. Vidal and Durfee investigate the advantages of learning agents that learn models of other agents [58]. They empirically characterize situations when it is beneficial for agents selling information to model other sellers and prospective buyers.

Learning to play better against an opponent: In adversarial domains like board games, classical maximin strategy provides a conservative approach to playing games. If the strategy used by the opponent to choose moves can be approximated, exploitation of weaknesses in the strategy can lead to better results when playing against that particular opponent [7, 46].

All of the domains discussed below involve isolated learning in a distributed sense. One or more agents may be concurrently learning in the environment. The agents interact frequently, and information from such interactions is used by agents to develop models about other agents. Since each agent learns separately, every agent has to execute all learning activities. Most of the learning mechanisms used are variants of reinforcement learning approaches discussed before.

6.4.1 Learning Organizational Roles

Nagendra Prasad, Lesser, and Lander [35] address the important multiagent learning problem of agents learning to adopt situation-specific roles in a cooperative problem-solving domain. Each agent is assumed to have the capability of playing one of several roles in a situation. The learning goal is for an agent to be able to select the most appropriate role to play in a problem-solving state that is likely to lead to better problem solving with less cost.

The basic framework includes the use of Utility, Probability and Cost (UPC) estimates of a role adopted at a particular situation. World states, $\mathcal{S}$, are mapped into a smaller set of situations. Utility represents an agent's estimate of a desired final state's worth if the agent adopted the given role in the current situation. Probability represents the likelihood of reaching a successful final state given the agent plays the adopted role in the current situation, and cost is the associated computational cost incurred. In addition, potentials for roles are maintained, which estimate the usefulness of a role in discovering pertinent global information and constraints. This measure can be orthogonal to the utility measure.

Let S_k and R_k be the sets of situation vectors and roles for agent k respectively. An agent maintains up to $|S_k| * |R_k|$ vectors of UPC and potential values describing the estimates of different roles in different situations. During the learning phase, the probability of selecting a given role r in a situation s is given by

$$Pr(r) = \frac{f(U_{rs}, P_{rs}, C_{rs}, Potential_{rs})}{\sum_{j \in R_k} f(U_{js}, P_{js}, C_{js}, Potential_{js})} \quad ,$$

where f is an objective function used to rate a role by combining the different component measures mentioned before. After the learning phase is over, the role to be played in situation s is chosen deterministically as follows:

$$r = \arg\max_{j \in R_k} f(U_{js}, P_{js}, C_{js}, Potential_{js}) \quad .$$

The abstracting of states to situations, and selecting the highest rated role for the situation is suggestive of a memory based approach. The estimation of role UPC and potential values, however, is learned using a reinforcement learning framework. Repeated problem solving is used to incrementally update estimates of these values. Let $\hat{U}^n_{rs}$, $\hat{P}^n_{rs}$, $\widehat{Potential}^n_{rs}$, represent estimates of the utility, probability, and potential of role r in situation s after n updates. Let S be the situations encountered between the time of adopting role r in situation s and reaching a final state F. A learning rate of $0 \leq \alpha \leq 1$ is used for updating estimates.

If U_F is the utility of the final state reached, then the utility values are updated as follows:

$$\hat{U}^{n+1}_{rs} \leftarrow (1 - \alpha)\hat{U}^n_{rs} + \alpha U_F \quad .$$

This and other updates shown below are performed for all roles chosen in each of the situations, S, that are encountered on the path to the final state.

Let $O : \mathcal{S} \rightarrow [0, 1]$, be a function which returns 1 if the given state is successful and 0 otherwise. Then the update rule for probability is as follows:

$$\hat{P}^{n+1}_{rs} \leftarrow (1 - \alpha)\hat{P}^n_{rs} + \alpha O(F) \quad .$$

Let $Conf(S)$ be a function which returns 1 if in the path to the final state, conflicts between agents are detected followed by information exchange to resolve these conflicts. $Conf(S)$ returns 0 otherwise. Then the update rule for potential is

$$\widehat{Potential}^{n+1}_{rs} \leftarrow (1 - \alpha)\widehat{Potential}^n_{rs} + \alpha Conf(S) \quad .$$

The update rules for cost are domain dependent as is the nature of the function f. Prasad, Lesser, and Lander have successfully used this learning organization role approach in a steam condenser design domain. The evaluation function used by them ignores the cost metric: $f(U, P, C, Potential) = U * P + Potential$.

Related Approaches to Learning Organizational Roles

In a related approach, Haynes and Sen [22] present a multiagent case-based learning (MCBL) algorithm by which agents can learn complementary behaviors to improve group performance. The domain of experimentation is the predator-prey domain [57]. Agents are initialized with hand-crafted behavioral strategies which are modified based on their interaction with the world. Failures to successfully execute actions suggested by default rules trigger learning of negative cases. These negative cases alter the agent policies, and with experience, team members are shown to improve problem-solving performance.

Stone and Veloso [55] investigate the effectiveness of teammates learning to coordinate their actions against opponent teams. The domain of study is a simulated robotic soccer game. Their approach is interesting in the novel use of a layered learning methodology, where learning of low-level skills is followed by learning of higher-level decision making. For example, a neural network–based approach is used to learn how to shoot the ball towards a chosen direction. After this skill is acquired, a decision tree–based method is used to select a teammate to pass the ball to. Higher-level decision making in the context of a team of agents, such as moving into open positions expecting a pass from the teammate with the ball, is possible in such a layered learning approach.

6.4.2 Learning in Market Environments

Vidal and Durfee [58] investigate the use of agents to buy and sell information in electronic marketplaces like digital libraries. They assume such environments are open in nature as new agents (either buyers or sellers of information) can enter or leave the marketplace at will. A practical approach to implementing such systems would be to consider each agent as a self-interested entity with the goal of maximizing local utility. A market mechanism is used to control the transfer of information units between agents that can supply the information and agents that need it. Quality of information available to different sellers may not be the same, and the pricing and buying decisions are left to individual sellers and buyers respectively.

It is assumed that information can be reproduced arbitrarily at negligible cost and agents have uniform access to all other agents in the marketplace. In such scenarios, a seller needs to provide value-added services to differentiate its products from other sellers. In such a market, a buyer announces for a good it needs. Sellers bid with prices for delivering such goods. The buyer then selects from these bids and pays the corresponding seller the bid price. This seller then provides the good to the buyer. The buyer can assess the quality of the received good only after it receives it from the seller, i.e., it cannot examine the quality of the good before buying. The profit of a seller s in selling a good g at price p is $p - c_s^g$, where c_s^g is its cost of producing that good. If this good was of quality q, its value to a buyer b is $V_b^g(p, q)$. In a transaction, the goal of the buyer and the seller is to maximize value and profit respectively.

Three types of agents are investigated in such a market economy:

0-level agents: These are agents that do not model the behavior of other agents. They set their buying and selling prices based on aggregate past experience.

1-level agents: These are agents that analyze the past behavior of other agents and try to predict their buying or selling price preferences. Other agents, however, are just modeled as 0-level agents or agents with no model of other agents. That is, if an 1-level agent A is modeling a 0-level agent B, A does not consider the fact that B is also modeling A. Note that 1-level agents have information about individual agents in the environment, where 0-level agents just use their aggregate past experience.

2-level agents: These are agents that model other agents as 1-level agents. That is, these agents view other agents as agents which are modeling others as 0-level agents or agents having no models of others.

In the following the strategies of 0-level and 1-level agents are only described concisely. The performance comparison of such agents will be presented next.

Strategy of 0-level Agents

A 0-level buyer chooses the seller s^* for supplying a good g, such that

$$s^* = \arg\max_{s \in S} f^g(p_s^g) \quad ,$$

where S is the set of sellers and the function $f^g(p)$ returns the expected value to the buyer of buying g at price p. This value function is incrementally learned in a reinforcement learning framework:

$$f_{t+1}^g = (1-\alpha) f_t^g(p) + \alpha V_b^g(p,q) \quad ,$$

where α is the learning rate which is decreased over time from a starting value of 1 to a final value close to α_{min}. The buyer also explores randomly (picks a random seller) with probability ϵ, with this probability also decreased over time in a manner similar to that of α.

A seller s has to sell a good g at a price greater than or equal to its cost, i.e., $p_s^g \geq c_s^g$. The actual price p_s^* is chosen to maximize expected profit:

$$p_s^* = \arg\max_{p \in P \& p \geq c_s^g} h_s^g(p) \quad ,$$

where P is the set of prices and the function $h_s^g(p)$ returns the expected profit for the seller if it offers good g at a price p. This expected profit function is learned as

$$h_{t+1}^g(p) = (1-\alpha) h_t^g(p) + \alpha Profit_s^g(p) \quad ,$$

where $Profit_s^g(p) = p - c_s^g$ if it wins the auction and is 0 otherwise.

Strategy of 1-level Agents

A 1-level buyer models each seller by a probability density function, $q_s^g(x)$ over the qualities x returned by s when providing good g in the past. Such a buyer chooses the seller s^* for supplying a good g to obtain the highest expected value:

$$\begin{aligned} s^* &= \arg\max_{s \in S} E(V_b^g(p_s^g, q_s^g(x))) \\ &= \arg\max_{s \in S} \frac{1}{|Q|} \sum_{x \in Q} q_s^g(x) V_b^g(p_s^g, x), \end{aligned}$$

where Q is the set of possible quality levels. The 1-level buyer does not model other buyers.

The 1-level seller models each buyer b for good g by a probability density function $m_b^g(p)$ that returns the probability that b will choose price p for good g. It also models every seller s for good g by a probability density function $n_s^g(y)$, which gives the probability that s will bid y for good g. With these information, the 1-level seller can determine its bid to maximize expected profits as

$$p^* = \arg\max_{p \in P} (p - c_s^g) \prod_{s' \in \overline{s}} \sum_{p'} N(g, b, s, s', p, p') \quad ,$$

where $\overline{s} = S - \{s\}$, and $N(g, b, s, s', p, p') = n_{s'}^g(p')$ if $m_b^g(p') \leq m_b^g(p)$ and is 0 otherwise. The function chooses the best bid by calculating for each possible bid the product of the probability of winning the auction with that bid and the profit from that bid. The probability of winning a bid is obtained by multiplying the probabilities of bidding lower than each of the other sellers. The probability of bidding lower than a given seller is calculated by summing the probabilities corresponding to all bids by that seller for which the buyer will prefer the bid of the learning agent.

Vidal and Durfee [58] simulated different artificial economies with 5 buyers and 8 sellers with the value function used by buyers being $V_b(p, q) = 3q - p$ for all goods. The following list shows the major conclusions from the observed behavior of learning mechanisms described above:

- In a group consisting of 0-level agents only, isolated learning produced equilibrium prices when all seller agents offered goods of the same quality. If the latter condition was violated, price fluctuations prevent equilibrium.
- If buyers are 0-level agents, 1-level sellers can benefit based on price volatility as the buyers try to figure out the price-quality correlation. The 1-level sellers can pretend to be high-quality goods sellers by bidding high prices and thus obtain substantial profits at the expense of the buyer.
- If the buyers are 1-level agents, they learn to buy from sellers who can provide them with the highest value. Interestingly enough, 1-level sellers suffer, because they assume buyers are 0-level agents and hence try to over-price their goods.

The above observations suggest that if the model of the other agents is accurate, an agent can gain substantially from it. But if the model underestimates the true capability of the other agent, the modeling agent can also lose out.

6.4.3 Learning to Exploit an Opponent

Two player zero-sum games have been studied widely within both the game theory and artificial intelligence communities. The most prominent approach in AI for developing game playing programs has been the use of the minimax algorithm (developed from the maximin strategy espoused in the game theory literature). In the absence of any knowledge of the opponent's strategy, the maximin approach assumes that the opponent will chose a move that is the worst from the player's viewpoint.

If an accurate model of the opponent is available, such a model can be used to predict the exact move the opponent is going to play corresponding to each of the moves that the player can play from the current board configuration. Carmel and Markovitch [7] present an M^* algorithm, a generalization of minimax, that can use an opponent model to choose a more appropriate move to play against that player. Given the set of possible game states S, a successor function $\sigma : S \to 2^S$, an opponent model to specify opponent's move from any given state, $\varphi : S \to S$, from a given state s and for a search depth d, the M^* algorithm returns the following value:

$$M(s,d,f,\varphi) = \begin{cases} f(s) & d \leq 0 \\ \max s' \in \sigma(s)(f(s')) & d = 1 \\ \max s' \in \sigma(s)(M(\varphi(s'), d-2, f, \varphi)) & d > 1 \end{cases} \quad .$$

If the player is using an evaluation function of f_0, the standard minimax algorithm can be written as a special form of M as

$$M^0_{(\langle f_0 \rangle, d)}(s) = M(s, d, f_0, M^0_{(\langle -f_0 \rangle, d-1)})$$

which denotes the fact that minimax assumes the opponent is minimizing the player's payoff by searching up to a depth of $d-1$.

If the player was using an evaluation of f_1 and the actual evaluation function, f_0, used by the opponent was known, then another special case of M, the M^1 algorithm, can be defined as

$$M^1_{(\langle f_1, f_0 \rangle, d)}(s) = M(s, d, f_1, M^0_{(\langle f_0 \rangle, d-1)}) \quad .$$

The M^1 algorithm first finds the opponents choice move by performing the opponent's minimax search to depth $d-1$. It then evaluates the selected moves by calling itself recursively to depth $d-2$.

In the general case, it is possible to define the M^n algorithm to be the M algorithm for which $\varphi = M^{n-1}$:

$$M^n_{(\langle f_n,\ldots,f_0\rangle,d)}(s) = M(s, f_n, d, M^{n-1}_{(\langle f_{n-1},\ldots,f_0\rangle,d-1)}) \quad .$$

For example, The player with the M^1 algorithm assumes that its opponent is a M^0 or minimax player, the M^2 player assumes that its opponent is a M^1 player, and so on.

Carmel and Markovitch use the domain of checkers to show that the M^1 player performs better than M^0 or minimax player against different opponents when the model of the opponent is accurately known. The problem in approaches like this is how one gets to know about the evaluation function of the opponent.

In a related work Carmel and Markovitch have developed a learning approach to approximating the opponent model [8]. Given a set of opponent moves from specific board configurations, they first present an algorithm to calculate the depth of search being used by the opponent. If the assumed function model is accurate then few examples suffice to induce the depth of search.

They also present an algorithm to learn the opponent's game-playing strategy. The assumptions made are the following: the opponent's evaluation function is a linear combination of known board features, and the opponent does not change its function while playing (because this would eliminate the possibility of concurrent learning). A hill-climbing approach is used to select the weight vector on the features and depth of search. They also experimentally demonstrate the effectiveness of this learning approach for different opponent strategies.

Related Approaches to Opponent Modeling

In a similar approach to developing game players that can exploit weaknesses of a particular opponent, Sen and Arora [46] have used a Maximum Expected Utility (MEU) principle approach to exploiting learned opponent models. In their approach, conditional probabilities for different opponent moves corresponding to all moves from the current state are used to compute expected utilities of each of the possible moves. The move with the maximum expected utility is then played. A probabilistic model of the opponent strategy is developed by observing moves played by the opponent in different discrepancy ranges as measured by the evaluation function of the player.

Let the player and the opponent be required to choose from move sets $\{\alpha_1, \alpha_2, \ldots\} = \alpha$ and $\{\beta_1, \beta_2, \ldots\} = \beta$ respectively, and the utility received by A for a (α_i, β_j) pair of moves be $u(\alpha_i, \beta_j)$. The MEU principle can be used to choose a move as follows:

$$\arg\max_{\alpha_i \in \alpha} \sum_{\beta_j \in \beta} p(\beta_j|\alpha_i)\, u(\alpha_i, \beta_j) \quad ,$$

where $p(\beta_j|\alpha_i)$ is the conditional probability that the opponent chooses the move

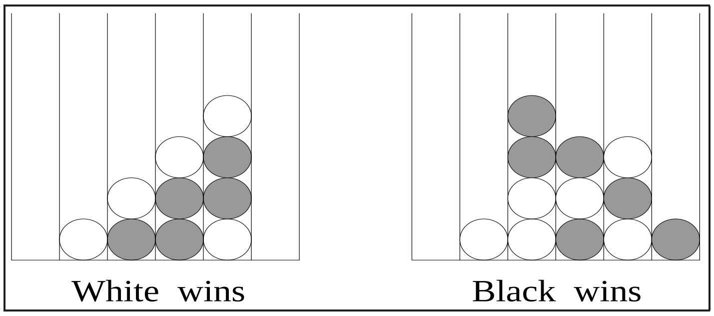

Figure 6.3 Winning scenarios in the game of Connect-Four.

β_j given that the agent plays its move α_i. The maximin strategy can be shown to be a special case of the MEU strategy. If the opponent strategy can be accurately modeled by the learning mechanism, the MEU player will be able to exploit the opponent's weaknesses.

The initial domain of application of this approach involves the two-player zero-sum game of Connect-Four. Connect-Four is a popular two-player board game. Each player has several round tokens of a specific color (black or white). The board is placed vertically and is divided into six slots (the actual game sold in the market has seven slots, but most of the AI programs use the six-slot version of the game). Each slot has room for six tokens. Players alternate in making moves. A player wins if it is able to line up four tokens horizontally, vertically, or diagonally. The game ends in a draw if the board fills up with neither player winning. Examples of winning and losing scenarios are shown in Figure 6.3. In this board game, the MEU player is shown to be able to beat a simple opponent in fewer moves compared to the maximin player.

Other related work worthy of mention include Carmel and Markovitch's work on modeling opponent strategies with a finite automaton [9]; Bui, Kieronska and Venkatesh's work on learning probabilistic models of the preferences of other agents in the meeting scheduling domain [5]; and Zeng and Sycara's work on using Bayesian updating by bargainers to learn opponent preferences in sequential decision making situations [69].

Explanation-Based Learning

Sugawara and Lesser [56] present an explanation-based learning [17] approach to improving cooperative problem-solving behavior. Their proposed learning framework contains a collection of heuristics for recognizing inefficiencies in coordinated behavior, identifying control decisions causing such inefficiencies, and rectifying these decisions.

The general procedure is to record problem-solving traces including tasks and operations executed, relationships existing between tasks, messages communicated between agents, resource usage logs, domain data, and knowledge and control knowledge used for problem solving. Local traces and models of problem-solving activities are exchanged by agents when a coordination inefficiency is detected. This information is used to construct a global model and to review the problem-solving activities. A *lack-of-information* problem is solved by choosing alternative tasks to satisfy certain goals. An *incorrect-control* problem requires more elaborate processing and coordination strategies need to be altered in such cases.

To identify the type of problem confronting the system, agents analyze traces to identify mainstream tasks and messages. Based on this identification, *learning analysis problem* (LAPs) situations are identified which include execution of unnecessary actions, task processing delays, longer task durations, redundant task processing, etc. After some LAP is detected, agents try to locally generate the existing task relationships that may have caused the LAP. Information is exchanged incrementally to form a more comprehensive description of the problem. The purpose of this analysis is to identify whether the LAP is of *lack-of-control* or *incorrect control* problem type. Problems of the former type can normally be resolved in a relatively straightforward manner. For incorrect-control problems, the following solution methods are applied: changing the rating of specific goals and messages, changing the order of operations and communications, allocating tasks to idle agents, and using results calculated by other agents. For both types encountered, the system learns to avoid similar problems in the future. To accomplish this, the system learns situation-specific rules using an inductive learning scheme.

The learning approach discussed above relies extensively on domain models and sophisticated diagnostic reasoning. In contrast, most of the other multiagent learning approaches that have been studied in literature rely very little on prior domain knowledge.

6.5 Learning and Communication

The focus of this section is on how learning and communication are related to each other. This relationship is mainly concerned with requirements on the agents' ability to effectively exchange useful information. The available work on learning in multiagent systems allows us to identify two major relationships and research lines:

- *Learning to communicate*: Learning is viewed as a method for reducing the load of communication among individual agents.

- *Communication as learning*: Communication is viewed as a method for exchanging information that allows agents to continue or refine their learning activities.

Work along the former line starts from the fact that communication usually is very slow and expensive, and therefore should be avoided or at least reduced whenever this is possible (see also 6.3.2). Work along the latter line starts from the fact that learning (as well as, e.g., planning and decision making) is inherently limited in its potential effects by the information that is available to and can be processed by an agent. Both lines of research have to do with improving communication and learning in multiagent systems, and are related to the following issues:

- What to communicate (e.g., what information is of interest to the others).
- When to communicate (e.g., what efforts should an agent investigate in solving a problem before asking others for support).
- With whom to communicate (e.g., what agent is interested in this information, what agent should be asked for support).
- How to communicate (e.g., at what level should the agents communicate, what language and protocol should be used, should the exchange of information occur directly—point-to-point and broadcast—or via a blackboard mechanism).

These issues have to be addressed by the system designer or derived by the system itself. The following two subsections illustrate the two lines of research by describing representative approaches to "learning to communicate" and "communication as learning."

There is another aspect that is worth stressing when talking about learning and communication in multiagent systems. A necessary condition for a useful exchange of information is the existence of a common ontology. Obviously, communication is not possible if the agents assign different meanings to the same symbols without being aware of the differences (or without being able to detect and handle them). The development of a common and shared meaning of symbols therefore can be considered as an essential learning task in multiagent systems (see [18] for further considerations). This "shared meaning problem" is closely related to (or may be considered as the DAI variant of) the symbol grounding problem [21], that is, the problem of grounding the meaning of symbols in the real world. According to the physical grounding hypothesis [4], which has received particular attention in behavior-oriented AI and robotics, the grounding of symbols in the physical world is a necessary condition for building a system that is intelligent. This hypothesis was formulated as a counterpart to the symbol system hypothesis [36] upon which classical knowledge-oriented AI is based and which states that the ability to handle, manipulate, and operate on symbols is a necessary and sufficient condition for general intelligence (independent of the symbols' grounding).

6.5.1 Reducing Communication by Learning

Consider the contract-net approach (e.g., [54]) as described in Chapter 2. According to this approach the process of task distribution consists of three elementary activities: announcement of tasks by *managers* (i.e., agents that want to allocate tasks to other agents); submission of bids by potential *contractors* (i.e., agents that could execute announced tasks); and conclusion of contracts among managers and contractors. In the basic form of the contract net a broadcasting of task announcements is assumed. This works well in small problem environments, but runs into problems as the problem size—the number of communicating agents and the number of tasks announced by them—increases. What therefore is needed in more complex environments are mechanisms for reducing the communication load resulting from broadcasting. Smith [53] proposed several such mechanisms like focused addressing and direct contracting which aim at substituting point-to-point communication for broadcasting. A drawback of these mechanisms is, however, that direct communication paths must be known in advance by the system designer, and that the resulting communication patterns therefore may be too inflexible in non-static environments. In the following, an alternative and more flexible learning-based mechanism called addressee learning [37] is described (in a slightly simplified form).

The primary idea underlying addressee learning is to reduce the communication efforts for task announcement by enabling the individual agents to acquire and refine knowledge about the other agents' task solving abilities. With the help of the acquired knowledge, tasks can be assign more directly without the need of broadcasting their announcements to all agents. Case-based reasoning (e.g., [27, 60]) is employed as an experience-based mechanism for knowledge acquisition and refinement. Case-based reasoning is based on the observation that humans often solve a problem on the basis of solutions that worked well for similar problems in the past. Case-based reasoning aims at constructing cases, that is, problem-solution pairs. Whenever a new problem arises, it is checked whether it is completely unknown or similar to an already known problem (case retrieval). If it is unknown, a solution must be generated from scratch. If there is some similarity to a known problem, the solution of this problem can be used as a starting point for solving the new one (case adaptation). All problems encountered so far, together with their solutions, are stored as cases in the case base (case storage). This mechanism can be applied to communication reduction in a contract net as follows. Each agent maintains its own case base. A case is assumed to consist of *(i)* a task specification and *(ii)* information about which agent already solved this task in the past and how good or bad the solution was. The specification of a task T_i is of the form

$$T_i = \{A_{i1}V_{i1}, \ldots, A_{im_i}V_{im_i}\} \quad ,$$

where A_{ij} is an attribute of T_i and V_{ij} is the attribute's value. What is needed in order to apply case-based reasoning is a measure for the similarity between the tasks. In the case of addressee learning, this measure is reduced to the similarity between attributes and attribute values. More precisely, for each two attributes A_{ir} and A_{js} the distance between them is defined as

$$\mathrm{DIST}(A_{ir}, A_{js}) = \text{SIMILAR-ATT}(A_{ir}, A_{js}) \cdot \text{SIMILAR-VAL}(V_{ir}, V_{js}) \quad ,$$

where SIMILAR-ATT and SIMILAR-VAL express the similarity between the attributes and the attribute values, respectively. How these two measures are defined depends on the application domain and on the available knowledge about the task attributes and their values. In the most simplest form, they are defined as

$$\text{SIMILAR-ATT}(x, y) = \text{SIMILAR-VAL}(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \quad ,$$

which means that similarity is equal to identity. With the help of the distance DIST between attributes, now the similarity between two tasks T_i and T_j can be defined in an intuitively clear and straightforward way as

$$\mathrm{SIMILAR}(T_i, T_j) = \sum_r \sum_s \mathrm{DIST}(A_{ir}, A_{js}) \quad .$$

For every task, T_i, a set of similar tasks, $S(T_i)$, can be defined by specifying the demands on the similarity between tasks. An example of such a specification is

$$S(T_i) = \{T_j : \mathrm{SIMILAR}(T_i, T_j) \geq 0.85\} \quad ,$$

where the tasks T_j are contained in the case base of the agent searching for similar cases. Now consider the situation in which a agent N has to decide about assigning some task T_i to another agent. Instead of broadcasting the announcement of T_i, N tries to preselect one or several agents which it considers as appropriate for solving T_i by calculating for each agent M the suitability

$$\mathrm{SUIT}(M, T_i) = \frac{1}{|S(T_i)|} \sum_{T_j \in S(T_i)} \mathrm{PERFORM}(M, T_j) \quad ,$$

where PERFORM(M, T_j) is an experience-based measure indicating how good or bad T_j has been performed by M in the past. (The specification of PERFORM again depends on the application domain.) With that, agent N just sends the announcement of T_i to the most appropriate agent(s), instead of all agents.

6.5.2 Improving Learning by Communication

As an agent usually can not be assumed to be omnipotent, in most problem domains it also can not be assumed to be omniscient without violating realistic assumptions. The lack of information an agent suffers from may concern

- the environment in which it is embedded (e.g., the location of obstacles) and the problem to be solved (e.g., the specification of the goal state to be reached);
- other agents (e.g., their abilities, strategies, and knowledge);
- the dependencies among different activities and the effects of one own's and other agents' activities on the environment and on potential future activities (e.g., an action a carried out by an agent A may prevent an agent B from carrying out action b and enable an agent C to carry out action c).

Agents having a limited access to relevant information run the risk of failing in solving a given learning task. This risk may be reduced by enabling the agents to explicitly exchange information, that is, to communicate with each other. Generally, the following two forms of improving learning by communication may be distinguished:

- learning based on low-level communication, that is, relatively simple query-and-answer interactions for the purpose of exchanging missing pieces of information (knowledge and belief); and
- learning based on high-level communication, that is, more complex communicative interactions like negotiation and mutual explanation for the purpose of combining and synthesizing pieces of information.

Whereas the first form of communicative learning results in shared information, the second form results in shared understanding. Below two communication-based learning approaches are described which illustrate these two forms.

In both forms communication is used as a means for improving learning. Aside from this "assisting view" of communication, the reader should keep in mind that communication as such can be viewed as learning, because it is a multiagent-specific realization of knowledge acquisition. Whether learning should be enriched by communication is a very difficult question. In the light of the standard evaluation criteria for learning algorithms—speed, quality, and complexity—this question can be decomposed into the following three subquestions:

- How fast are the learning results achieved with/without communication?
- Are the learning results achieved with/without communication of sufficient quality?
- How complex is the overall learning process with/without communication?

The above considerations should make clear that communication offers numerous possibilities to improve learning, but that it is not a panacea for solving learning problems in multiagent systems. Combining them therefore has to be done very carefully. In particular, it is important to see that communication itself may bring in incomplete and false information into an agent's information base (e.g., because of transmission errors) which then makes it even more difficult to solve a learning task.

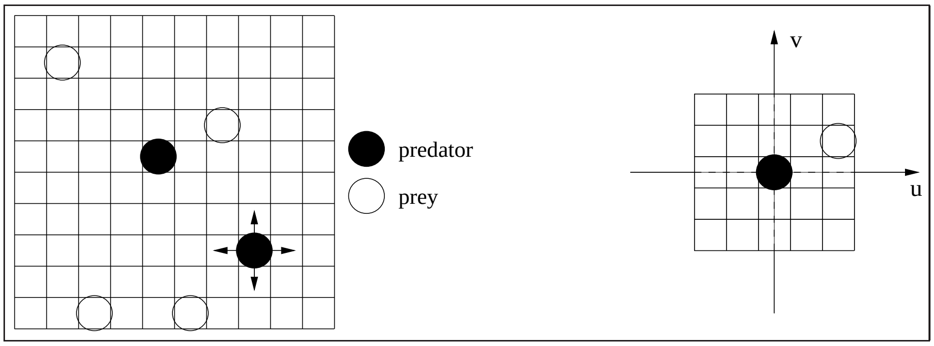

Figure 6.4 Predator-prey domain: a 10 by 10 grid world (left) and a visual field of depth 2 (right).

Illustration 1: Let's Hunt Together!

Many attempts have been made to improve learning in multiagent systems by allowing low-level communication among the learners. Among them is the work by Tan [57] which is also well suited for illustrating this form of learning. Related work that focuses on multirobot learning was presented, e.g., by Matarić [31, 32] and Parker [38, 39].

Tan investigated learning based on low-level communication in the context of the predator-prey domain shown in Figure 6.4. The left part of this figure shows a two-dimensional world in which two types of agents, predators and prey, act and live. The task to be solved by the predators is to catch a prey by occupying the same position. Each agent has four possible actions a to choose from: *moving up*, *moving down*, *moving left*, and *moving right*. On each time step each prey randomly moves around and each predator chooses its next move according to the decision policy it has gained through Q-learning (see Section 6.3.1). Each predator has a limited visual field of some predefined depth. The sensation of a predator is represented by $s = [u, v]$, where u and v describe the relative distance to the closest prey within its visual field. This is illustrated by the right part of Figure 6.4; here the perceptual state is represented by $[2, 1]$. Tan identified two kinds of information that the learners could exchange in order to support each other in their learning:

- *Sensor data.* Here the predators inform each other about their visual input. If the predators know their relative positions (e.g., by continuously informing each other about their moves), then they can draw inferences about the prey's actual positions. This corresponds to a pooling of sensory resources, and thus aims at a more centralized control of distributed sensors.
- *Decision/Activity policies.* Here the predators inform each other about what they have learned so far w.r.t. their decisions/activities (i.e., the values $Q(s, a)$ in the case of Q-learning). This corresponds to a pooling of motor resources, and thus aims at a more centralized control of distributed effectors.

The experimental investigations reported by Tan show that these kinds of information exchange clearly lead to improved learning results. The fact that these two kinds of information exchange are applicable in most problem domains makes them essential. It is stressed that it is an important but still unanswered question how closely a centralized control of sensors and effectors should be approached. It is obvious, however, that an optimal degree of centralization of control depends on the problem domain under consideration and on the abilities of the individual agents.

Illustration 2: What Will a Cup of Coffee Cost?

Learning based on high-level communication—which is a characteristic of human-human learning—is rather complex, and so it is not surprising that not many approaches to this form of learning are available so far. In the following, an idea of this form of learning is given by describing the approach by Sian [48, 49] called consensus learning (details omitted and slightly simplified). According to this approach a number of agents is assumed to interact through a blackboard. The agents use a simple language for communication that consists of the following nine operators for hypotheses:

- *Introduction and removal of hypotheses* to/from the blackboard

  $ASSERT(H)$ – Introduction of a non-modifiable hypothesis H.

  $PROPOSE(H,C)$ – Proposal of a new hypothesis H with confidence value C.

  $WITHDRAW(H)$ – Rejection of a hypothesis H.

- *Evaluation of hypotheses*

  $CONFIRM(H,C)$ – Indication of confirmatory evidence for a hypothesis H with confidence value C.

  $DISAGREE(H,C)$ – Indication of disagreement with a hypothesis H with confidence value C.

  $NOOPINION(H)$ – Indication that no opinion is available with regards to a hypothesis H.

  $MODIFY(H,G,C)$ – Generation of a modified version G (hence, of a new hypothesis) of H with confidence value C.

- *Modification of the status of hypotheses and acceptance*

  $AGREED(H,T)$ – Change of status of a hypothesis H from "proposed" to "agreed" with the resultant confidence value T (see below).

  $ACCEPT(H)$ – Acceptance of a previously agreed hypothesis H.

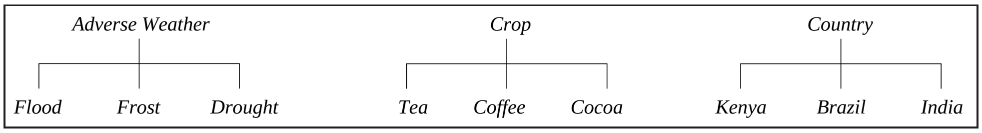

Figure 6.5 Taxonomies available to the agents.

After an agent introduced a hypothesis H (by means of $PROPOSE$) and the other agents responded (by means of $CONFIRM$, $DISAGREE$, $NOOPINION$, or $MODIFY$), the introducing agent can determine the resultant confidence value T of H. Let $\{C_1^+, \ldots, C_m^+\}$ be the confidence values associated with the $CONFIRM$ and $MODIFY$ responses of the other agents, and $\{C_1^-, \ldots, C_n^-\}$ the confidence values associated with the $DISAGREE$ responses of the other agents. Then

$$T = SUPPORT(H) \cdot [1 - AGAINST(H)]$$

where $SUPPORT(\mathrm{H}) = V(C_m^+)$ and $AGAINST(H) = V(C_n^-)$ with

$$V(C_m^+) = \begin{cases} V(C_{m-1}^+) + C_m^+ \cdot [1 - V(C_{m-1}^+)] & if \quad m \geq 1 \\ 0 & if \quad m = 0 \end{cases}$$

and

$$V(C_n^-) = \begin{cases} V(C_{n-1}^-) + C_n^- \cdot [1 - V(C_{n-1}^-)] & if \quad n \geq 1 \\ 0 & if \quad n = 0 \end{cases} .$$

For instance, $V(C_3^+) = C_1^+ + C_2^+ + C_3^+ - C_1^+C_2^+ - C_1^+C_3^+ - C_2^+C_3^+ + C_1^+C_2^+C_3^+$. The definition of V aims at adding confidence values (which represent a measure of belief on the part of an agent) and, at the same time, taking their potential overlaps into consideration.

For an illustration of consensus learning, consider the case of three agents who want to find out how the prices for coffee, tea, and cocoa will develop. The common knowledge available to the three agents is shown in Figure 6.5. In addition, the agents have the following local domain knowledge:

Agent 1:	$Major\text{-}Producer(Kenya, Coffee)$ $Major\text{-}Producer(Kenya, Tea)$
Agent 2:	$Major\text{-}Producer(Brazil, Coffee)$ $Major\text{-}Producer(Brazil, Cocoa)$
Agent 3:	$Major\text{-}Producer(India, Tea)$

Assume that after a period of time the agents observed the following data and have constructed the following generalizations:

Agent 1:	$Weather(Kenya, Drought)$, $Price(Tea, Rising)$ $Weather(Kenya, Drought)$, $Price(Cocoa, Steady)$

$$
\begin{array}{ll}
 & Weather(Kenya, Frost),\ Price(Coffee, Rising) \\
 & \text{GEN: } Weather(Kenya, Adverse)\ and \\
 & \qquad Major\text{-}Producer(Kenya, Crop) \rightarrow Price(Crop, Rising) \\
\text{Agent 2:} & Weather(Brazil, Frost),\ Price(Coffee, Rising) \\
 & Weather(Brazil, Flood),\ Price(Cocoa, Rising) \\
 & \text{GEN: } Weather(Brazil, Adverse) \rightarrow Price(Crop, Rising) \\
\text{Agent 3:} & Weather(India, Flood),\ Price(Tea, Rising) \\
 & \text{GEN: } Weather(India, Flood) \rightarrow Price(Tea, Rising)
\end{array}
$$

Figure 6.6 shows a potential interaction sequence. The Agent 3 has enough confidence in its generalization, and starts the interaction with the hypothesis $H1$. The other agents respond to $H1$. Agent 2 has no direct evidence for H1, but its generalization totally subsumes $H1$. It therefore proposes its generalization as a modification of $H1$, leading to the hypothesis $H2$. The situation is similar with Agent 3, and this agent proposes the hypothesis $H3$. At this point, Agent 3 can calculate the resultant confidence value for its hypothesis $H1$. In the sequel, the non-proposing agents respond to the hypotheses $H2$ and $H3$, and the proposing agents calculate the resultant confidence values. Based on the confidence values Agent 2 and Agent 3 withdraw their hypotheses. After Agent 1 has agreed, the others accept $H3$. What has been gained is the broad acceptance of the hypothesis $H3$ which is less specific than $H1$ and less general than $H2$.

6.6 Conclusions

Summary. This chapter concentrated on the area of learning in multiagent systems. It was argued that this area is of particular interest to DAI as well as ML. Two principal categories of learning—centralized and decentralized learning—were distinguished and characterized from a more general point of view. Several concrete learning approaches were described that illustrate the current stage of development in this area. They were chosen because they reflect very well the current methodological main streams and research foci in this area: learning and activity coordination; learning about and from other agents; and learning and communication. It is very important to see that these foci are not orthogonal, but complementary to each other. For instance, agents may learn to cooperate by learning about each other's abilities, and in order to learn from one another the agents may communicate with each other. It is stressed that several interesting and elaborated approaches to learning in multiagent systems other than those described here are available. Space did not allow us to treat them all, and the reader therefore is referred to the literature mentioned thoughout this chapter.

Open research issues. Learning in multiagent systems constitutes a relatively young area that brings up many open questions. The following areas of research are of particular interest:

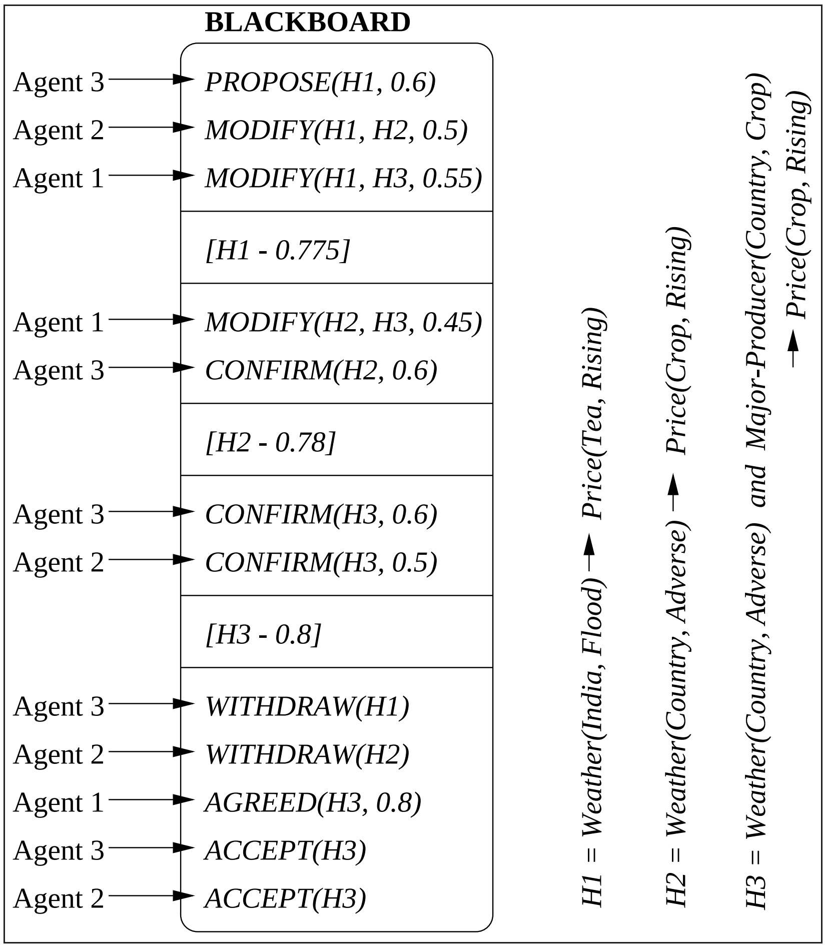

Figure 6.6 An example of an interaction sequence.

- The identification of general principles and concepts of multiagent learning. Along this direction questions arise like *What are the unique requirements and conditions of multiagent learning?* and *Are there general guidelines for the design of multiagent learning algorithms?*
- The investigation of the relationships between single-agent and multiagent learning. This necessitates to answer questions like *Do centralized and decentralized learning qualitatively differ from each other?* and *How and under what conditions can a single-agent learning algorithm be applied in multiagent contexts?*
- The application of multiagent learning in complex real-world environments. Going in this direction helps to further improve our understanding of the benefits and limitations of this form of learning.

- The development of theoretical foundations of decentralized learning. This ranges from convergence proofs for particular algorithms to general formal models of decentralized learning.

An overview of challenges for ML in cooperative information systems is presented in [51]. In this overview a useful distinction is made between requirements for learning about passive components (e.g., databases), learning about active components (e.g., workflows and agents), and learning about interactive components (e.g., roles and organizational structures).

Pointers to relevant related work. As already mentioned, this chapter is restricted to learning in multiagent systems. The reader interested in textbooks on single-agent learning is referred to [28] and [34]. There is a number of approaches to distributed reinforcement learning that are not covered by this chapter; see, e.g., [12, 30, 41, 65]. Moreover, there is much work in ML that does not directly deal with learning in multiagent systems, but is closely related to it. There are three lines of ML research that are of particular interest from the point of view of DAI:

- *Parallel and distributed inductive learning* (e.g., [10, 40, 50]). Here the focus is on inductive learning algorithms that cope with massive amounts of data.
- *Multistrategy learning* (e.g., [33]). Here the focus is on the development of learning systems that employ and synthesize different learning strategies (e.g., inductive and analogical, or empirical and analytical).
- *Theory of team learning* (e.g., [25, 52]). Here the focus is on teams of independent machines that learn to identify functions or languages, and on the theoretical characterization—the limitations and the complexity—of this kind of learning.

Research along these lines is much concerned with the decentralization of learning processes, and with combining learning results obtained at different times and/or locations.

Apart from ML, there is a considerable amount of related work in economics. Learning in organizations like business companies and large-scale institutions constitutes a traditional and well-established subject of study. Organizational learning is considered as a fundamental requirement for an organization's competitiveness, productivity, and innovativeness in uncertain and changing technological and market circumstances. With that, organizational learning is essential to the flexibility and sustained existence of an organization. Part II of the Bibliography provided in [63] offers a number of pointers to this work.

There is also a large amount of related work in psychology. Whereas economics mainly concentrates on organizational aspects, psychology mainly focuses on the cognitive aspects underlying the collaborative learning processes in human groups. The reader interested in related psychological research is referred to [2] and, in particular, to [13]. A guide to research on collaborative learning can be found in [14]. Interdisciplinary research that, among other things, is aimed at identifying essential differences between available approaches to multiagent learning and collaborative human-human learning is described in [67].

These pointers to related work in ML, economics, and psychology are also intended to give an idea of the broad spectrum of learning in multiagent systems. In attacking the open questions and problems sketched above it is likely to be helpful and inspiring to take this related work into consideration.

6.7 Exercises

1. *[Level 1]* Consider a group of students who agreed to work together in preparing an examination in DAI. Their goal is to share the load of learning. Identify possible forms of interactive learning. How do the forms differ from each other (e.g., w.r.t. efficiency and robustness) and what are their advantages and disadvantages? What abilities must the students have in order to be able to participate in the different forms of learning? Do you think it is possible to apply the different forms in (technical) multiagent contexts? What are the main difficulties in such an application?
2. *[Level 2]* Design domains with varying agent couplings, feedback delays, and optimal strategy combinations, and run experiments with isolated reinforcement learners. Summarize and explain the success and failures of developing coordinated behaviors using isolated, concurrent reinforcement learners in the domains that you have investigated.
3. Consider the algorithms ACE and AGE.
   (a) *[Level 2]* Calculate and compare the computational complexities per action selection cycle of both algorithms.
   (b) *[Level 2]* Evaluate the scale up in speed of both algorithms with increasing number of agents in the group.
   (c) *[Level 3]* How could the complexity be reduced? Do you see any possibility to reduce the number of activity contexts to be considered by the agents? Implement and test your solution.
4. *[Level 2/3]* Implement and experiment with 0, 1, and 2-level agents in an information economy. How does 2-level buyer agent benefit compare to 1-level buyer agents when the seller agents are 0-level agents? How does 2-level buyer agent benefit compare to 1-level buyer agents when the seller agents are 1-level agents?
5. Consider the problem of learning an opponent strategy.
   (a) *[Level 2]* Formulate this problem in a two player zero-sum game as a reinforcement learning problem.
   (b) *[Level 3]* Implement a reinforcement learning algorithm to learn the opponent strategy in a simple two-player zero-sum game. Show how

the learned opponent model can be used to exploit weaknesses in the strategies of a weaker player.

6. A popular multiagent learning task is block pushing. As described in this chapter, this task requires that (at least) two agents learn to work together in pushing a box from a start to a goal position, where the box chosen is large enough so that none of the agents can solve this problem alone. This learning task becomes especially challenging under two reasonable assumptions: each agent is limited in its sensory abilities (i.e., its sensors provide incomplete and noisy data), and learning feedback is provided only when the agents are successful in moving the block into the goal position (i.e., no intermediate feedback is provided).

   (a) *[Level 2/3]* Assume that both agents are capable of Q-learning and that they select and perform their actions simultaneously. Furthermore, assume that *(i)* the agents do not communicate and *(ii)* that at each time each of the agents knows only its own position, the goal position, and the position of the block. Implement this learning scenario and run some experiments. What can be observed?

   (b) *[Level 3/4]* Now assume that the agents are able to communicate with each other. What information should they exchange in order to improve their overall performance? Implement your ideas and compare the results with those gained for non-communicating learning agents. Do your ideas result in faster learning? What about the quality of the learning results and the complexity of learning?

7. Another popular learning task is multiagent foraging. This task requires that multiple agents learn to collect food in a confined area (their "living environment") and take it to a predefined region (their "home"). An agent receives positive learning feedback whenever it arrived at home with some food (each agent is able to collect food without requiring help from the others).

   (a) *[Level 1]* What are the essential differences between this learning task and the block pushing task?

   (b) *[Level 2/3]* Assume that the agents are capable of Q-learning. Implement this learning scenario and run some experiments.

   (c) *[Level 3/4]* Additionally assume that that there are two different types of food: food of type A can be carried by a single agent, while food of type B must be carried by two agents. Furthermore assume that the learning feedback for collecting food of type B is four times higher than for type A, and that some agents are better (e.g., faster) in collecting food of type A while others are better in collecting (together with others) food of type B. What information should the agents exchange and what communication and coordination mechanisms should they use in order to collect both type-A and type-B food as fast as possible? Think about equipping the individual agents with the ability to learn about other agents. Implement your ideas, and compare the results with those achieved by the more primitive non-communicating agents (i.e., agents that do neither communicate nor learn about each other).

8. *[Level 3/4]* Consider Exercise 14 of Chapter 1 (vacuum world example). Instead of implementing chains of sample passing agents, the agents themselves could learn to form appropriate chains. (Alternatively, the agents could learn to appropriately divide the vacuum world into smaller sections that are then occupied by fixed sets or teams of agents.) Identify criteria according to which the agents can decide when and how to form chains. Run experiments with the learning agents and analyze, e.g., the orientation and the position of the chains learned. Identify criteria according to which the agents can decide when and how to dissolve chains. Again run experiments. Give particular attention to the learning feedback (immediate vs. delayed) and the communication and negotiation abilities of the agents.
9. *[Level 3/4]* Consider Exercise 11 of Chapter 2 (package-moving robots). How could the robots learn to build appropriate roadways and drop-off points? (What exactly does appropriate mean in this example? What communication and negotiation abilities should the robots possess?) Implement your ideas, and compare the results achieved by learning and non-learning robots.
10. *[Level 3/4]* Consider Exercise 8 of Chapter 4 (multiagent LRTA* algorithm). How could the agents learn to coordinate their activities? What activities should be coordinated at all? What information must be exchanged by the agents in order to achieve a higher degree of coordination? Choose one of the search problems described in Chapter 4, and run some experiments.
11. *[Level 3/4]* Consider Exercise 8 of Chapter 5 (lookahead in contracting). Choose one of the contracting scenarios described in that chapter; alternatively, you may choose the multiagent foraging scenario (see Exercise 7 above), the vacuum world scenario (Exercise 8), or the package-moving domain (Exercise 9). Give examples of criteria for deciding about the depth of lookahead in contracting. Implement an algorithm for lookahead contracting, where the depth of lookahead is adapted by the agents themselves.

6.8 References

1. T. Balch. Learning roles: Behavioral diversity in robot teams. In *Collected Papers from the AAAI-97 Workshop on Multiagent Learning*, pages 7–12. AAAI, 1997.
2. A. Bandura. *Social learning theory.* Prentice-Hall, Englewood Cliffs, NJ, 1977.
3. A.B. Barto, R.S. Sutton, and C. Watkins. Sequential decision problems and neural networks. In *Proceedings of 1989 Conference on Neural Information Processing*, 1989.
4. R.A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*,

6:3–15, 1990.

5. H.H. Bui, D. Kieronska, and S. Venkatesh. Learning other agents' preferences in multiagent negotiation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 114–119, Menlo Park, CA, 1996. AAAI Press.
6. J.G. Carbonell, R.S. Michalski, and T.M. Mitchell. An overview of machine learning. In J.G. Carbonell and T.M. Mitchell, editors, *Machine learning – An artificial intelligence approach*, pages 3–23. Springer-Verlag, Berlin, 1994.
7. D. Carmel and S. Markovitch. Incorporating opponent models into adversary search. In *Thirteenth National Conference on Artificial Intelligence*, pages 120–125, Menlo Park, CA, 1996. AAAI Press/MIT Press.
8. D. Carmel and S. Markovitch. Learning and using opponent models in adversary search. Technical Report Technical Report 9609, Technion, 1996.
9. D. Carmel and S. Markovitch. Learning models of intelligent agents. In *Thirteenth National Conference on Artificial Intelligence*, pages 62–67, Menlo Park, CA, 1996. AAAI Press/MIT Press.
10. P.K. Chan and S.J. Stolfo. Toward parallel and distributed learning by meta-learning. In *Working Notes of the AAAI Workshop on Know. Disc. Databases*, pages 227–240, 1993.
11. C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Collected papers from the AAAI-97 Workshop on Multiagent Learning*, pages 13–18. AAAI, 1997.
12. R.H. Crites and A.G. Barto. Improving elevator performances using reinforcement learning. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in neural information processing systems 8*. MIT Press, Cambridge, MA, 1996.
13. P. Dillenbourg, editor. *Collaborative learning: Cognitive and computational approaches*. Pergamon Press, 1998.
14. P. Dillenbourg, M. Baker, A. Blaye, and C. O'Malley. The evolution of research on collaborative learning. In H. Spada and P. Reimann, editors, *Learning in humans and machines*. Elsevier Science Publ., Amsterdam, 1996.
15. M. Dorigo and H. Bersini. A comparison of Q-learning and classifier systems. In *Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, 1994.
16. E.H. Durfee, V.R. Lesser, and D.D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, 1987.
17. T. Ellman. Explanation-based learning: A survey of programs and perspectives. *ACM Computing Surveys*, 21(2):163–221, 1989.
18. H. Friedrich, M. Kaiser, O. Rogalla, and R. Dillmann. Learning and communication in multi-agent systems. In G. Weiß, editor, *Distributed artificial intelligence meets machine learning*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1221, pages 259–275. Springer-Verlag, Berlin, 1997.
19. P. Gu and A.B. Maddox. A framework for distributed reinforcement learning. In Gerhard Weiß and Sandip Sen, editors, *Adaptation and Learning in Multi–Agent Systems*, Lecture Notes in Artificial Intelligence, pages 97–112. Springer Verlag, Berlin, 1996.
20. J. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. A preliminary version

appeared in *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

21. S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.
22. T. Haynes and S. Sen. Learning cases to compliment rules for conflict resolutions in multiagent systems. *International Journal of Human-Computer Studies*, to appear, 1998.
23. M. Huhns and G. Weiß, editors. Special Issue on Multiagent Learning of the *Machine Learning Journal*. Vol. 33(2-3), 1998.
24. I.F. Imam. Intelligent adaptive agents. Papers from the 1996 AAAI Workshop. Technical Report WS-96-04, AAAI Press, 1996.
25. S. Jain and A. Sharma. On aggregating teams of learning machines. *Theoretical Computer Science A*, 137(1):85–105, 1982.
26. L.P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.
27. J.L. Kolonder. *Case-based reasoning*. Morgan Kaufmann, San Francisco, 1993.
28. P. Langley. *Elements of machine learning*. Morgan Kaufmann, San Francisco, 1995.
29. V.R. Lesser. Multiagent systems: An emerging subdiscipline of AI. *ACM Computing Surveys*, 27(3):340–342, 1995.
30. M.L. Littmann and J.A. Boyan. A distributed reinforcement learning scheme for network routing. Report CMU-CS-93-165, School of Computer Science, Carnegie Mellon University, 1993.
31. M. Matarić. Learning in multi-robot systems. In G. Weiß and S. Sen, editors, *Adaption and learning in multi-agent systems*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1042, pages 152–163. Springer-Verlag, Berlin, 1996.
32. M. Matarić. Using communication to reduce locality in distributed multi-agent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, to appear, 1998.
33. R. Michalski and G. Tecuci, editors. *Machine learning. A multistrategy approach*. Morgan-Kaufmann, San Francisco, CA, 1995.
34. T. Mitchell. *Machine learning*. McGraw-Hill, New York, 1997.
35. M.V. Nagendra Prasad, V.R. Lesser, and S.E. Lander. Learning organizational roles in a heterogeneous multi-agent system. In *Proceedings of the Second International Conference on Multiagent Systems*, pages 291–298, 1996.
36. A. Newell and H.A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.
37. T. Ohko, K. Hiraki, and Y. Anzai. Addressee learning and message interception for communication load reduction in multiple robot environments. In G. Weiß, editor, *Distributed artificial intelligence meets machine learning*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1221, pages 242–258. Springer-Verlag, Berlin, 1997.
38. L.E. Parker. Task-oriented multi-robot learning in behavior-based systems. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1478–1487, 1996.
39. L.E. Parker. L-alliance: Task-oriented multi-robot learning in behavior-based systems. *Journal of Advanced Robotics*, to appear, 1997.
40. F.J. Provost and J.M. Aronis. Scaling up inductive learning with massive parallelism. *Machine Learning*, 23:33f, 1996.

41. A. Schaerf, Y. Shoham, and M. Tennenholtz. Adaptive load balancing: a study in multi-agent learning. *Journal of Artificial Intelligence Research*, 2:475–500, 1995.

42. J. Schmidhuber. A general method for multi-agent reinforcement learning in unrestricted environments. In Sandip Sen, editor, *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 84–87, Stanford University, CA, 1996.

43. S. Sen. Adaptation, coevolution and learning in multiagent systems. Papers from the 1996 Spring Symposium. Technical Report SS-96-01, AAAI Press, 1996.

44. S. Sen. IJCAI-95 workshop on adaptation and learning in multiagent systems. *AI Magazine*, 17(1):87–89, Spring 1996.

45. S. Sen, editor. Special Issue on Evolution and Learning in Multiagent Systems of the *International Journal of Human-Computer Studies*. Vol. 48(1), 1998.

46. S. Sen and N. Arora. Learning to take risks. In *Collected papers from AAAI-97 workshop on Multiagent Learning*, pages 59–64. AAAI, 1997.

47. S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *National Conference on Artificial Intelligence*, pages 426–431, 1994.

48. S.S. Sian. Adaptation based on cooperative learning in multi-agent systems. In Y. Demazeau and J.-P. Müller, editors, *Decentralised AI (Vol. 2)*, pages 257–272. Elsevier Science Publ., Amsterdam, 1991.

49. S.S. Sian. Extending learning to multiple agents: Issues and a model for multi-agent machine learning (ma-ml). In Y. Kodratoff, editor, *Machine learning – EWSL-91*, pages 440–456. Springer-Verlag, Berlin, 1991.

50. R. Sikora and M.J. Shaw. A distributed problem-solving approach to inductive learning. Faculty Working Paper 91-0109, College of Commerce and Business Administration, University of Illinois at Urbana-Champaign, 1991.

51. M.P. Singh and M.N. Huhns. Challenges for machine learning in cooperative information systems. In G. Weiß, editor, *Distributed artificial intelligence meets machine learning*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1221, pages 11–24. Springer-Verlag, Berlin, 1997.

52. C. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982.

53. R.G. Smith. A framework for problem solving in a distributed processing environment. Stanford Memo STAN-CS-78-700, Department of Computer Science, Stanford University, 1978.

54. R.G. Smith. The contract-net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

55. P. Stone and M. Veloso. A layered approach to learning client behaviors in the robocup soccer. *Applied Artificial Intelligence*, to appear, 1998.

56. T. Sugawara and V. Lesser. On-line learning of coordination plans. In *Working Papers of the 12th International Workshop on Distributed Artificial Intelligence*, 1993.

57. M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

58. J.M. Vidal and E.H. Durfee. The impact of nested agent models in an information economy. In *Proceedings of the Second International Conference on Multiagent*

Systems, pages 377–384, Menlo Park, CA, 1996. AAAI Press.

59. C.J.C.H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge University, 1989.
60. I. Watson and F. Marir. Case-based reasoning: A review. *The Knowledge Engineering Review*, 9(4):327–354, 1994.
61. G. Weiß. Action selection and learning in multi-agent environments. In *From animals to animats 2 – Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 502–510, 1993.
62. G. Weiß. Learning to coordinate actions in multi-agent systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 311–316, 1993.
63. G. Weiß. Adaptation and learning in multi-agent systems: Some remarks and a bibliography. In G. Weiß and S. Sen, editors, *Adaption and learning in multiagent systems*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1042. Springer-Verlag, Berlin, 1996.
64. G. Weiß, editor. *Distributed artificial intelligence meets machine learning.* Lecture Notes in Artificial in Artificial Intelligence, Vol. 1221. Springer-Verlag, Berlin, 1997.
65. G. Weiß. A multiagent perspective of parallel and distributed machine learning. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 226–230, 1998.
66. G. Weiß, editor. Special Issue on Learning in Distributed Artificial Intelligence Systems of the *Journal of Experimental and Theoretical Artificial Intelligence.* Vol. 10(3), 1998.
67. G. Weiß and P. Dillenbourg. What is "multi" in multiagent learning? In P. Dillenbourg, editor, *Collaborative learning: Cognitive and computational approaches.* Pergamon Press, 1998.
68. G. Weiß and S. Sen, editors. *Adaption and learning in multiagent systems.* Lecture Notes in Artificial in Artificial Intelligence, Vol. 1042. Springer-Verlag, Berin, 1996.
69. D. Zeng and K. Sycara. Bayesian learning in negotiation. International Journal of Human Computer Studies (to appear), 1998.