Liviu Panait and Sean Luke. 2005. Cooperative Multi-Agent Learning: The State of the Art. In Autonomous Agents and Multi-Agent Systems. 11(3) 387-434. Springer

Cooperative Multi-Agent Learning: The State of the Art

Liviu Panait and Sean Luke
George Mason University

Abstract

Cooperative multi-agent systems are ones in which several agents attempt, through their interaction, to jointly solve tasks or to maximize utility. Due to the interactions among the agents, multi-agent problem complexity can rise rapidly with the number of agents or their behavioral sophistication. The challenge this presents to the task of programming solutions to multi-agent systems problems has spawned increasing interest in machine learning techniques to automate the search and optimization process.

We provide a broad survey of the cooperative multi-agent learning literature. Previous surveys of this area have largely focused on issues common to specific subareas (for example, reinforcement learning or robotics). In this survey we attempt to draw from multi-agent learning work in a spectrum of areas, including reinforcement learning, evolutionary computation, game theory, complex systems, agent modeling, and robotics.

We find that this broad view leads to a division of the work into two categories, each with its own special issues: applying a single learner to discover joint solutions to multi-agent problems (*team learning*), or using multiple simultaneous learners, often one per agent (*concurrent learning*). Additionally, we discuss direct and indirect communication in connection with learning, plus open issues in task decomposition, scalability, and adaptive dynamics. We conclude with a presentation of multi-agent learning problem domains, and a list of multi-agent learning resources.

1 Introduction

In recent years there has been increased interest in decentralized approaches to solving complex real-world problems. Many such approaches fall into the area of *distributed systems,* where a number of entities work together to cooperatively solve problems. The combination of distributed systems and artificial intelligence (AI) is collectively known as *distributed artificial intelligence* (DAI). Traditionally, DAI is divided into two areas. The first area, *distributed problem solving*, is usually concerned with the decomposition and distribution of a problem solving process among multiple slave nodes, and the collective construction of a solution to the problem. The second class of approaches, *multi-agent systems* (MAS), emphasizes the joint behaviors of agents with some degree of autonomy and the complexities arising from their interactions.

In this survey, we will focus on the application of *machine learning* to problems in the MAS area. Machine learning explores ways to automate the inductive process: getting a machine agent to discover on its own, often through repeated trials, how to solve a given task or to minimize error. Machine learning has proven a popular approach to solving multi-agent systems problems because the inherent complexity of many such problems can make solutions by hand prohibitively difficult. Automation is attractive. We will specifically focus on problem domains in which the multiple agents are *cooperating* to solve a joint task or to maximize utility; as opposed to *competing* with one another. We call this specific subdomain of interest *cooperative multi-agent learning*. Despite the relative youth of the field, the number of cooperative multi-agent learning papers is large, and we hope that this survey will prove helpful in navigating the current body of work.

1.1 Multi-Agent Systems

The terms *agent* and *multi-agent* are not well-defined in the community; we offer our own, admittedly broad, definitions of the concepts here as we intend to use them later in the survey. An *agent* is a computational mechanism that

exhibits a high degree of autonomy, performing actions in its environment based on information (sensors, feedback) received from the environment. A *multi-agent* environment is one in which there is more than one agent, where they interact with one another, and further, where there are constraints on that environment such that agents may not at any given time know *everything* about the world that other agents know (including the internal states of the other agents themselves).

These constraints are, we argue, important to the notion of the multi-agent system problem definition. Otherwise, the distributed "agents" can act in sync, knowing exactly what situation the other agents are in and what behavior they will pursue. This "omniscience" permits the agents to act as if they were really mere appendages of a single master controller. Additionally, if the domain requires no interaction at all, then it may be decomposed into separate, fully independent tasks each solvable by a single agent.

Consider the application of cooperative foraging, where multiple robot agents are tasked to discover rock samples and bring them back to specific places. We will return to this example throughout the paper. In its simplest form, this is a problem domain which could be solved by a single robot, and multiple (totally independent) robots merely parallelize the effort. The problem becomes more complex when the robots can, through their interactions, gradually suggest to one another good regions to forage. Additionally, if all robots instantaneously know of all rock discoveries, and further know where the other robots will choose to forage, they could be coded to operate identically to a basic master-slave configuration. As Jennings et al. [131] suggest, few real-world problem domains permit such reductions to single-agent equivalence: most feature agents with incomplete information about the environment, lack of centralized control, decentralized and distributed information, and asynchronous computation.

Depending on their interest, several authors have provided different taxonomies for MAS applications. For example, Dudek et al. [71] classify swarm robotics applications according to team size, range, communication topology and throughput, team composition and reconfigurability, and the processing ability of individual agents. In a collection describing the application of Distributed Artificial Intelligence to industry, Van Dyke Parunak [273] differentiates between *agent characteristics* (team heterogeneity, control architectures, input/output abilities) and *system characteristics* (for example, communication settings). Stone and Veloso [255] explicitly distinguish among four groups divided by heterogeneity versus homogeneity and by communication versus lack thereof. Last, Weiß [285, 287] and Huhns and Singh [123] characterize multi-agent systems based on types of environments, agents, and inter-agent interactions.

1.2 Multi-Agent Learning

Much of the multi-agent learning literature has sprung from historically somewhat separate communities — notably reinforcement learning and dynamic programming, robotics, evolutionary computation, and complex systems. Existing surveys of the work have likewise tended to define multi-agent learning is ways special to these communities. Instead, we begin this survey by defining multi-agent learning broadly: it is the application of machine learning to problems involving multiple agents.

We think that there are two features of multi-agent learning which merit its study as a field separate from ordinary machine learning. First, because multi-agent learning deals with problem domains involving multiple agents, the search space involved can be unusually large; and due to the interaction of those agents, small changes in learned behaviors can often result in unpredictable changes in the resulting macro-level ("emergent") properties of the multi-agent group as a whole. Second, multi-agent learning *may* involve *multiple learners*, each learning and adapting in the context of others; this introduces game-theoretic issues to the learning process which are not yet fully understood.

Except for section 3.2.3, the survey primarily covers topics of interest to *cooperative* multi-agent learning. While one might define cooperative domains restrictively in terms of joint shared reward, much of the literature tends to use a much broader notion of cooperation, particularly with the introduction of credit assignment (discussed later). As such, we feel that cooperative multi-agent learning should be loosely defined in terms of the *intent* of the experimenter. If the design of the problem and the learning system is constructed so as to (hopefully) encourage cooperation among the agents, then this is sufficient, even if in the end they fail to do so.

One large class of learning in multi-agent systems that we will ignore in this survey involves situations where a single agent learns while the other agents' behaviors are fixed. An example of such situations is presented in [96]. This is single-agent learning: there is only one learner, and the behaviors are plugged into only one agent, rather than distributed into multiple agents.

1.3 Machine Learning Methods

There are three main approaches to learning: *supervised*, *unsupervised*, and *reward-based* [1] learning. These methods are distinguished by what kind of feedback the critic provides to the learner. In supervised learning, the critic provides the correct output. In unsupervised learning, no feedback is provided at all. In reward-based learning, the critic provides a quality assessment (the "reward") of the learner's output.

Because of the inherent complexity in the interactions of multiple agents, various machine learning methods — notably supervised learning methods — are not easily applied to the problem because they typically assume a critic that can provide the agents with the "correct" behavior for a given situation (a notable exception involving teaching in the context of mutual supervised learners is presented in [83, 88, 302]). Thus the very large majority of papers in this field have used reward-based methods. The reward-based learning literature may be approximately divided into two subsets: *reinforcement learning* methods which estimate value functions; and *stochastic search* methods such as evolutionary computation, simulated annealing, and stochastic hill-climbing, which directly learn behaviors without appealing to value functions. In the stochastic search literature, most multi-agent discussion concentrates on evolutionary computation. The similarities between these two classes of learning mechanisms permit a rich infusion of ideas from one to the other: the bucket-brigade algorithm [115], the Samuel system [97], and the recent Stochastic Direct Reinforcement policy gradient algorithm [172].

Reinforcement Learning Reinforcement learning (RL) methods are particularly useful in domains where reinforcement[2] information (expressed as penalties or rewards) is provided after a sequence of actions performed in the environment. Q-Learning and Temporal-Difference (TD(λ)) Learning are two common RL methods; the former learns the utility of performing actions in states, while the latter usually learns the utility of being in the states themselves. Reinforcement learning methods are inspired by dynamic programming concepts and define formulas for updating the expected utilities and for using them for the exploration of the state space. The update is often a weighted sum of the current value, the reinforcement obtained when performing an action, and the expected utility of the next state reached after the action is performed. While exploring, deterministic strategies may choose the most promising action to be performed in each state (according to the expected utilities). Other stochastic techniques may lead to better exploration of the state space. Reinforcement learning methods have theoretical proofs of convergence; unfortunately, such convergence assumptions do not hold for some real-world applications, including many multi-agent systems problems. For more information on reinforcement learning techniques, [11, 135, 260] are good starting points.

Evolutionary Computation Evolutionary Computation (EC) (or *Evolutionary Algorithms* (EAs)) is a family of techniques in which abstract Darwinian models of evolution are applied to refine populations of candidate solutions (known as "individuals") to a given problem. An evolutionary algorithm begins with an initial population of randomly-generated individuals. Each member of this population is then evaluated and assigned a fitness (a quality assessment). The EA then uses a fitness-oriented procedure to select, breed, and mutate individuals to produce children which are then added to the population, replacing older individuals. One evaluation, selection, and breeding cycle is known as a *generation*. Successive generations continue to refine the population until time is exhausted or a sufficiently fit individual is discovered. Evolutionary computation methods include *genetic algorithms* (GA) and *evolution strategies* (ES), both of which are usually applied to the search of multidimensional parameter spaces; and *genetic programming* (GP), which concerns itself with evolving actual computer programs. There are many sources of additional information on EC; good choices include [6, 63, 64, 81, 87, 114, 141, 164].

Coevolutionary algorithms (CEAs) represent a natural approach to applying evolutionary computation to refine multi-agent behaviors. In a CEA, the fitness of an individual is based on its interaction with other individuals in the population: thus the fitness assessment is context-sensitive and subjective. In *competitive* coevolution, individuals benefit at the expense of their peers; but in *cooperative* coevolution, individuals succeed or fail together in collaboration.

[1] Some of the literature calls this "reinforcement learning". Confusingly, that term is used in two different ways: the general category of learning based on rewards or quality assessments as feedback; and a specific family of learning algorithms related to dynamic programming (including Q-learning, Sarsa, Temporal Difference Learning, etc.). To avoid confusion, we use "reward-based learning" for the former, and "reinforcement learning" for latter. The term "reward" is admittedly problematic though, as there exist "negative rewards" (punishments).

[2] The previous footnote notwithstanding, in this survey, we use the terms *reward* and *reinforcement* interchangeably to denote the information the agents receive from the environment as a consequence of their actions.

A standard approach to applying cooperative coevolutionary algorithms (or CCEAs) to an optimization problem starts by decomposing the problem representation into subcomponents, then assigning each subcomponent to a separate population of individuals [205, 206, 207, 208].

1.4 Survey Structure and Taxonomy

We believe there are two major categories of cooperative multi-agent learning approaches. The first one, which we term *team learning*, applies a single learner to search for behaviors for the entire team of agents. Such approaches are more along the lines of traditional machine learning methods, but they may have scalability problems as the number of agents is increased. A second category of techniques, *concurrent learning*, uses multiple concurrent learning processes. Rather than learning behaviors for the entire team, concurrent learning approaches typically employ a learner for each team member, in the hope that this reduces the joint space by projecting it into N separate spaces. However, the presence of multiple concurrent learners makes the environment non-stationary, which is a violation of the assumptions behind most traditional machine learning techniques. For this reason, concurrent learning requires new (or significantly modified versions of) machine learning methods.

To illustrate the difference between the two, consider again the cooperative foraging scenario, where the task is to maximize the number of items gathered from the environment. A team learning approach employs a single learner to iteratively improve the "team behavior" — which consists of *all* foragers — using the totality of items gathered as a performance measure. The concurrent learning approach allows each forager to modify its own behavior via the forager's *own* learning process. However, the performance measure must now be apportioned among the various foragers (for example, dividing the team reward equally among the team members, or based on individual merits — exactly how many items each forager gathered). The foraging agents will improve their behaviors independent of one another, but have little or no control over how the others decide to behave.

The survey continues in the next Section with a presentation of research in team learning. When a single learner is employed, most research has focused on the *representation* of candidate solutions that the learner is developing: in particular, the degree of heterogeneity among the team members. The section after this discusses concurrent learning. Research in concurrent learning has broken down along different lines than that in team learning, primarily because of the effect of multiple learners. We divide concurrent learning research into credit assignment, analysis of the dynamics of the learning process, and modeling other agents in order to better cooperate with them. In Section 4 we discuss inter-agent communication and how it may be applied to the multi-agent learning process. We follow this with a discussion in Section 5 of three areas in multi-agent learning (scalability, adaptive dynamics, and problem decomposition) which we feel present large open research questions that have so far been insufficiently addressed by the community. The remaining sections provide a list of multi-agent problem domains, a collection of available resources, and the paper's conclusion and suggestions for scaling up multi-agent learning problems.

2 Team Learning

In team learning, there is a single learner involved: but this learner is discovering a set of behaviors for a team of agents, rather than a single agent. This lacks the game-theoretic aspect of multiple learners, but still poses challenges because as agents interact with one another, the joint behavior can be unexpected. This notion is often dubbed the *emergent complexity* of the multi-agent system.

Team learning is an easy approach to multi-agent learning because its single learned can use standard single-agent machine learning techniques. This sidesteps the difficulties arising from the co-adaptation of several learners that we will later encounter in concurrent learning approaches. Another advantage of a single learner is that it is concerned with the performance of the entire team, and not with that of individual agents. For this reason, team learning approaches may (and usually do) ignore inter-agent credit assignment — discussed later — which is usually difficult to compute.

Team learning has some disadvantages as well. A major problem with team learning is the large state space for the learning process. For example, if agent A can be in any of 100 states and agent B can be in any of another 100 states, the team formed from the two agents can be in as many as $10,000$ states. This explosion in the state space size can be overwhelming for learning methods that explore the space of state utilities (such as reinforcement learning), but it may not so drastically affect techniques that explore the space of behaviors (such as evolutionary computation)

[127, 220, 221, 237]. A second disadvantage is the centralization of the learning algorithm: all resources need to be available in the single place where all computation is performed. This can be burdensome in domains where data is inherently distributed.

Team learning may be divided into two categories: *homogeneous* and *heterogeneous* team learning. Homogeneous learners develop a single agent behavior which is used by every agent on the team. Heterogeneous team learners can develop a unique behavior for each agent. Heterogeneous learners must cope with a larger search space, but hold the promise of better solutions through agent specialization. There exist approaches in the middle-ground between these two categories: for example, dividing the team into squads, with squadmates sharing the same behavior. We will refer to these as *hybrid* team learning methods.

Consider a possible application of a team learning approach to the cooperative foraging scenario described earlier. The single learning process is concerned with improving the foraging performance of the entire team of agents. As such, it needs to somehow encode how each agent will behave in any situation it may encounter. As there may be *many* agents, the encoding may occupy *a lot* of memory. Heterogeneous team learning approaches may thus demand a very small number of agents. Yet it may be helpful for some agents to act as scouts, while the other agents carry the items as fast as possible. Homogeneous learners cannot readily discover such spacialization, though a hybrid method could.

Choosing among these approaches depends on whether specialists are needed in the team or not. Experiments conducted by Balch [8], Bongard [21] and Potter et al. [209] address exactly this issue (though some of them use concurrent learning approaches). Balch [8] suggests that domains where single agents can perform well (for example, foraging) are particularly suited for homogeneous learning, while domains that require task specialization (such as robotic soccer) are more suitable for heterogeneous approaches. His argument is bolstered by Bongard [21], who hypothesizes that heterogeneity may be better in inherently decomposable domains. Potter et al. [209] suggest that domain difficulty is not a determinant factor in selecting a heterogeneous approach. They experiment with increasingly difficult versions of a multi-agent herding domain obtained by adding predators, and argue that increasing the number of different skills required to solve the domain is a determinant factor.

2.1 Homogeneous Team Learning

In homogeneous team learning, all agents are assigned identical behaviors, even though the agents may not be identical (for example, different agents may take a different amount of time to complete the same task). Because all agents have the same behavior, the search space for the learning process is drastically reduced. The appropriateness of homogeneous learning depends on the problem: some problems do not require agent specialization to achieve good performance. For other problem domains, particularly ones with very large numbers of agents ("swarms"), the search space is simply too large to use heterogeneous learning, even if heterogeneity would ultimately yield the best results. Many homogeneous team learning papers are concerned with communication issues; we discuss such literature in Section 4.

In a straightforward example of successful homogeneous team learning, Haynes et al. [103], Haynes and Sen [104], Haynes et al. [109, 110, 111] evolved behaviors for a predator-prey pursuit domain. When using fixed algorithms for the prey behavior, the authors report results competitive with the best human-coded greedy predator algorithms: but when competitively coevolving the prey and predator behaviors (see Section 3.2.3), the learner discovered a prey that evades all previously reported hand-coded, greedy, and evolved predators.

Agents can act heterogeneously even in homogeneous team learning, if the homogeneous behavior specifies sub-behaviors that differ based an agent's initial condition (location etc.) or its relationship with other agents ("always follow agent 2"). For example, Quinn et al. [213] investigate the use of evolutionary computation techniques for a team formation problem. Three agents start from random positions but close enough to sense one another. They are required to move the team centroid a specific distance while avoiding collisions and remaining within sensor range. Quinn et al. investigate the roles of team members by removing the agents one at a time. They conclude that the rear agent is essential to sustain locomotion, but it is not essential to the other two agents' ability to maintain formation. The middle agent is needed to keep the two others within sensor range, and the front agent is crucial to team formation. Therefore, even though the agents are homogeneous, they specialize (based on their relative positions) to perform better as a team.

Salustowicz et al. [220, 221] compare different machine learning methods applied to homogeneous learning (the

EC-related PIPE and CO-PIPE algorithms versus Q-learning with all agents using the same Q-table). They argue that Q-learning has serious problems, attributed by the authors to the algorithm's need to search for a value function. On the other hand, both PIPE and CO-PIPE search directly in the policy space and show good performance. A contradicting result is reported in [301], where a variation of Q-learning outperforms the methods earlier reported in [220, 221].

A cellular automaton (CA) is an oft-overlooked paradigm for homogeneous team learning (only a few CA papers examine heterogeneous agents). A CA consists of a neighborhood (often a row or grid) of agents, each with its own internal state, plus a state-update agent behavior (the *rule*) applied to all the agents synchronously. This rule is usually based on the current states of an agent's neighbors. CAs have many of the hallmarks of a multi-agent system: interactions and communications are local, and behaviors are performed independently. A good survey of existing work in learning cellular automata rules is presented in [167].

One common CA problem, the Majority (or Density) Classification task, asks for an update rule which — given initial configurations of agents, each agent with an internal state of 1 or 0 — correctly classifies as many of them as possible based on whether the initial configuration had more 1's than 0's or more 0's than 1's. This is done by repeatedly applying the update rule for some N iterations; if the agents have converged to all 1's, the rule is said to have classified the initial configuration as majority-1's (similarly for 0's). If it has not converged, the rule has not classified the initial configuration. The goal is to discover a rule which classifies most configurations correctly given specific standard settings (in terms of number of agents, size of neighborhood, etc). Due to the complexity of the emergent behavior, it is exceptionally difficult to create good performing solutions for this problem by hand. The best human-coded result, of 82.178% accuracy, was proposed by Das et al. [60]. Much better results have been produced with evolutionary computation methods: Andre et al. [2] report a rule with accuracy 82.326% using genetic programming. Several authors, including Juille and Pollack [134], Pagie and Mitchell [186], Werfel et al. [291], suggest that coevolution[3] might perform well in this problem domain; at present the best known result, with accuracy 86.3%, was obtained via coevolution [134].

2.2 Heterogeneous Team Learning

In heterogeneous team learning, the team is composed of agents with different behaviors, with a single learner trying to improve the team as a whole. This approach allows for more diversity in the team at the cost of increasing the search space [89]. The bulk of research in heterogeneous team learning has concerned itself with the requirement for or the emergence of specialists.

Luke and Spector [153] investigate possible alternatives for evolving teams of agents. In their experiments, Luke and Spector compare homogeneity, heterogeneity with restricted breeding (agents could only breed with like agents from other teams), and heterogeneity with no breeding restrictions. The authors suggest that the restricted breeding works better than having no restrictions for heterogeneous teams, which may imply that the specialization allowed by the heterogeneous team representation conflicts with the inter-agent genotype mixture allowed by the free interbreeding. Similarly, Andre and Teller [3] apply genetic programming to develop a team of soccer playing agents for the RoboCup simulator. The individuals encode eleven different behaviors (one for each player). Andre and Teller mention that the crossover operator (between teams of agents) was most successful when performing restricted breeding.

Haynes and Sen [106, 108, 112] investigate the evolution of homogeneous and heterogeneous teams for the predator-prey pursuit domain. The authors present several crossover operators that may promote the appearance of specialists within the teams. The results indicate that team heterogeneity can significantly help despite apparent domain homogeneity. The authors suggest that this may be due to deadlocks generated by identical behaviors of homogeneous agents when positioned in the same location. Haynes and Sen report that the most successful crossover operator allows arbitrary crossover operations between behaviors for different agents, a result contradicting Andre and Teller [3] and Luke and Spector [153].

So-called "one-population" coevolution can be used with heterogeneous team learning in an unusual way: a single population is evolved using an ordinary evolutionary algorithm, but agents are tested by teaming them up with partners

[3]The papers refer to a particular case of coevolution, namely competitive coevolution, and a special setting containing two subpopulations. The first subpopulation consists of candidate solutions for the problem (in this case, behaviors for the CA agents). The second subpopulation contains possible test cases — initial configurations that the candidate solutions should solve. Candidate solutions in the first subpopulation are considered better fit when solving more test cases existing in the second subpopulation; meanwhile, the fitness of a test case is based on how many candidate solutions it stumps.

chosen at random from the same population, to form a heterogeneous team [41, 211]. Quinn shows a situation where this method outperforms a homogeneous team learning approach. Similarly, Miconi [165] evaluates an individual by forming a team from the population, then comparing how much worse the team performs when the individual is not in it. Miconi reports the appearance of squads and "subspecies" when using this method. It is not clear, however, where one-population cooperative coevolution should be categorized. It is clearly a single learner (one EC population), yet the team is formed from separately-learned teammates, and the method exhibits many of the game-theoretic oddities discussed in the Concurrent Learning Sections, particularly Section 3.2.3.

2.3 Hybrid Team Learning

In hybrid team learning, the set of agents is split into several squads, with each agent belonging to only one squad. All agents in a squad have the same behavior. One extreme (a single squad), is equivalent to homogeneous team learning, while the other extreme (one agent per squad) is equivalent to heterogeneous team learning. Hybrid team learning thus permits the experimenter to achieve some of the advantages of each method.

Luke [151], Luke et al. [152] focus on evolving soccer teams for the RoboCup competition, and they mention that the limited amount of time available before the competition diminished the probability of obtaining good heterogeneous teams. Instead, they compare the fully homogeneous results with a hybrid combination that divides the team into six squads of one or two agents each, and then evolves six behaviors, one per squad. The authors report that homogeneous teams performed better than the hybrid approach, but mention that the latter exhibited initial offensive-defensive squad specialization and suggest that hybrid teams might have outperformed the homogeneous ones given more time.

Hara and Nagao [101] present an innovative method for hybrid group learning. Faced with the specialization/search-space trade-off inherent in heterogeneity, the authors suggest an automated grouping technique called Automatically Defined Groups (ADG). They apply it successfully to a simple load transportation problem and to a modified tile-world domain. In ADG, the team of agents is composed of several groups of homogeneous agents (similar to [151, 152]); however, ADG automatically discovers the optimum number of groups and their compositions. A similar approach is reported in [21], where GP individuals contain partitioning instructions used for the creation of squads.

3 Concurrent Learning

The most common alternative to team learning in cooperative multi-agent systems is *concurrent learning*, where multiple learning processes attempt to improve parts of the team. Typically each agent has it own unique learning process to modify its behavior. There are other degrees of granularity of course: the team may be divided into "squads", each with its own learner, for example. However, we are not aware of any concurrent learning literature which assigns learners to anything but individual agents.

Concurrent learning and team learning each have their champions and detractors. Bull and Fogarty [40] and Iba [125, 126] present experiments where concurrent learning outperforms both homogeneous and heterogeneous team learning, while Miconi [166] reports that team learning is preferable in certain conditions. When then would each method be preferred over the other? Jansen and Wiegand [130] argue that concurrent learning may be preferable in those domains for which some decomposition is possible and helpful, and when it is useful to focus on each subproblem to some degree independently of the others. The reason for this is that concurrent learning projects the large joint team search space onto separate, smaller individual search spaces. If the problem can be decomposed such that individual agent behaviors are relatively disjoint, then this can result in a dramatic reduction in search space and in computational complexity. A second, related advantage is that breaking the learning process into smaller chunks permits more flexibility in the use of computational resources to learn each process because they may, at least partly, be learned independently of one another.

The central challenge for concurrent learning is that each learner is adapting its behaviors in the context of other co-adapting learners over which it has no control. In single-agent scenarios (where standard machine learning techniques may be applicable), a learner explores its environment, and while doing so, improves its behavior. Things change with multiple learners: as the agents learn, they modify their behaviors, which in turn can ruin other agents' learned

behaviors by making obsolete the assumptions on which they are based [223, 282]. One simplistic approach to deal with this co-adaptation is to treat the other learners as part of a dynamic environment to which the given learner must adapt. This idea was used in early multi-agent learning literature [228, 229, 312]. But things are more complicated in concurrent learning: the other agents are not merely changing, but are in fact co-adapting to the original learner's adaptation *to them*. Therefore, the agents' adaptation to the environment can change the environment itself in a way that makes that very adaptation invalid. This is a significant violation of the basic assumptions behind most traditional machine learning techniques.

Concurrent learning literature breaks down along different lines than team learning literature. Since each agent is free to learn separately, heterogeneity versus homogeneity has been considered an emergent aspect rather than a design decision in concurrent learning (for example, Balch [7, 9] argues that the more local a reinforcement signal, the more homogeneous the final team). We argue that there are instead three main thrusts of research in the area of concurrent learning. First, there is the *credit assignment* problem, which deals with how to apportion the reward obtained at a team level to the individual learners. Second, there are challenges in the *dynamics of learning*. Such research aims to understand the impact of co-adaptation on the learning processes. Third, some work has been done on *modeling other agents* in order to improve the interactions (and collaboration) with them.

3.1 Credit Assignment

When dealing with multiple learners, one is faced with the task of divvying up among them the reward received through their joint actions. The simplest solution is to split the team reward equally among each of the learners, or in a larger sense, divide the reward such that whenever a learner's reward increases (or decreases), *all* learners' rewards increase (decrease). This credit assignment approach is usually termed *global reward*.

There are many situations where it might be desirable to assign credit in a different fashion, however. Clearly if certain learners' agents did the lion's share of the task, it might be helpful to specially reward those learners for their actions, or to punish others for laziness. Similarly, Wolpert and Tumer [304] argue that global reward does not scale well to increasingly difficult problems because the learners do not have sufficient feedback tailored to their own specific actions. In other situations credit assignment *must* be done differently because global reward cannot be efficiently computed, particularly in distributed computation environments. For example, in a robotics foraging domain, it may not be easy to globally gather the information about all items discovered and foraged.

If we're not to equally divide the team reward among the agents, what options are there, and how do they impact on learning? One extreme is to assess each agent's performance based solely on its individual behavior. This approach discourages laziness because it rewards agents only for those tasks they have actually accomplished. However, agents do not have any rational incentive to help other agents, and greedy behaviors may develop. We call this approach *local reward*.

Balch [7, 9] experiments with different credit assignment policies to explain when one works better than the others. He argues that local reward leads to faster learning rates, but not necessarily to better results than global reward. For one problem (foraging), local reward produces better results, while in another (soccer) global reward is better. The author suggests that using local reward increases the homogeneity of the learned teams. This in turn suggests that the choice of credit assignment strategy should depend on the desired degree of specialization.

Mataric [159] argues that agents' separate learning processes can be improved by combining individual local reinforcement with types of *social reinforcement*. One such reward, *observational reinforcement*, is obtained by observing other agents and imitating their behaviors; this may improve the overall team behavior by reproducing rare behaviors. Agents additionally receive small *vicarious reinforcements* whenever other agents are directly rewarded; the purpose of this type of social reinforcement is to spread individual rewards to other agents, and thus balance between local and global rewards. Mataric shows that a weighted combination of these components, together with local reward, has better results in a foraging application.

Chang et al. [49] take a different approach to credit assignment: each agent assumes the observed reward signal is a sum of the agent's direct contribution and some random Markov process that estimates the contributions of teammates. The agent may therefore employ a Kalman filter to separate the two terms and compute the agent's true contribution to the global reward. The authors show that reward filtering provides a better feedback for learning in simple cooperative multi-agent domains. Rather than apportion rewards to an agent based on its contribution to the team, one might instead apportion reward based on how the team would have fared had the agent never existed. Tumer et al. [270], Wolpert

and Tumer [304] call this the *Wonderful Life Utility*, and argue that it is better than both local and global reward, particularly when scaling to large numbers of agents.

Tangamchit et al. [266] take a different tack to credit assignment, arguing that averaging rewards over sequences of tasks is better than discounting rewards (making immediate rewards more important than future ones) in many multi-agent scenarios. The authors set up an experiment involving two robots learning to work together: one robot ideally handing off items to the other, which in turn carries them to a final destination. Discounting rewards results in the first robot receiving significantly less reward the the second one. The authors show that collaboration is achieved only when robots are rewarded based on a non-discounted global reward averaged over time.

The wide variety of credit assignment methods have a significant impact on our coverage of research in the dynamics of learning, which follows in the next section. Our initial focus will be on the study of concurrent learning processes in fully cooperative scenarios, where global reward is used. But as just mentioned, global reward may not always work best for concurrent learning agents: instead, various other schemes have been proposed. However these credit assignment schemes may run counter the researchers' intention for the agents to cooperate, resulting in dynamics resembling general-sum or even competitive games, which we also discuss in the next section.

To understand why credit assignment policies can complicate the dynamics of learning, suppose two foraging agents are rewarded solely based on the speed at which they are gathering items. A narrow bridge separates the source of items from the location where items are to be deposited. If both robots arrive at the bridge at the same time, what is the rational thing for them to do? If they have a local credit assignment procedure, neither is willing to let the other go first: that would cost time and would decrease an agent's own credit. This is reminiscent of social scientists' notion of being in the individual interest of none, but in the collective interest of all [146].

3.2 The Dynamics of Learning

When applying single-agent learning to stationary environments, the agent experiments with different behaviors until hopefully discovering a globally optimal behavior. In dynamic environments, the agent may at best try to keep up with the changes in the environment and constantly track the shifting optimal behavior. Things are even more complicated in multi-agent systems, where the agents may adaptively change each others' learning environments.

The range of tools to model and analyze the dynamics of concurrent learners is unfortunately very limited. Many such tools are particularly suited to only certain learning methods, and only a few offer a common framework for multiple learning techniques. Evolutionary game theory is the main tool in this second category, and it was successfully used to study the properties of cooperative coevolution [78, 296], to visualize basins of attraction to Nash equilibria for cooperative coevolution [191], and to study trajectories of concurrent Q-learning processes [271, 263]. Another tool for modeling and predicting the dynamics of concurrent multi-agent learners was recently proposed in Vidal and Durfee [276, 277]. The system uses parameters such as rate of behavior change per agent, learning and retention rates, and the rate at which other agents are learning. Combining this information permits approximating the error in the agents' decision function during the learning process.

Many studies in concurrent learning have investigated the problem from a game-theoretic perspective (for example, [82]). An important concept in such investigations is that of a Nash equilibrium, which is a joint strategy (one strategy for each agent) such that no single agent has any rational incentive (in terms of better reward) to change its strategy away from the equilibrium. As the learners do not usually have control over each others' behaviors, creating alliances to escape this equilibrium is not trivial. For this reason, many concurrent learning methods will converge to Nash equilibria, even if such equilibria correspond to suboptimal team behaviors.

In a fully-cooperative scenario with only global reward (Section 3.2.1), the rewards received by agents are correlated, and so increasing one's reward implies increasing everybody else's reward. In such cases, it is relatively straightforward to check that the concurrent learning approach has converged to the globally optimal Nash equilibrium. Section 3.2.2 covers learning in more general settings where the relationship among rewards received by learners is less clear. Such research is particularly apropos to learning in combination with non-global credit assignment schemes. The extreme of such environments includes competing scenarios, where rewards for learners are inversely correlated — this is the focus of Section 3.2.3.

3.2.1 Fully Cooperative Scenarios

A "fully cooperative" scenario employs a global reward scheme to divvy reinforcement equally among all agents. In such environments, proof of convergence to global Nash equilibria is relatively straightforward, and there has been a significant amount of research in the area. The research focuses on repeated and stochastic games, described below.

A repeated game consists of a series of interactions among two or more agents. After each interaction, each agent may receive some reward (or punishment). Reinforcements for interactions are independent of any previous interactions. Claus and Boutilier [51] propose two benchmark games (*climb* and *penalty*) and show that convergence to global optimum is not always achieved in these games even if each agent can immediately perceive the actions of all other agents in the environment. This result is disturbing, given the fact that usually agents do not have such *complete information* about the team and the environment.

Lauer and Riedmiller [142] suggest updating an agent's policy (Q-values) by estimating the likely best cooperation possible for the agent's action, and prove that this will converge to the optimum in deterministic repeated games. Kapetanakis and Kudenko [136, 137] point out possible flaws in Lauer's approach when dealing with stochastic environments, and present a modified exploration strategy that improves cooperation under these new conditions. Brafman and Tennenholtz [31] introduce a stochastic sampling technique that is guaranteed to converge to optimal Nash equilibria. The algorithm is polynomial in the number of actions of the agents, but it assumes *a priori* coordination of the agents' learning processes: the agents agree to a joint exploration phase of some length, then agree to a joint exploitation phase (where each agent settles on the behavior that yielded maximum reward).

Stochastic games are an extension of repeated games where at any point in time the game is in some *state*. The game transitions to a new state based on a stochastic function of the old state and the interactions among the agents in the old state. Reinforcements are based both on the interactions of the agents and on the current state value. With a single state, a stochastic game reduces to a repeated game; with a single agent, a stochastic game reduces to a Markov decision process. Most learning approaches in stochastic games target general-sum stochastic games; they will be presented in the next section. In contrast, Wang and Sandholm [279] present the *Optimal Adaptive Learning* algorithm, which is guaranteed to converge to optimal Nash equilibria if there are a finite number of actions and states. The algorithm creates "virtual games" for each state in order to eliminate suboptimal Nash equilibria. This is to our knowledge the only algorithm guaranteed to find the global optimum in fully cooperative stochastic games. Unfortunately, the optimality guarantees comes at a cost in scalability: the number of virtual games that need to be solved is exponential in the number of agents.

Partially observable Markov decision processes (POMDPs) extend stochastic games by adding constraints on the agents' ability to observe the state of the environment. The task of finding the optimal policies in POMDPs is PSPACE-complete [193], it becomes NEXP-complete for decentralized POMDPs [18]. Peshkin et al. [200] investigate a distributed reinforcement learning approach to partially observable fully cooperative stochastic games, and show that their algorithm converges to local optima that may not necessarily be Nash equilibria. Another approach involves agents adapting one at a time while keeping fixed the policies of all other agents; in combination with dynamic programming, this can lead to exponential speedups [180].

Cooperative coevolutionary algorithms may be used to learn solutions to repeated and stochastic games. CCEAs were originally proposed as ways to (possibly automatically) decompose problems and to concurrently search for solutions to such subproblems [205, 206]. Some work has also been done in tuning parameters of the system in order to increase the performance of the algorithm [38, 39, 297]. Recent work has analyzed the conditions under which coevolutionary systems gravitate towards Nash equilibria rather than providing globally optimal solutions for the team as a whole [296, 298, 299]. Panait et al. [190, 191, 192] point out that such standard coevolutionary approaches, when applied to cooperative problem domains, are sometimes driven by *balance* considerations rather than performance. That is, agents tend to co-adapt to one another (including very poor collaborators) rather than trying to optimize their performance with the rational ideal collaborating agents. Panait *et al* show that evaluating an agent in the context of agents chosen as estimates of its best possible collaborators yields significantly improved results over the standard coevolutionary approach.

But choosing such estimates is non-trivial. One way to attack the problem is to test individuals in the context of well-known "good partners" from previous experience. In [90, 210], a coevolutionary algorithm is augmented with a "hall of fame" repository consisting of the N best teams discovered so far. Individuals are evaluated by pairing them up with teammates chosen from this hall of fame; this approach generally outperforms ordinary cooperative coevolution

methods. A related approach is reported in [19], where learners periodically provide representative behaviors for the other learners to use for training.

3.2.2 General Sum Games

Due to unequal-share credit assignment, increasing the reward of an agent may not necessarily result in increasing the reward of all its teammates. Indeed, such credit assignment can inadvertently create highly non-cooperative scenarios. For such reasons, general sum games are applicable to the cooperative learning paradigm, even though in some situations such games may not be in any way cooperative. Following the early work of Littman [147], there has been significant recent research in concurrent (and not necessarily cooperative) learning for general-sum stochastic games. Some work has applied Q-learning to extensive-form games [268] where players alternate in performing actions; but most work in the area has instead been in normal-form games where players act simultaneously.

Bowling and Veloso [26] examine a number of game theory and reinforcement learning approaches to stochastic games, and describe the differences in the assumptions they make. Hu and Wellman [119] (revised and extended by Bowling [24], Hu and Wellman [121]) introduce a distributed reinforcement learning algorithm where agents do not learn just a single table of Q-values, but also tables for all other agents. This extra information is used later to approximate the actions of the other agents. Nagayuki et al. [178] present an alternative approach where agents approximate the policies, rather than the tables of Q-values, of the other agents. Suematsu and Hayashi [258] point out that the algorithm of Hu and Wellman does not adapt to the other agent's policy, but only tries to converge to Nash equilibria. In the case when the other agent has a fixed policy, reaching a Nash equilibrium may be impossible. They then introduce the EXORL algorithm which learns optimal response policies to both adaptive and fixed policies of the other agent. Littman [148] introduces the Friend-or-Foe Q-learning algorithm to find either a coordination equilibrium (with cooperative agents) or an adversarial one (with competitive agents). Greenwald and Hall [95] introduce Correlated-Q — a learning algorithm based on the "correlated equilibrium" solution concept (which assumes additional coordination among agents via a mechanism to specify what action each agent should play).

Bowling and Veloso [27] describe two desirable properties for learning agents, namely rationality (the agent should converge optimally when the other agents have converged to stationary strategies) and convergence (under specified conditions, all agents should converge to stationary strategies). They then present a learning algorithm that exhibits these two properties. Bowling and Veloso [28] investigate the existence of equilibria points for agents with limitations which may prevent them from reaching optimality. Bowling and Veloso [29] also introduce the WoLF algorithm (Win or Learn Fast), which varies the learning rate from small and cautious values when winning, to large and aggressive values when losing to the other agents.

In [182], two agents with two actions each participate in a repeated game with two Nash equilibria. In one Nash equilibrium, one agent receives more reward than the other agent, while in the other Nash equilibrium the situation is reversed. The authors propose a "homo-equalis" reinforcement approach, where communication is used to alternate between the two unfair optimal points in order to fairly share the rewards received by the two agents. Peeters et al. [199] describe an extension of the algorithm for games with multiple states.

3.2.3 Related Pathologies from Competitive Learning

Though this survey focuses on cooperative multi-agent learning, many pathologies in competitive learning environments arise from similar dynamics, and so are worthwhile mentioning as hints into difficulties that may plague learning in cooperative scenarios as well. Competitive learning pits two or more agents against one another, hopefully establishing a beneficial "arms race" among them. For example, if two agents learn to play one another at chess, it may be desirable for both to learn at about the same pace, creating a gradual ramping up of difficulty. This has often been used in the context of a single learner playing a multi-agent game against itself [4]. The most famous examples of this are learned Backgammon [202, 203, 267] and Checkers [80, 222]. Other games have included Tic-Tac-Toe [4], Mancala [61], Othello [241], Go [149], Chess [138, 269], Poker [139], Blackjack [261], Nim and 3D Tic-TacToe [219], and Tag [215]. Some domains contain a mixture of competition and cooperation. For example, Luke [151] evolves a team of players for a soccer domain: the players need to cooperate as a team, but also compete against an opposing team.

Competitive learning can of course also be done with multiple learners, each co-adapting to the others. But the intended outcome is usually different. Instead of learning to compete against like foes, multi-learner scenarios usually

try to discover a solution which is robust over a wide range of problem test cases. Thus one target learner discovers the solutions, while a competing learner co-adaptively adjusts the set of test cases. This is sometimes known as "host-parasite" coevolution. Examples include sorting networks [113] and predator-prey scenarios [52, 102].

However, if unchecked, one learner can come to dominate the other learner. In this situation, no one receives sufficient feedback to improve — no matter what the dominant learner does, it always wins, and no matter what the subjugated learner(s) do, they always lose. This is usually termed *loss of gradient* [203, 280], and it is closely related to "laziness" and similar pathologies in cooperation. For example, Wiegand and Sarma [300] investigate the behavior of cooperative coevolution in a domain that provides different learning opportunities to different agents (one agent may learn faster than the other). The authors' spatially embedded CCEAs (which place constraints on the learners' interactions) may better cope with the loss of gradient created by these asymmetries.

It is also difficult to monitor progress in competitive coevolution. Consider again our chess players: they initially draw most of the games, while later on, the first agent defeats the other one. What does that tell us? We're tempted to say the first agent learned to play better. However, it may be that the first player has not changed at all, but rather that the second managed to decrease its playing ability. The first agent might in fact have gotten *worse* at chess, yet the second agent decreased in performance *even more*. This phenomenon is known as the *Red-Queen*[4] *effect* [52]. A related question concerns cooperative multi-agent learning when a local reward credit assignment is used: while each learner improves with respect to its own individual reward, are there any guarantees that the entire team of agents is improving as well?

Last, competition can result in *cyclic behaviors*, where agents circle about one another due to non-transitive relationships between agent interactions. For example, in the rock-scissors-paper game, Agent A picks Rock and Agent B picks Paper. Agent A then adapts to Scissors, causing Agent B to adapt to Rock. Then Agent A adapts to Paper, Agent B adapts to Scissors, and Agent A adapts *again* to Rock. If a multi-agent environment is fraught with such non-transitive cycles, competing agents may be caught in them, rather than building the desired arms race [52, 77, 219, 280]. Luke and Wiegand [155] cite (harsh) conditions under which single-learner coevolution can be rid of these difficulties. Popovici and DeJong [204] argue that criteria used to analyze single-agent domains may be insufficient for classifying, at a high-level, the dynamics such environments will incur.

3.3 Teammate Modeling

A final area of research in concurrent learning is teammate modeling: learning about other agents in the environment so as to make good guesses of their expected behavior, and to act accordingly (to cooperate with them more effectively, for example). Such an approach is used by Boutilier [22] and Chalkiadakis and Boutilier [47], who employ a Bayesian learning method for updating models of other agents. Based on these models, agents may estimate the current behavior of other agents and try to better cooperate with them. Suryadi and Gmytrasiewicz [259] present a similar agent modeling approach consisting of learning the beliefs, capabilities and preferences of teammates. As the correct model cannot usually be computed, the system stores a set of such models together with their probability of being correct, given the observed behaviors of the other agents.

As other agents are likely modeling *you*, modeling *them* in turn brings up the spectre of infinite recursion: "Agent A is doing X because it thinks that agent B thinks that agent A thinks that agent B thinks that ..." This must be rationally sorted out in finite time. Vidal and Durfee [275] categorize agents based on the complexity they assume for their teammates. A 0-level agent believes that none of its teammates is performing any learning activity and it does not consider their changing behaviors as "adaptive" in its model. A 1-level agent models its teammates as 0-level agents. In general, an N-level agent models its teammates as (N-1)-level agents.

Mundhe and Sen [176] investigate the use of 0-level, 1-level and 2-level modeling agents. The authors report a very good performance for the 0-level learners, suggesting that for some domains teammate modeling may not be necessary. Similar results showing good coordination without modeling other agents are reported in [238, 239], where two robots learn to cooperatively push a box without either being aware of the other's presence. Banerjee et al. [10], Mukherjee and Sen [173] present experiments in which 1-level agents model each others' action probability distributions; this produces a form of mutual trust. Tambe [264] briefly discusses the benefits of modeling other agents and recursively

[4]The name refers to Lewis Carroll's *Through the Looking Glass*, where the Red Queen constantly runs because "it takes all the running you can do to keep in the same place".

tracking their decisions (either for individual agents, or for groups of agents). Benefits from modeling agents in other purely competitive settings are also reported in [272, 45].

When dealing with larger numbers of teammates, a simpler modeling approach is to presume that the entire team consists of agents identical to the modeling agent. Haynes et al. [103], Haynes and Sen [105, 107] use this approach complemented with a case-based learning mechanism for dealing with exceptions from this assumed behavior.

Hu and Wellman [118], Wellman and Hu [290] suggest that when learning models of other agents in a multi-agent learning scenario, the resulting behaviors are highly sensitive to the agents' initial beliefs. Depending on these initial beliefs, the final performance may be better or even *worse* than when no teammate modeling is performed — agent modeling may prevent agents from converging to optimal behaviors. A similar conclusion is reported by Hu and Wellman [120]: the authors suggest that the best policy for creating learning agents is to *minimize* the assumptions about the other agents' policies. An alternative is to automatically discover if the other agents in the environment are cooperating with you, competing with you, or are in some other relationship to you. Sekaran and Sen [233], Sen and Sekaran [236] investigate the application of reciprocity concepts to multi-agent domains where not enough information on the intentions of other agents is known *a priori*. They apply a stochastic decision-making algorithm that encourages reciprocity among agents while avoiding being taken advantage of by unfriendly agents. The authors show that those agents that prefer not to cooperate end up with worse performance. Nowak and Sigmund [181] mention two types of reciprocity: *direct* (agent A helps another agent B and so expects B to help him in the future) and *indirect* (an agent helps another in return for future help from other agents). Such reciprocity improves an agent's "reputation". They argue that a necessary condition for cooperation is that the "degree of acquaintanceship" (the probability that an agent knows another agent's reputation) should exceed the ratio of cost of altruistic help relative to the benefit to the recipient. Similar analysis can be done without the need to model reputation *per se*: Ito [129] pitted game-players against one another, where each agent had access to a history of his opponent's previous actions against other players.

We conclude this section with work in agent modeling under communication. Ohko et al. [184] use a communication protocol for agents to subcontract subtasks to other agents. In their approach, each agent tries to decompose tasks into simpler subtasks and broadcasts announcements about the subtasks to all other agents in order to find "contractors" who can solve them more easily. When agents are capable of learning about other agents' task-solving abilities, communication is reduced from broadcasting to everyone to communicating exact messages to only those agents that have high probabilities to win the bids for those tasks. A related approach is presented in [37, 36]: here, Bayesian learning is used to incrementally update models of other agents to reduce communication load by anticipating their future actions based on their previous ones. Case-based learning has also been used to develop successful joint plans based on one's historical expectations of other agents' actions [83].

4 Learning and Communication

For some problems communication is a necessity; for others, communication may nonetheless increase agent performance. We define *communication* as altering the state of the environment such that other agents can perceive the modification and decode information from it. Among other reasons, agents communicate in order to coordinate more effectively, to distribute more accurate models of the environment, and to learn subtask solutions from one another.

But are communicating agents really *multi-agent?* Stone and Veloso [255] argue that unrestricted communication reduces a multi-agent system to something isomorphic to a single-agent system. They do this by noting that without restrictions, the agents can send complete external state information to a "central agent", and to execute its commands in lock-step, in essence acting as effectors for the central agent. A central agent is not even necessary: as long as agents can receive all the information they need to known about the current states of all the other agents, they can make independent decisions knowing exactly what the other agents will do, in essence enabling a "central controller" on-board each individual agent, picking the proper sub-action for the full joint action. Thus we feel that a true multi-agent problem necessitates restrictions on communication. At any rate, while full, unrestricted communication can orthogonalize the learning problem into a basic single-agent problem, such an approach requires very fast communication of large amounts of information. Real-time applications instead place considerable restrictions on communication, in terms of both throughput and latency. Unfortunately, learning is a challenging issue in itself, and difficulties associated with it often resulted in a simplified approach to communication, usually neglecting costs of communication with other agents. We feel further research needs to address the issue of using selective communication only when necessary.

Explicit communication can also significantly increase the learning method's search space, both by increasing the size of the external state available to the agent (it now knows state information communicated from other agents), and by increasing the agent's available choices (perhaps by adding a "communicate with agent *i*" action). As noted in [73], this increase in search space can hamper learning an optimal behavior by more than communication itself may help. Thus even when communication is required for optimal performance, for many applications the learning method must disregard communication, or hard-code it, in order to simplify the learning process. For example, when learning in a predator-prey pursuit domain, Luke and Spector [153] assume that predators can sense each other's position no matter what distance separates them, and Berenji and Vengerov [17] use a blackboard communication scheme to allow agents to know of other agents' locations.

4.1 Direct Communication

Many agent communication techniques employ, or assume, an external communication method by which agents may share information with one another. The method may be constrained in terms of throughput, latency, locality, agent class, etc. Examples of direct communication include shared blackboards, signaling, and message-passing. The literature has examined both hard-coded communication methods and learned communication methods, and their effects on cooperative learning overall.

Tan [265] suggests that cooperating learners can use communication in a variety of ways in order to improve team performance. For example, agents can inform others of their current state by sharing immediate sensor information. Another approach is for agents to share information about past experiences in the form of episodes (sequences of $\langle state, action, reward \rangle$ previously encountered by the agent) that others may not have experienced yet. Yet another alternative is for agents to share knowledge related to their current policies (for example, in the form of $\langle state, action, utility \rangle$ for cooperating reinforcement learning agents).

Some research, particularly in reinforcement learning, has simply presumed that the agents have access to a joint utility table or to a joint policy table to which each may contribute in turn, even though the agents are separate learners. For example, Berenji and Vengerov [16] investigate a cooperative learning setting where multiple agents employ communication to use and update the same policy. Berenji and Vengerov suggest that the simultaneous update of a central policy reduces early tendencies for convergence to suboptimal behaviors. We argue that this is an implicit hard-coded communication procedure: the learners are teaching each other learned information.

Much of the remaining research provides the agents with a communication channel but does not hard-code its purpose. In a simple situation, agents' vocabulary may consist of a single signal detectable by other agents [278]. In other work (for example [309]) mobile robots in a cooperative movement task are endowed with a fixed but undefined communication vocabulary. The robots learn to associate meanings with words in various trials. When circumstances change, the robots learn to adjust their communication language as appropriate. A similar approach is reported in [133] to learn a language for communication in a predator-prey domain. The authors use a blackboard communication scheme and present a rule for determining a pessimistic estimate on the minimum language size that should be used for multi-agent problems. Steels [246] reports the emergence of a spontaneous coherent lexicon that may adapt to cope with new meanings during the lifetime of the agents. Steels and Kaplan [251] continue this investigation to show that agents are able to create general words through collective agreement on their meanings and coverage. Similar approaches to evolving communication languages are presented in [42, 62, 225, 245, 247, 248, 249, 250, 302].

Many such methods provide an incentive to the agents to communicate (perhaps by sharing the resulting reward). However, Ackley and Littman [1] show that reward incentives are not necessary for the agents to communicate: instead, agents learn to communicate even when the agents receive no benefit from this action.

4.2 Indirect Communication

We define indirect communication methods as those which involve the *implicit* transfer of information from agent to agent through modification of the world environment. Examples of indirect communication include: leaving footsteps in snow, leaving a trail of bread crumbs in order to find one's way back home, and providing hints through the placement of objects in the environment (perhaps including the agent's body itself).

Much of the indirect communication literature has drawn inspiration from social insects' use of pheromones to mark trails or to recruit other agents for tasks [116]. Pheromones are chemical compounds whose presence and

concentration can be sensed by fellow insects [20], and like many other media for indirect communication, pheromones can last a long time in the environment, though they may diffuse or evaporate. In some sense, pheromone deposits may be viewed as a large blackboard or state-utility table shared by all the agents; but they are different in that pheromones can only be detected locally.

Several pheromone-based learning algorithms have been proposed for foraging problem domains. A series of reinforcement learning algorithms have adopted a fixed pheromone laying procedure, and use current pheromone amounts for additional sensor information while exploring the space or while updating the state-action utility estimates [143, 168, 169, 170, 171]. Evolutionary computation techniques have also been applied to learn exploration/exploitation strategies using pheromones deposited by hard-coded mechanisms. For example, Sauter et al. [226, 227] show how EC can be used to tune an agent policy in an application involving multiple digital pheromones. A similar idea applied to network routing is presented in [294].

Another research direction is concerned with studying whether agents can learn not only to use pheromone information but to deposit the pheromones in a rational manner. This question was first examined in AntFarm, a system that combines communication via pheromones and evolutionary computation [54, 53]. AntFarm ants use a single pheromone to mark trails toward food sources, but use a compass to point themselves along the shortest path back to the nest. Panait and Luke [187, 188, 189] present a related algorithm that exhibits good performance at various foraging tasks in the presence of obstacles. This algorithm uses multiple pheromones, and in doing so it eliminates the explicit requirement for ants to precisely know the direction towards the nest from any point. Rather, ants use pheromone information to guide themselves to the food source and back to the nest. The authors note a hill-climbing effect that leads to straight (and locally optimal) paths, and demonstrate both hard-coded and learned foraging behaviors which build these paths.

Werger and Mataric [292] present another approach to indirect communication among agents: using the agents' body positions themselves to convey meaning. The authors present a foraging application where a number of robots need to collect items and deposit them at a pre-specified location. The robots use their bodies to mark the path for other robots to follow. In some sense, the robots' bodies are acting essentially as a pheromone trail for the other robots. Quinn [212] argues that this is a form of communication by citing Wilson's statement that communication "is neither the signal by itself, nor the response, [but] instead the relationship between the two" [303]. This definition suggests that a shared dedicated channel is not necessary for communication. In his study, Quinn investigates the evolution of strategies in a two-agent collaborative-movement domain and reports that robots were able to coordinate by communicating their adopted roles of leader or follower via sequences of moves. For example, after initial phases of alignment, the robots use rotation and oscillatory back-and-forth movements to decide who leads the way and who follows.

5 Major Open Topics

Multi-agent learning is a new field and as such its open research issues are still very much in flux. Here, we single out three issues we observed recurring while surveying past literature. We believe that these specific areas have proven themselves important open questions to tackle in order to make multi-agent learning more broadly successful as a technique. These issues arise from the *multi* in multi-agent learning, and may eventually require new learning methods special to multiple agents, as opposed to the more conventional single-agent learning methods (case-based learning, reinforcement learning, traditional evolutionary computation) now common in the field.

5.1 Scalability

Scalability is a problem for many learning techniques, but especially so for multi-agent learning. The dimensionality of the search space grows rapidly with the number and complexity of agent behaviors, the number of agents involved, and the size of the network of interactions between them. This search space grows so rapidly that we believe one *cannot* learn the entire joint behavior of a large, heterogeneous, strongly intercommunicating multi-agent system. Effective learning in an area this complex requires some degree of sacrifice: either by isolating the learned behaviors among individual agents, by reducing the heterogeneity of the agents, or by reducing the complexity of the agent's capabilities. Techniques such as learning hybrid teams, decomposing behaviors, or partially restricting the locality of

reinforcement provide promising solutions in this direction, but it is not well understood under which constraints and for which problem domains these restricted methods will work best.

We believe multi-agent learning techniques should be evaluated especially carefully with respect to their scalability. For example, it is not unusual (on the contrary, it is widely practiced) for concurrent learning techniques to be studied in two-agent scenarios, for the environment to be stateless (no history of past interactions is necessary), and for each agent to have only two or three actions to choose from. But when scaled up to dozens or hundreds of agents, to stochastic environments with partially-observable states, and to many more available actions, current methods seem likely to fail. This is of course not just a cooperative learning problem. Recent research by Sturtevant and Korf [256] in the area of search in competitive games has shown that mini-max and alpha-beta pruning are ineffective at significantly reducing the search for games with more than two players.

These simple scenarios have also presumed so trivial an environment that it is straightforward to identify the resulting outcome of the interactions of specific agents and their behaviors. But as problem complexity increases, it gives rise to the spectre of *emergent behavior*, where the global effects of simple per-agent behaviors cannot be readily predicted. This is an area of considerable study and excitement in the artificial life community: but we view it as a major problem for machine learning. To what degree does emergent behavior prevent the solution space from being smooth? If small perturbations in agent behavior result in radical swings in emergent behavior, can learning methods be expected to scale well at all in this environment? These questions have not been well studied.

5.2 Adaptive Dynamics and Nash Equilibria

Multi-agent systems are typically dynamic environments, with multiple learning agents vying for resources and tasks. This dynamism presents a unique challenge not normally found in single-agent learning: as the agents learn, their adaptation to one another changes the world scenario. How do agents learn in an environment where the goalposts are constantly and adaptively being moved? As mentioned before, this co-adaptation of learners to one another leads to a violation of a fundamental assumption of most machine learning techniques; for this reason, entirely new multi-agent learning algorithms may be required to deal with this issue. This is exacerbated by credit assignment schemes, which while often important, can convert an ordinary cooperative scenario into a general-sum or even (inadvertently) competitive environment.

This constant "moving the goalposts" presents a challenge to finding optimal behaviors. In many cases the learning methods may converge not to optima but to suboptimal Nash equilibria, if they converge at all. While avoiding convergence to suboptimal Nash equilibria has become a focus in coevolution [192, 296], we are concerned that much of the concurrent learning literature has merely satisfied itself to demonstrate convergence to Nash equilibria rather than to optima. A notable exception worth following is presented by Chalkiadakis and Boutilier [47], where costs are associated with exploration to balance its potential risks and benefits.

We echo opinions from [146, 240] and express our concern with the use of Nash equilibria in cooperative multi-agent learning. Such "rational" convergence to equilibria may well be movement away from globally *team-optimal* solutions [146]. As mentioned earlier, optimism about partners seems better than rationality alone for learners concerned with team optimality [51, 142, 136, 192]. Shoham et al. [240] also point out that learning techniques that specifically target Nash equilibria have problems coordinating when multiple optimal equilibria exist.

We argue that, in the context of cooperative agents, the requirement of rationality should be secondary to that of optimal team behavior. In the context of the Prisoner's Dilemma (Section 6.2) as a model for a *cooperative* game, it is clearly irrational for agents to choose lower payoffs just because they are afraid of being stumped by other agents. We believe that rational agents may learn better if they trust that their teammates share a common goal of achieving better team behavior.

5.3 Problem Decomposition

The state space of a large, joint multi-agent task can be overwhelming. An obvious way to tackle this is to use domain knowledge to simplify the state space, often by providing a smaller set of more "powerful" actions customized for the problem domain. For example, Mataric [162, 160] applies Q learning to select from hand-coded reactive behaviors such as *avoid, head-home, search* or *disperse* for robot foraging tasks. An alternative has been to reduce complexity by heuristically decomposing the problem, and hence the joint behavior, into separate, simpler behaviors for the agents

to learn. Such decomposition may be done at various levels (decomposing team behaviors into sub-behaviors for each agent; decomposing an agents' behavior into sub-behaviors; etc.), and the behaviors may be learned independently, iteratively (each depending on the earlier one), or in a bottom-up fashion (learning simple behaviors, then grouping into "complex" behaviors).

One approach to such decomposition is to learn basic behaviors first, then set them in stone and learn more complex behaviors based on them. This method is commonly known as *layered learning*, and was proposed by Stone [252, 253] and successfully applied to robotic soccer. Further applications of the technique are reported by Gustafson [99], Gustafson and Hsu [100], Hsu and Gustafson [117], Whiteson and Stone [295] in keep-away soccer. Another approach, *shaping*, gradually changes the reward function from favoring easier behaviors to favoring more complex ones based on those easy behaviors. Balch [9] uses a shaped reinforcement reward function (earlier suggested by Mataric [161]) which depends on the number of partial steps fulfilled towards accomplishing the joint task. Balch shows that using a shaped reward leads to similar results to using a local reward, but in a significantly shorter time. A related method, *fitness switching*, adaptively changes the reward function to emphasize those behaviors the agents have made the least progress [311].

Shaping, layered learning, and fitness switching, are not multi-agent learning techniques, but they have often been applied in such a context. Less work has been done on formal methods for decomposing tasks (and behaviors) into subtasks (sub-behaviors) appropriate for multi-agent solutions, how agents' sub-behaviors interact, and how and when the learning of these sub-behaviors may be parallelized. Consider robot soccer as an example: while it is true that agents must learn to acquire a ball and to kick it before they can learn to pass the ball, their counterparts must *also* have learned to receive the ball, and to ramp up difficulty, opponents may simultaneously co-adaptively learn to intercept the ball. Few papers have examined how to form these "decomposition dependency graphs", much less have the learning system develop them automatically. Yet to parallelize the learning process, to simplify the search space, and to produce more robust multi-agent behaviors, understanding these interactions is important. One notable exception: Guestrin et al. [98] note that in many domains the actions of some agents may be independent. Taking advantage of this, they suggest partially decomposing the joint Q-values of agents based on a *coordination graph* that heuristically spells out which agents must interact in order to solve the problem. This partial decomposition permits a heuristic middle-ground between learning a full joint utility table and learning separate independent tables. Also, Makar et al. [157], Ghavamzadeh and Mahadevan [84] suggest a different hierarchical approach to simplifying the inter-agent coordination task, where agents coordinate their high-level behaviors, rather than each primitive action they may perform.

6 Problem Domains and Applications

Despite the relative young age of the field, the multi-agent systems area contains a very large number of problem domains. The list in this survey is far from complete, but it contains many of the common problems. The problem domains are divided into three classes: embodied agents, game-theoretic environments, and applications to real-world problems.

6.1 Embodied Agents

The cost of robots has decreased significantly, making it feasible to purchase and use several (tens, hundreds, or even thousands of) robots for a variety of tasks. This drop in cost has spurred research in multi-agent cooperative robotics. Additionally, computer hardware is cheap enough that what cannot be performed with real robots can now be done in simulation; though the robotics community still strongly encourages validation of results on real robots.

Predator-Prey Pursuit This is one of the most common environments in multi-agent learning research, and it is easy to implement. Pursuit games consist of a number of agents (predators) cooperatively chasing a prey. Individual predator agents are usually not faster than the prey, and often agents can sense the prey only if it is close by. Therefore, the agents need to actively cooperate in order to successfully capture the prey. The earliest work using this domain is [15], but there are many variations of the problem [68].

Foraging The domain consists of a large map with agents (robots) and items to forage (pucks or cans). The task is to carry the items to specially designated areas. Variations may include multiple item locations, item types, and drop locations. The efficiency of an approach may be defined by how quickly it completes the foraging task [59, 158], or by the number of items collected in a fixed amount of time. Ostergaard et al. [185] provide an extended survey of previous work in this domain. A variation of the problem allows agents to communicate by depositing pheromones in the environment to mark trails connecting rich pick-up and drop-off areas [20, 189]. In a related problem domain, *clustering*, the agents have no pre-specified drop locations; rather, they only need to pile all the items together [14], or sort them by class into separate piles [67].

Box Pushing This domain involves a two-dimensional bounded area containing obstacles and a number of boxes. Agents in this environment need to arrange the boxes to specified final positions by pushing them. Sometimes a robot is capable of pushing one box by itself (see for example the single-agent experiments reported in [35, 156]). A more complicated version requires two or more agents to cooperate in order to move a single box in simulation [311], or on real robots [162, 163]. Buffet et al. [34], Dutech et al. [75] present reinforcement learning approaches for a similar problem domain where agents receive rewards only when they merge together pucks of different colors.

Soccer The game of soccer, both in simulation and with real robots, is one of the most widely used domains for research in multi-agent systems [140]. The domain consists of a soccer field with two goals, a ball, and two teams with a number of agents (from 5 to 11, depending on the sizes of the robots and of the field). The performance of a team is usually assessed based on the difference in number of goals scored. Other performance metrics have included length of time in control of the ball, successful interceptions, and average location of the ball. Successful applications of learning techniques to this domain are reported in [8, 152, 217, 252, 255]. The strong interest in this domain has led to several annual "world cup" robot soccer championships. The most well-known such competition, RoboCup, has different leagues divided by continent and by type of robot or simulation environment.

Keep-Away Soccer Gustafson and Hsu [100] use a simpler version of the soccer domain which contains only one defensive and three offensive players and a ball. The defensive player is twice as fast than the offensive ones, and the ball, when passed, can move at twice the speed of the defensive player. The objective is to keep the ball away from the defensive player by moving and passing; there is a penalty each time the defensive player is within one grid unit away from the ball. A similar domain is presented in [254].

Cooperative Navigation This task, as described in [8], is to have a team of agents move across a field in minimal time without colliding with obstacles or other robots. Each agent selects from a number of predefined behaviors, and the performance is assessed based on the maximum time necessary for the agents to accomplish the task. The questions investigated by Balch include the benefits from formation participation and the impact of diversity on performance. The Opera Problem, discussed in [56], represents another challenge problem in cooperative navigation: a large number of agents located in a bounded space need to coordinate in order to quickly leave through a small fixed exit area. An application of learning to this problem is shown in [218].

Cooperative Target Observation Introduced by Parker [195], this domain involves a two dimensional bounded area in which a number of robots have to keep several moving targets under observation. Performance is based on the total number of targets within the "observable" distance to any team-member robot during the time period. Parker investigates an initial approach to learning behaviors for this domain and reports improvements over naive, random approaches, but also notes the superiority of the hand generated solutions. Additional research using this domain includes [76, 154, 197, 196]. A related problem domain is described in [13, 12, 308]: a team of flying agents needs to perform a surveillance task in a region of space. The agents coordinate to avoid collisions, and their reward is further increased when discovering and observing areas of high interest. Because of constraints placed on flying unmanned vehicles, the agents must each maintain a minimal speed to keep them in the air. Multi-agent patrolling is a non-stationary variant of this surveillance task: the agents are required to continuously explore the environment [224, 262].

Herding Schultz et al. [231] introduce a domain where a robot must herd another robot into a designated area. The second robot (the sheep) will avoid obstacles and the "shepherd" robot, but otherwise may either move randomly or try to avoid the herding area. The shepherd moves close to the sheep robot to encourage the sheep to move in the desired direction. Potter et al. [209] try a multi-shepherd version of the domain, where many faster and "stubborn" sheep require coordination from the several herding agents. Additionally, predators may exist in the environment and try to kill the sheep while avoiding the shepherds, thus complicating the shepherds' task. Another application of learning techniques to a herding domain is presented in [293].

6.2 Game-Theoretic Environments

Many multi-agent systems may be cast in game-theoretic terms; essentially as strategy games consisting of matrices of payoffs for each agent based on their joint actions. In addition to game-theoretic analysis of multi-agent systems, some common problem domains are also taken from game theory.

Coordination Games Various repeated games described in terms of joint reward matrices have been previously introduced in the literature to highlight specific issues associated with multi-agent learning. For example, Claus and Boutilier [51] introduce two simple 3x3 matrix games: a coordination game with two optima and high penalties for mis-coordination; and a second game with two Nash-equilibrium points, one of them corresponding to a suboptimal collaboration. These games are later used in [137, 142, 192] to investigate multi-agent reinforcement learning and evolutionary computation approaches.

Social Dilemmas These problems concern the individual decisions of several agents, all of which receive a joint reward [85]. The *Iterated Prisoners' Dilemma*, *Tragedy of the Commons*, *Braess Paradox* and *Santa Fe Bar* are examples of social dilemma games. The Iterated Prisoner's Dilemma involves two or more agents supposedly accused of a robbery; the agents have to choose between to actions: *confess* the crime or *deny* participation in it. The settings are such that it is rational for individual agents to deny, but it is in their collective interest for all to confess. In the Tragedy of the Commons, a number of agents share a resource of limited capacity. When the joint usage of the resource exceeds the capacity, the service deteriorates, and so do the rewards received by the agents. In the Braess Paradox problem, agents share two resources. The dilemma arises when agents must decide to start accessing the less utilized resource: if all agents decide to do so, it will become overwhelmed and rewards will drop. Further details on these problems, accompanied by a coevolutionary approach to learning solutions to them, can be found in [177]. In the Santa Fe Bar problem, a large number of agents individually must decide whether to go to a bar in Santa Fe. If too many or too few agents opt to go, their satisfaction is lower than when a reasonable number of them decide to go [5, 94, 306]. All these social dilemma problems have been used to model practical issues in real multi-agent problems, such as network routing for example. We believe a valuable source of inspiration for solutions to such problems is the area of social sciences. For example, Lichbach [146] analyzes a large number of social science solutions to the Iterated Prisoner's Dilemma problem.

6.3 Real-World Applications

This section describes a set of such problems drawn from real-world domains previously used in MAS investigations. Many of the described problem domains are logistics, planning, and constraint-satisfaction problems requiring real-time, distributed decision making. Because the applications are often very complex and highly distributed, learning techniques have rarely been applied to them, and so they are presented here primarily as example challenge problems for multi-agent learning.

Distributed Vehicle Monitoring The task in this domain is to maximize the throughput of cars through a grid of intersections. Each intersection is equipped with a traffic light controlled by an agent. The agents need to coordinate to deal with fluctuations in traffic [145, 183]. Dresner and Stone [70] additionally assign agents to cars and propose a reservation system for cars to negotiate for green lights upon arrival at the intersection.

Air Traffic Control For security purposes, the airspace is divided into three-dimensional regions used by air traffic controllers to guide the airplanes to their final destination. Each such region has an upper bound (called the *sector capacity*) on the number of airplanes it can hold at any time. The task is to guide the planes from sector to sector along minimal length routes, while ensuring that constraints are met; the solution needs to be fast to handle real-time data. A multi-agent approach for this domain is reported in [244].

Network Management and Routing This domain consists of a large, distributed network. Agents are deployed to distributively and cooperatively control and manage the network, handle failures, and balance its load [281]. Boyan and Littman [30], Subramanian et al. [257], Wolpert et al. [305], Chang et al. [50] introduce various learning approaches to route packets in static and ad-hoc networks.

Electricity Distribution Management Here the problem is to maintain an optimal power grid configuration that keeps all customers supplied and minimizes losses; while at the same time dealing with possible damage to the network, variable demand from customers, scheduled maintenance operations, and equipment failures and upgrades [274]. Schneider et al. [230] present a reinforcement learning approach to managing a power grid.

Distributed Medical Care This domain applies AI to assist clinical staff in making diagnoses, decide on therapy and tests, determine prescriptions, and perform other health care tasks [122]. The problem is well suited for multi-agent systems because of the decentralization of data and resources, the high costs for obtaining comprehensive information, and the stochasticity and dynamism of data.

Supply Chains This classic planning and scheduling problem involves managing the process of producing complex items through a series of steps, where there are different constraints and costs associated with each step. The task consists of building a plan (a *production sequence*) that specifies the order of operations for different items such that the production costs are minimized while satisfying customer orders [307]. Brauer and Weiß [32] present a learning approach for such a domain.

Hierarchical Multi-Agent Systems Problems Some multi-agent domains are of particular interest because of the different levels at which problems can be formulated. For example, in the "Transportation" problem, several trucking companies transport goods between locations. Depending on problem formulation, agents may represent whole companies, sets of trucks from the same or from different companies, or even individual trucks. The task is to complete requests from customers under specific constraints (maximum time required to finish the job, minimal cost of delivery, etc.). A multi-agent approach to this domain is reported in [79]. The "Loading Dock" is a similar problem, where several forklifts load and unload trucks according to task requirements; either individual forklifts or groups of forklifts may be modeled as an agent [175]. A related "Factory Floor" problem is investigated in [198], where products are manufactured from raw material by several machines under time and cost minimization constraints, and agents can represent individual machines or groups of machines.

Models of Social Interaction Many natural and social systems have very large numbers of interacting agents. Accordingly, such interactions need also be present in simulations of these natural phenomena. Examples of work in this area include hard-coded and learned collective behaviors such as flocking [216, 214, 242, 243], learning of dispersion and aggregation [310], learning social behaviors in virtual worlds [92, 91], and the modeling of the formation of early countries [46].

Meeting Scheduling In this domain, each person has an agent that knows his/her preferences (for example, no weekends) and obligations (possibly other meetings or travel). Upon request to schedule a meeting that requires the presence of multiple persons, their agents are solicited to find a time and a place to hold the meeting. Although most work in this area does not involve learning (for example, the Electric Elves system [48]), Crawford and Veloso [55] identify several opportunities for learning in the domain.

Other multi-agent systems domains investigated include particle accelerator control [132], intelligent document retrieval [174], spacecraft control [232], and concurrent engineering [58]. Further applications of Distributed AI in industry are reported in [273].

7 Resources

Conferences and Workshops Various AI and machine learning meetings welcome multi-agent topics. The largest such conference is the biannual International Joint Conference on Artificial Intelligence (IJCAI). Regional conferences include the annual conference of the American Association for Artificial Intelligence (AAAI) and its European equivalent, the European Conference on Artificial Intelligence (ECAI). Machine learning conferences include the International Conference on Machine Learning (ICML) and the European Conference on Machine Learning (ECML). Major evolutionary computation conferences include the Genetic and Evolutionary Computation Conference (GECCO), the Congress on Evolutionary Computation (CEC), and the European Conference on Parallel Problem-Solving from Nature (PPSN).

Robotics conferences of interest include the IEEE International Conference on Robotics and Automation (ICRA) and the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Multi-agent learning research is also found at Artificial Life and Complexity conferences such as the Conference on Artificial Life (ALife) and the International Conference on Simulation of Adaptive Behavior (SAB). Some conferences specialize in multi-agent systems specifically. The current largest such event is the conference on Autonomous Agents and Multi-Agent Systems (AAMAS). Other notable meetings include the symposium series on Adaptive Agents and Multi-Agent Systems, the European Workshop on Multi-Agent Systems (EUMAS) and the UK Workshop on Multiagent Systems (UKMAS).

Larger conferences usually host smaller workshops in narrower topics; some collections of papers presented at such multi-agent learning workshops are published in [124, 128, 235, 285, 286, 289]. There have also been independent, smaller symposia focused specifically on multi-agent learning: for example, the AAAI-1996 Spring Symposium on Adaptation, Coevolution and Learning in Multiagent Systems; the AAAI-2002 Spring Symposium on Collaborative Learning Agents; and the AAAI-2004 Fall Symposium on Artificial Multiagent Learning.

Texts Issues directly relevant to the area of multi-agent learning are covered in [284, 287]. Other surveys of existing work in multi-agent learning and related domains include [23, 33, 43, 66, 65, 72, 74, 144, 150, 194, 273, 234, 255, 283, 288]. There are also several PhD theses investigating aspects of multi-agent learning ranging from communication and agent modeling, to credit assignment and co-adaptation: [8, 25, 44, 57, 69, 86, 93, 158, 179, 205, 253, 296].

Journals As of the time of writing this article, there is no journal focused on multi-agent learning issues alone. Rather, articles on such topics may be found in journals in related fields, such as the Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS), Machine Learning Journal (MLJ), Journal of Machine Learning Research (JMLR), Evolutionary Computation Journal (ECJ), IEEE Transactions on Evolutionary Computation, and Artificial Life.

Online Resources Several websites offer up-to-date information on events, companies, laboratories, and publications in the area of multi-agents. We found http://www.multiagent.com to be a good source of information in the field. Another such site is http://agents.umbc.edu, though its updates are less frequent.

Competitions The annual RoboCup competition (http://www.robocup.org) offers robotic soccer as a testbed for cooperative multi-agent strategies pitted against other soccer teams submitted by other researchers. RoboCup includes many leagues: middle size robots, small "table-top" robots, quadrupeds, bipeds, and softbots. RoboCup Rescue (http://www.rescuesystem.org/robocuprescue/) is another cooperative multi-agent competition in simulation or with real robots, in which heterogeneous teams of agents must perform disaster rescue missions. Various conferences and research groups organize cooperative multi-agent competitions or challenges from time to time: for example, the GECCO 2004 conference held a competition called Cellz for evolving an optimal controller for a colony of self-replicating cells.

8 Conclusions

Cooperative multi-agent learning is a relatively new area, fraught with complex and rich dynamics, but which also holds the promise of widespread applicability. Researchers have traditionally approached the problem from a number of different directions: reinforcement learning and stateless environments, evolutionary computation and stochastic optimization, cellular automata and other "swarm complexity" models, robotics, and game theory. In this survey we attempted to collect these disparate approaches to multi-agent learning and unify them under two broad multi-agent learning methods (team learning and concurrent learning). The difference in how these methods approach the learning task results in their respective literature breaking down along rather different lines. We also discussed the crucial issue of communication in multi-agent learning, detailed open topics in the field (scalability, adaptive dynamics, and credit assignment), and provided common problem domains and multi-agent resources.

Initial work in multi-agent learning has generally focused on relatively simple environments. Indeed we are surprised by the number of multi-agent learning papers involving a mere *two* agents. We think there are several areas in which multi-agent learning field can "scale up" to more realistic and interesting environments. Here are four:

- **Multiple Agents** The "multi" in multi-agent learning cries out for larger numbers of agents, in the range of ten to thousands or more. Two- and three-agent scenarios are reasonable simplifications to make theoretical analysis feasible: but the experimental and empirical literature ought to strive for more.
- **Team Heterogeneity** If there are more than two agents, most of the literature presumes that the agents are identical in behavior and in ability. Yet it is common for agents to have different capabilities, for missions to require agents with explicitly different behaviors, etc. When dealing with large numbers of agents it is of course not plausible to have total heterogeneity: but agents may fall under some number of heterogeneous classes. These directions of research have not been adequately explored.
- **Agents with Internal State or other Complexity** Much of the multi-agent learning literature simplifies agent behaviors in order to reduce the total search space and to enable agents to understand each other more easily. But many multi-agent environments, particularly in robotics, benefit considerably from agents with more complexity, and especially more internal (hidden) state.
- **Dynamically Changing Teams and Scenarios** What happens when various agents fail, or are required to perform some different task? When new agents are added to the scenario? When new abilities come on-line? We applaud efforts such as RoboCup [140] to push the field to consider more dynamic scenarios, but more needs to be done yet.

As discussed in Section 5, these and other areas present serious scalability challenges due to larger numbers of agents, their possible heterogeneity, and their interactions. Such areas also present more complex game-theoretic difficulties as the agents constantly "move the goalposts" on one another. And last, the environments demand (and present interesting approaches to) new ways of problem decomposition to simplify the search space.

Along with multi-agent systems in general, multi-agent learning is at the cusp of a major growth spurt. Many fields have finally adopted multi-agent models and frameworks due to inexpensive computational brawn. These include models of social, political, and economic behavior; population-, systems-, and cellular-biology models and genomics; and robotics, video-game and virtual-reality agents, and simulations for animation. Multi-agent frameworks have come into their own with the onset of the Internet, cluster computing, and peer-to-peer architectures. We imagine that these environments will all call upon machine learning to assist in the automatic development of multi-agent behaviors, particularly as the complexity of such environments and their "emergent phenomena" becomes such that hand-coded solutions are not readily obtained.

9 Acknowledgments

The authors would like to thank Ken De Jong, Paul Wiegand, Gabriel Balan, Jeff Bassett, Mitch Potter, Valentin Prudius, Jayshree Sarma, Marcel Barbulescu, Adrian Grajdeanu, Elena Popovici, Keith Sullivan, Zbigniew Skolicki, Joerg Denzinger, Michael Littman, the participants to 2004 AAAI Fall Symposium on Artificial Multiagent Learning, and the reviewers for help in preparing this survey.

References

[1] D. H. Ackley and M. Littman. Altruism in the evolution of communication. In *Artificial Life IV: Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems, third edition*. MIT Press, 1994.

[2] D. Andre, F. Bennett III, and J. Koza. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In *Genetic Programming 1996: Proceedings of the First Annual Conference*. MIT Press, 1996.

[3] D. Andre and A. Teller. Evolving team Darwin United. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, 1999.

[4] P. Angeline and J. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, pages 264–270, San Mateo, CA, 1993. Morgan Kaufmann.

[5] W. Arthur. Inductive reasoning and bounded rationality. *Complexity in Economic Theory*, 84(2):406–411, 1994.

[6] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolutionary Straegies, Evolutionary Programming, and Genetic Algorithms*. Oxford Press, 1996.

[7] T. Balch. Learning roles: Behavioral diversity in robot teams. Technical Report GIT-CC-97-12, Georgia Institute of Technology, 1997.

[8] T. Balch. *Behavioral Diversity in Learning Robot Teams*. PhD thesis, College of Computing, Georgia Institute of Technology, 1998.

[9] T. Balch. Reward and diversity in multirobot foraging. In *IJCAI-99 Workshop on Agents Learning About, From and With other Agents*, 1999.

[10] B. Banerjee, R. Mukherjee, and S. Sen. Learning mutual trust. In *Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies*, pages 9–14, 2000.

[11] A. Barto, R. Sutton, and C. Watkins. Learning and sequential decision making. In M. Gabriel and J. Moore, editors, *Learning and computational neuroscience : foundations of adaptive networks*. M.I.T. Press, Cambridge, Mass, 1990.

[12] J. Bassett and K. De Jong. Evolving behaviors for cooperating agents. In Z. Ras, editor, *Proceedings from the Twelfth International Symposium on Methodologies for Intelligent Systems*, pages 157–165, Charlotte, NC., 2000. Springer-Verlag.

[13] J. K. Bassett. A study of generalization techniques in evolutionary rule learning. Master's thesis, George Mason University, Fairfax VA, USA, 2002.

[14] R. Beckers, O. E. Holland, and J.-L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Artificial Life IV: Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems , third edition*. MIT Press, 1994.

[15] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources — an empirical investigation. Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computer Services, 1986.

[16] H. Berenji and D. Vengerov. Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. In *Proceedings of 9th IEEE International Conference on Fuzzy Systems*, 2000.

[17] H. Berenji and D. Vengerov. Learning, cooperation, and coordination in multi-agent systems. Technical Report IIS-00-10, Intelligent Inference Systems Corp., 333 W. Maude Avennue, Suite 107, Sunnyvale, CA 94085-4367, 2000.

[18] D. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of MDPs. In *Proceedings of UAI-2000: The Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[19] H. J. Blumenthal and G. Parker. Co-evolving team capture strategies for dissimilar robots. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[20] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. SFI Studies in the Sciences of Complexity. Oxford University Press, 1999.

[21] J. C. Bongard. The legion system: A novel approach to evolving heterogeneity for collective problem solving. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, editors, *Genetic Programming: Proceedings of EuroGP-2000*, volume 1802, pages 16–28, Edinburgh, 15-16 2000. Springer-Verlag. ISBN 3-540-67339-3.

[22] C. Boutilier. Learning conventions in multiagent stochastic domains using likelihood estimates. In *Uncertainty in Artificial Intelligence*, pages 106–114, 1996.

[23] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK96)*, pages 195–210, 1996.

[24] M. Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 89–94. Morgan Kaufmann, San Francisco, CA, 2000.

[25] M. Bowling. *Multiagent learning in the presence of agents with limitations*. PhD thesis, Computer Science Department, Carnegie Mellon University, 2003.

[26] M. Bowling and M. Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical Report CMU-CS-00-165, Computer Science Department, Carnegie Mellon University, 2000.

[27] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1021–1026, 2001.

[28] M. Bowling and M. Veloso. Existence of multiagent equilibria with limited agents. Technical Report CMU-CS-02-104, Computer Science Department, Carnegie Mellon University, 2002.

[29] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2): 215–250, 2002.

[30] J. A. Boyan and M. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 671–678. Morgan Kaufmann Publishers, Inc., 1994.

[31] R. Brafman and M. Tennenholtz. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems (NIPS-2002)*, 2002.

[32] W. Brauer and G. Weiß. Multi-machine scheduling — a multi-agent learning approach. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 42–48, 1998.

[33] P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. van de Velde. Learning in distributed systems and multi-agent environments. In Y. Kodratoff, editor, *Lecture Notes in Artificial Intelligence, Volume 482*, pages 412–423. Springer-Verlag, 1991.

[34] O. Buffet, A. Dutech, and F. Charpillet. Incremental reinforcement learning for designing multi-agent systems. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 31–32, Montreal, Canada, 2001. ACM Press.

[35] O. Buffet, A. Dutech, and F. Charpillet. Learning to weigh basic behaviors in scalable agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'02)*, 2002.

[36] H. Bui, S. Venkatesh, and D. Kieronska. A framework for coordination and learning among team of agents. In W. Wobcke, M. Pagnucco, and C. Zhang, editors, *Agents and Multi-Agent Systems: Formalisms, Methodologies and Applications, Lecture Notes in Artificial Intelligence, Volume 1441*, pages 164–178. Springer-Verlag, 1998.

[37] H. Bui, S. Venkatesh, and D. Kieronska. Learning other agents' preferences in multi-agent negotiation using

the Bayesian classifier. *International Journal of Cooperative Information Systems*, 8(4):275–294, 1999.

[38] L. Bull. Evolutionary computing in multi-agent environments: Partners. In T. Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 370–377. Morgan Kaufmann, 1997.

[39] L. Bull. Evolutionary computing in multi-agent environments: Operators. In D. W. V W Porto, N Saravanan and A. E. Eiben, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 43–52. Springer Verlag, 1998.

[40] L. Bull and T. C. Fogarty. Evolving cooperative communicating classifier systems. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming (EP94)*, pages 308–315, 1994.

[41] L. Bull and O. Holland. Evolutionary computing in multiagent environments: Eusociality. In *Proceedings of Seventh Annual Conference on Genetic Algorithms*, 1997.

[42] A. Cangelosi. Evolution of communication and language using signals, symbols, and words. *IEEE Transactions on Evolutionary Computation*, 5(2):93–101, 2001.

[43] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, 1997.

[44] D. Carmel. *Model-based Learning of Interaction Strategies in Multi-agent systems*. PhD thesis, Technion — Israel Institute of Technology, 1997.

[45] D. Carmel and S. Markovitch. The M* algorithm: Incorporating opponent models into adversary search. Technical Report 9402, Technion - Israel Institute of Technology, March 1994.

[46] L.-E. Cederman. *Emergent Actors in World Politics: How States and Nations Develop and Dissolve*. Princeton University Press, 1997.

[47] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *Proceedings of The Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2003)*. ACM, 2003. ISBN 1-58113-683-8.

[48] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe. Electric elves: agent technology for supporting human organizations. In *AI Magazine - Summer 2002*. AAAI Press, 2002.

[49] Y.-H. Chang, T. Ho, and L. Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Proceedings of Neural Information Processing Systems (NIPS-03)*, 2003.

[50] Y.-H. Chang, T. Ho, and L. Kaelbling. Multi-agent learning in mobilized ad-hoc networks. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[51] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of National Conference on Artificial IntelligenceAAAI/IAAI*, pages 746–752, 1998.

[52] D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *Proceedings of the Third European Conference on Artificial Life*, pages 200–218. Springer-Verlag, 1995.

[53] R. Collins and D. Jefferson. An artificial neural network representation for artificial organisms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature: 1st Workshop (PPSN I)*, pages 259–263, Berlin, 1991. Springer-Verlag.

[54] R. Collins and D. Jefferson. AntFarm : Towards simulated evolution. In C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 579–601. Addison-Wesley, Redwood City, CA, 1992.

[55] E. Crawford and M. Veloso. Opportunities for learning in multi-agent meeting scheduling. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[56] V. Crespi, G. Cybenko, M. Santini, and D. Rus. Decentralized control for coordinated flow of multi-agent systems. Technical Report TR2002-414, Dartmouth College, Computer Science, Hanover, NH, January 2002.

[57] R. H. Crites. *Large-Scale Dynamic Optimization Using Teams of Reinforcement Learning Agents*. PhD thesis, University of Massachusetts Amherst, 1996.

[58] M. R. Cutkosky, R. S. Englemore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum, and J. C. Weber. PACT: An experiment in integrating concurrent engineering systems. In M. N. Huhns and M. P. Singh, editors, *Readings in Agents*, pages 46–55. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[59] T. Dahl, M. Mataric, and G. Sukhatme. Adaptive spatio-temporal organization in groups of robots. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-02)*, 2002.

[60] R. Das, M. Mitchell, and J. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In *Parallel Problem Solving from Nature III, LNCS 866*, pages 344–353. Springer-Verlag, 1994.

[61] J. Davis and G. Kendall. An investigation, using co-evolution, to evolve an awari player. In *Proceedings of 2002 Congress on Evolutionary Computation (CEC2002)*, 2002.

[62] B. de Boer. Generating vowel systems in a population of agents. In *Proceedings of the Fourth European Conference Artificial Life*. MIT Press, 1997.

[63] K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

[64] K. De Jong. *Evolutionary Computation: A unified approach*. MIT Press, 2005.

[65] K. Decker, E. Durfee, and V. Lesser. Evaluating research in cooperative distributed problem solving. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, pages 487–519. Pitman Publishing and Morgan Kaufmann, 1989.

[66] K. Decker, M. Fisher, M. Luck, M. Tennenholtz, and UKMAS'98 Contributors. Continuing research in multi-agent systems. *Knowledge Engineering Review*, 14(3):279–283, 1999.

[67] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ants and ant-like robots. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 356–363. MIT Press, 1991.

[68] J. Denzinger and M. Fuchs. Experiments in learning prototypical situations for variants of the pursuit game. In *Proceedings on the International Conference on Multi-Agent Systems (ICMAS-1996)*, pages 48–55, 1996.

[69] M. Dowell. *Learning in Multiagent Systems*. PhD thesis, University of South Carolina, 1995.

[70] K. Dresner and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[71] G. Dudek, M. Jenkin, R. Milios, and D. Wilkes. A taxonomy for swarm robots. In *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems*, 1993.

[72] E. Durfee. What your computer really needs to know, you learned in kindergarten. In *National Conference on Artificial Intelligence*, pages 858–864, 1992.

[73] E. Durfee, V. Lesser, and D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, 1987.

[74] E. Durfee, V. Lesser, and D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, KDE-1(1):63–83, March 1989.

[75] A. Dutech, O. Buffet, and F. Charpillet. Multi-agent systems by incremental gradient reinforcement learning. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 833–838, 2001.

[76] F. Fernandez and L. Parker. Learning in large cooperative multi-robot domains. *International Journal of Robotics and Automation*, 16(4):217–226, 2001.

[77] S. Ficici and J. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In C. Adami *et al*, editor, *Proceedings of the Sixth International Conference on Artificial Life*, pages 238–247, Cambridge, MA, 1998. MIT Press.

[78] S. Ficici and J. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*. Springer Verlag, 2000.

[79] K. Fischer, N. Kuhn, H. J. Muller, J. P. Muller, and M. Pischel. Sophisticated and distributed: The transportation domain. In *Proceedings of the Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'93)*, 1993.

[80] D. Fogel. *Blondie24: Playing at the Edge of Artificial Intelligence*. Morgan Kaufmann, 2001. ISBN 1-55860-783-8.

[81] L. Fogel. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. Wiley Series on Intelligent Systems, 1999.

[82] D. Fudenberg and D. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[83] A. Garland and R. Alterman. Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems*, 8:267–301, 2004.

[84] M. Ghavamzadeh and S. Mahadevan. Learning to communicate and act using hierarchical reinforcement learning. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[85] N. Glance and B. Huberman. The dynamics of social dilemmas. *Scientific American*, 270(3):76–81, March 1994.

[86] P. Gmytrasiewicz. *A Decision-Theoretic Model of Coordination and Communication in Autonomous Systems (Reasoning Systems)*. PhD thesis, University of Michigan, 1992.

[87] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[88] C. Goldman and J. Rosenschein. Mutually supervised learning in multiagent systems. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, pages 85–96. Springer-Verlag: Heidelberg, Germany, Berlin, 1996.

[89] B. M. Good. Evolving multi-agent systems: Comparing existing approaches and suggesting new directions. Master's thesis, University of Sussex, 2000.

[90] M. Gordin, S. Sen, and N. Puppala. Evolving cooperative groups: Preliminary results. In *Working Papers of the AAAI-97 Workshop on Multiagent Learning*, pages 31–35, 1997.

[91] S. Grand and D. Cliff. Creatures: Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 1(1):39–57, 1998.

[92] S. Grand, D. Cliff, and A. Malhotra. Creatures : Artificial life autonomous software agents for home entertainment. In *Proceedings of the First International Conference on Autonomous Agents (Agents-97)*, pages 22–29, 1997.

[93] D. L. Grecu. *Using Learning to Improve Multi-Agent Systems for Design*. PhD thesis, Worcester Polytechnic Institute, 1997.

[94] A. Greenwald, J. Farago, and K. Hall. Fair and efficient solutions to the Santa Fe bar problem. In *Proceedings of the Grace Hopper Celebration of Women in Computing 2002*, 2002.

[95] A. Greenwald and K. Hall. Correlated Q-learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[96] J. Grefenstette. Lamarckian learning in multi-agent environments. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 303–310, San Mateo, CA, 1991.

Morgan Kaufman.

[97] J. Grefenstette, C. L. Ramsey, and A. Schultz. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5:355–381, 1990.

[98] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of the 2002 AAAI Symposium Series: Collaborative Learning Agents*, 2002.

[99] S. M. Gustafson. Layered learning in genetic programming for a co-operative robot soccer problem. Master's thesis, Kansas State University, Manhattan, KS, USA, 2000.

[100] S. M. Gustafson and W. H. Hsu. Layered learning in genetic programming for a co-operative robot soccer problem. In J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, and W. B. Langdon, editors, *Genetic Programming: Proceedings of EuroGP-2001*, volume 2038, pages 291–301, Lake Como, Italy, 18-20 2001. Springer-Verlag. ISBN 3-540-41899-7.

[101] A. Hara and T. Nagao. Emergence of cooperative behavior using ADG; Automatically Defined Groups. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 1038–1046, 1999.

[102] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics : the Sussex approach. *Robotics and Autonomous Systems*, 1996.

[103] T. Haynes, K. Lau, and S. Sen. Learning cases to compliment rules for conflict resolution in multiagent systems. In S. Sen, editor, *AAAI Spring Symposium on Adaptation, Coevolution, and Learning in Multiagent Systems*, pages 51–56, 1996.

[104] T. Haynes and S. Sen. Evolving behavioral strategies in predators and prey. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multiagent Systems*, Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, Germany, 1995.

[105] T. Haynes and S. Sen. Adaptation using cases in cooperative groups. In I. Imam, editor, *Working Notes of the AAAI-96 Workshop on Intelligent Adaptive Agents*, Portland, OR, 1996.

[106] T. Haynes and S. Sen. Cooperation of the fittest. Technical Report UTULSA-MCS-96-09, The University of Tulsa, Apr. 12, 1996.

[107] T. Haynes and S. Sen. Learning cases to resolve conflicts and improve group behavior. In M. Tambe and P. Gmytrasiewicz, editors, *Working Notes of the AAAI-96 Workshop on Agent Modeling*, pages 46–52, Portland, OR, 1996.

[108] T. Haynes and S. Sen. Crossover operators for evolving a team. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 162–167, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[109] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright. Evolving a team. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 23–30, MIT, Cambridge, MA, USA, 10–12 Nov. 1995. AAAI.

[110] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright. Evolving multiagent coordination strategies with genetic programming. Technical Report UTULSA-MCS-95-04, The University of Tulsa, May 31, 1995.

[111] T. Haynes, R. Wainwright, S. Sen, and D. Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 271–278, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann. ISBN 1-55860-370-0.

[112] T. D. Haynes and S. Sen. Co-adaptation in a team. *International Journal of Computational Intelligence and Organizations (IJCIO)*, 1997.

[113] D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Artificial Life II, SFI Studies in the Sciences of Complexity*, 10:313–324, 1991.

[114] J. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, 1975.

[115] J. Holland. Properties of the bucket brigade. In *Proceedings of an International Conference on Genetic Algorithms*, 1985.

[116] B. Hölldobler and E. O. Wilson. *The Ants*. Harvard University Press, 1990.

[117] W. H. Hsu and S. M. Gustafson. Genetic programming and multi-agent layered learning by reinforcements. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 764–771, New York, 9-13 July 2002. Morgan Kaufmann Publishers. ISBN 1-55860-878-8.

[118] J. Hu and M. Wellman. Self-fulfilling bias in multiagent learning. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.

[119] J. Hu and M. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[120] J. Hu and M. Wellman. Online learning about other agents in a dynamic multiagent system. In K. P. Sycara and M. Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents'98)*, pages 239–246, New York, 1998. ACM Press. ISBN 0-89791-983-1.

[121] J. Hu and M. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[122] J. Huang, N. R. Jennings, and J. Fox. An agent architecture for distributed medical care. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 219–232. Springer-Verlag: Heidelberg, Germany, 1995.

[123] M. Huhns and M. Singh. Agents and multiagent systems: themes, approaches and challenges. In M. Huhns and M. Singh, editors, *Readings in Agents*, pages 1–23. Morgan Kaufmann, 1998.

[124] M. Huhns and G. Weiß. Special issue on multiagent learning. *Machine Learning Journal*, 33(2-3), 1998.

[125] H. Iba. Emergent cooperation for multiple agents using genetic programming. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature IV: Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *LNCS*, pages 32–41, Berlin, Germany, 1996. Springer Verlag. ISBN 3-540-61723-X.

[126] H. Iba. Evolutionary learning of communicating agents. *Information Sciences*, 108, 1998.

[127] H. Iba. Evolving multiple agents by genetic programming. In L. Spector, W. Langdon, U.-M. O'Reilly, and P. Angeline, editors, *Advances in Genetic Programming 3*, pages 447–466. The MIT Press, Cambridge, MA, 1999.

[128] I. Imam, editor. *Intelligent Adaptive Agents. Papers from the 1996 AAAI Workshop. Technical Report WS-96-04*. AAAI Press, 1996.

[129] A. Ito. How do selfish agents learn to cooperate? In *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 185–192. MIT Press, 1997.

[130] T. Jansen and R. P. Wiegand. Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA. In E. Cantu-Paz *et al*, editor, *Prooceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Springer-Verlag, 2003.

[131] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agents research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[132] N. Jennings, L. Varga, R. Aarnts, J. Fuchs, and P. Skarek. Transforming standalone expert systems into a community of cooperating agents. *International Journal of Engineering Applications of Artificial Intelligence*,

6(4):317–331, 1993.

[133] K.-C. Jim and C. L. Giles. Talking helps: Evolving communicating agents for the predator-prey pursuit problem. *Artificial Life*, 6(3):237–254, 2000.

[134] H. Juille and J. Pollack. Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In *Proceedings of the Third Annual Genetic Programming Conference (GP-98)*, 1998.

[135] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[136] S. Kapetanakis and D. Kudenko. Improving on the reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Second Symposium on Adaptive Agents and Multi-agent Systems (AISB02)*, 2002.

[137] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI02)*, 2002.

[138] G. Kendall and G. Whitwell. An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC-2001)*, pages 995–1002. IEEE Press, 27-30 2001.

[139] G. Kendall and M. Willdig. An investigation of an adaptive poker player. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI'01)*, 2001.

[140] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The robot world cup initiative. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press. ISBN 0-89791-877-0.

[141] J. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[142] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000.

[143] L. R. Leerink, S. R. Schultz, and M. A. Jabri. A reinforcement learning exploration strategy based on ant foraging mechanisms. In *Proceedings of the Sixth Australian Conference on Neural Networks*, Sydney, Australia, 1995.

[144] V. Lesser. Cooperative multiagent systems : A personal view of the state of the art. *IEEE Trans. on Knowledge and Data Engineering*, 11(1):133–142, 1999.

[145] V. Lesser, D. Corkill, and E. Durfee. An update on the distributed vehicle monitoring testbed. Technical Report UM-CS-1987-111, University of Massachessets Amherst, 1987.

[146] M. I. Lichbach. *The cooperator's dilemma*. University of Michigan Press, 1996. ISBN 0472105728.

[147] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[148] M. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328. Morgan Kaufmann Publishers Inc., 2001.

[149] A. Lubberts and R. Miikkulainen. Co-evolving a go-playing neural network. In *Coevolution: Turning Adaptive Algorithms upon Themselves, (Birds-on-a-Feather Workshop, Genetic and Evolutionary Computation Conference)*, 2001.

[150] M. Luck, M. d'Inverno, M. Fisher, and FoMAS'97 Contributors. Foundations of multi-agent systems: Techniques, tools and theory. *Knowledge Engineering Review*, 13(3):297–302, 1998.

[151] S. Luke. Genetic programming produced competitive soccer softbot teams for RoboCup97. In J. R. Koza *et*

al, editor, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 214–222. Morgan Kaufmann, 1998.

[152] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *Proceedings of the First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[153] S. Luke and L. Spector. Evolving teamwork and coordination with genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 28–31 1996. MIT Press.

[154] S. Luke, K. Sullivan, G. C. Balan, and L. Panait. Tunably decentralized algorithms for cooperative target observation. Technical Report GMU-CS-TR-2004-1, Department of Computer Science, George Mason University, 2004.

[155] S. Luke and R. P. Wiegand. Guaranteeing coevolutionary objective measures. In Poli et al. [201], pages 237–251.

[156] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. In *National Conference on Artificial Intelligence*, pages 768–773, 1991.

[157] R. Makar, S. Mahadevan, and M. Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 246–253, Montreal, Canada, 2001. ACM Press.

[158] M. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1994. Also Technical Report AITR-1495.

[159] M. Mataric. Learning to behave socially. In *Third International Conference on Simulation of Adaptive Behavior*, 1994.

[160] M. Mataric. Reward functions for accelerated learning. In *International Conference on Machine Learning*, pages 181–189, 1994.

[161] M. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[162] M. Mataric. Using communication to reduce locality in distributed multi-agent learning. *Joint Special Issue on Learning in Autonomous Robots, Machine Learning, 31(1-3), 141-167, and Autonomous Robots, 5(3-4), Jul/Aug 1998, 335-354*, 1998.

[163] M. Mataric, M. Nilsson, and K. Simsarian. Cooperative multi-robot box-pushing. In *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 556–561, 1995.

[164] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 3rd edition, 1996.

[165] T. Miconi. A collective genetic algorithm. In E. Cantu-Paz *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 876–883, 2001.

[166] T. Miconi. When evolving populations is better than coevolving individuals: The blind mice problem. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[167] M. Mitchell, J. Crutchfield, and R. Das. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA'96)*,, 1996.

[168] N. Monekosso, P. Remagnino, and A. Szarowicz. An improved Q-learning algorithm using synthetic pheromones. In E. N. B. Dunin-Keplicz, editor, *From Theory to Practice in Multi-Agent Systems, Second International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS 2001 Cracow, Poland, September 26-29, 2001. Revised Papers*, Lecture Notes in Artificial Intelligence LNAI-2296. Springer-Verlag, 2002.

[169] N. D. Monekosso and P. Remagnino. Phe-Q: A pheromone based Q-learning. In *Australian Joint Conference on Artificial Intelligence*, pages 345–355, 2001.

[170] N. D. Monekosso and P. Remagnino. An analysis of the pheromone Q-learning algorithm. In *Proceedings of the VIII Iberoamerican Conference on Artificial Intelligence IBERAMIA-02*, pages 224–232, 2002.

[171] N. D. Monekosso, P. Remagnino, and A. Szarowicz. An improved Q-learning algorithm using synthetic pheromones. In *Proceedings of the Second Workshop of Central and Eastern Europe on Multi-Agent Systems CEEMAS-01*, pages 197–206, 2001.

[172] J. Moody, Y. Liu, M. Saffell, and K. Youn. Stochastic direct reinforcement: Application to simple games with recurrence. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[173] R. Mukherjee and S. Sen. Towards a pareto-optimal solution in general-sum games. In *Agents-2001 Workshop on Learning Agents*, 2001.

[174] U. Mukhopadjyay, L. Stephens, and M. Huhns. An intelligent system for document retrieval in distributed office environment. *Journal of the American Society for Information Science*, 37, 1986.

[175] J. Muller and M. Pischel. An architecture for dynamically interacting agents. *Journal of Intelligent and Cooperative Information Systems*, 3(1):25–45, 1994.

[176] M. Mundhe and S. Sen. Evaluating concurrent reinforcement learners. In *Proceedings of the International Conference on Multiagent System*, 2000.

[177] M. Mundhe and S. Sen. Evolving agent societies that avoid social dilemmas. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 809–816, Las Vegas, Nevada, USA, 10-12 2000. Morgan Kaufmann. ISBN 1-55860-708-0.

[178] Y. Nagayuki, S. Ishii, and K. Doya. Multi-agent reinforcement learning: An approach based on the other agent's internal model. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-00)*, 2000.

[179] M. V. Nagendra-Prasad. *Learning Situation-Specific Control in Multi-Agent Systems*. PhD thesis, University of Massachusetts Amherst, 1997.

[180] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[181] M. Nowak and K. Sigmund. Evolution of indirect reciprocity by image scoring / the dynamics of indirect reciprocity. *Nature*, 393:573–577, 1998.

[182] A. Nowe, K. Verbeeck, and T. Lenaerts. Learning agents in a homo egualis society. Technical report, Computational Modeling Lab - VUB, March 2001.

[183] L. Nunes and E. Oliveira. Learning from multiple sources. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[184] T. Ohko, K. Hiraki, and Y. Arzai. Addressee learning and message interception for communication load reduction in multiple robots environments. In G. Weiß, editor, *Distributed Artificial Intelligence Meets Machine Learning : Learning in Multi-Agent Environments, Lecture Notes in Artificial Intelligence 1221*. Springer-Verlag, 1997.

[185] E. Ostergaard, G. Sukhatme, and M. Mataric. Emergent bucket brigading - a simple mechanism for improving performance in multi-robot constrainedspace foraging tasks. In *Proceedings of the Fifth International Conference on Autonomous Agents,*, 2001.

[186] L. Pagie and M. Mitchell. A comparison of evolutionary and coevolutionary search. In R. K. Belew and H. Juillè, editors, *Coevolution: Turning Adaptive Algorithms upon Themselves*, pages 20–25, San Francisco, California, USA, 7 2001.

[187] L. Panait and S. Luke. Ant foraging revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 2004.

[188] L. Panait and S. Luke. Learning ant foraging behaviors. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 2004.

[189] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[190] L. Panait, R. P. Wiegand, and S. Luke. A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization. In *Genetic and Evolutionary Computation Conference — GECCO-2004*. Springer, 2004.

[191] L. Panait, R. P. Wiegand, and S. Luke. A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature — PPSN-2004*. Springer, 2004.

[192] L. A. Panait, R. P. Wiegand, and S. Luke. Improving coevolutionary search for optimal multiagent behaviors. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[193] C. Papadimitriou and J. Tsitsiklis. Complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[194] L. Parker. Current state of the art in distributed autonomous mobile robotics. In L. Parker, G. Bekey, and J. Barhen, editors, *Distributed Autonomous Robotic Systems 4*,, pages 3–12. Springer-Verlag, 2000.

[195] L. Parker. Multi-robot learning in a cooperative observation task. In *Proceedings of Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000)*, 2000.

[196] L. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3), 2002.

[197] L. Parker, C. Touzet, and F. Fernandez. Techniques for learning in multi-robot teams. In T. Balch and L. Parker, editors, *Robot Teams: From Diversity to Polymorphism*. AK Peters, 2001.

[198] M. Peceny, G. Weiß, and W. Brauer. Verteiltes maschinelles lernen in fertigungsumgebungen. Technical Report FKI-218-96, Institut fur Informatik, Technische Universitat Munchen, 1996.

[199] M. Peeters, K. Verbeeck, and A. Nowe. Multi-agent learning in conflicting multi-level games with incomplete information. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[200] L. Peshkin, K.-E. Kim, N. Meuleau, and L. Kaelbling. Learning to cooperate via policy search. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–314. Morgan Kaufmann, 2000.

[201] R. Poli, J. Rowe, and K. D. Jong, editors. *Foundations of Genetic Algorithms (FOGA) VII*, 2002. Morgan Kaufmann.

[202] J. Pollack and A. Blair. Coevolution in the successful learning of backgammon strategy. *Machine Learning*, 32 (3):225–240, 1998.

[203] J. Pollack, A. Blair, and M. Land. Coevolution of a backgammon player. In C. G. Langton and K. Shimohara, editors, *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 92–98, Cambridge, MA, 1997. The MIT Press.

[204] E. Popovici and K. DeJong. Understanding competitive co-evolutionary dynamics via fitness landscapes. In *Artificial Multiagent Symposium. Part of the 2004 AAAI Fall Symposium on Artificial Intelligence*, 2004.

[205] M. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997.

[206] M. Potter and K. De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P. Schwefel, editors, *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pages 249–257. Springer-Verlag, 1994.

[207] M. Potter and K. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[208] M. Potter, K. De Jong, and J. J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In *Proceedings from the Sixth International Conference on Genetic Algorithms*, pages 366–372. Morgan Kaufmann Publishers, Inc., 1995.

[209] M. Potter, L. Meeden, and A. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of The Seventeenth International Conference on Artificial Intelligence (IJCAI-2001)*, 2001.

[210] N. Puppala, S. Sen, and M. Gordin. Shared memory based cooperative coevolution. In *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pages 570–574, Anchorage, Alaska, USA, 1998. IEEE Press.

[211] M. Quinn. A comparison of approaches to the evolution of homogeneous multi-robot teams. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, pages 128–135, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 2001. IEEE Press. ISBN 0-7803-6658-1.

[212] M. Quinn. Evolving communication without dedicated communication channels. In *Advances in Artificial Life: Sixth European Conference on Artificial Life (ECAL01)*, 2001.

[213] M. Quinn, L. Smith, G. Mayley, and P. Husbands. Evolving formation movement for a homogeneous multi-robot system: Teamwork and role-allocation with real robots. Cognitive Science Research Paper 515. School of Cognitive and Computing Sciences, University of Sussex, Brighton, BN1 9QG. ISSN 1350-3162, 2002.

[214] C. Reynolds. An evolved, vision-based behavioral model of coordinated group motion. In *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, pages 384–392, 1993.

[215] C. Reynolds. Competition, coevolution and the game of tag. In R. A. Brooks and P. Maes, editors, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.*, pages 59–69. MIT Press, 1994.

[216] C. W. Reynolds. Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[217] P. Riley and M. Veloso. On behavior classification in adversarial environments. In L. Parker, G. Bekey, and J. Barhen, editors, *Distributed Autonomous Robotic Systems 4*, pages 371–380. Springer-Verlag, 2000.

[218] A. Robinson and L. Spector. Using genetic programming with multiple data types and automatic modularization to evolve decentralized and coordinated navigation in multi-agent systems. In *In Late-Breaking Papers of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. The International Society for Genetic and Evolutionary Computation, 2002.

[219] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[220] R. Salustowicz, M. Wiering, and J. Schmidhuber. Learning team strategies with multiple policy-sharing agents: A soccer case study. Technical report, ISDIA, Corso Elvezia 36, 6900 Lugano, Switzerland, 1997.

[221] R. Salustowicz, M. Wiering, and J. Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 33(2/3):263–282, 1998.

[222] A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[223] T. Sandholm and R. H. Crites. On multiagent Q-learning in a semi-competitive domain. In *Adaption and Learning in Multi-Agent Systems*, pages 191–205, 1995.

[224] H. Santana, G. Ramalho, V. Corruble, and B. Ratitch. Multi-agent patrolling with reinforcement learning. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[225] G. Saunders and J. Pollack. The evolution of communication schemes over continuous channels. In *From Animals to Animats 4 - Proceedings of the Fourth International Conference on Adaptive Behaviour,*, 1996.

[226] J. Sauter, R. S. Matthews, H. Van Dyke Parunak, and S. Brueckner. Evolving adaptive pheromone path planning mechanisms. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 434–440, 2002.

[227] J. Sauter, H. Van Dyke Parunak, S. Brueckner, and R. Matthews. Tuning synthetic pheromones with evolutionary computing. In R. E. Smith, C. Bonacina, C. Hoile, and P. Marrow, editors, *Evolutionary Computation and Multi-Agent Systems (ECOMAS)*, pages 321–324, San Francisco, California, USA, 7 2001.

[228] J. Schmidhuber. Realistic multi-agent reinforcement learning. In *Learning in Distributed Artificial Intelligence Systems, Working Notes of the 1996 ECAI Workshop*, 1996.

[229] J. Schmidhuber and J. Zhao. Multi-agent learning with the success-story algorithm. In *ECAI Workshop LDAIS / ICMAS Workshop LIOME*, pages 82–93, 1996.

[230] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller. Distributed value functions. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 371–378, 1999.

[231] A. Schultz, J.Grefenstette, and W. Adams. Robo-shepherd: Learning complex robotic behaviors. In *Robotics and Manufacturing: Recent Trends in Research and Applications, Volume 6*, pages 763–768. ASME Press, 1996.

[232] U. M. Schwuttke and A. G. Quan. Enhancing performance of cooperating agents in realtime diagnostic systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, 1993.

[233] M. Sekaran and S. Sen. To help or not to help. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages 736–741, Pittsburgh, PA, 1995.

[234] S. Sen. Multiagent systems: Milestones and new horizons. *Trends in Cognitive Science*, 1(9):334–339, 1997.

[235] S. Sen. Special issue on evolution and learning in multiagent systems. *International Journal of Human-Computer Studies*, 48(1), 1998.

[236] S. Sen and M. Sekaran. Using reciprocity to adapt to others. In G. Weiß and S. Sen, editors, *International Joint Conference on Artificial Intelligence Workshop on Adaptation and Learning in Multiagent Sytems, Lecture Notes in Artificial Intelligence*, pages 206–217. Springer-Verlag, 1995.

[237] S. Sen and M. Sekaran. Multiagent coordination with learning classifier systems. In G. Weiß and S. Sen, editors, *Proceedings of the IJCAI Workshop on Adaption and Learning in Multi-Agent Systems*, volume 1042, pages 218–233. Springer Verlag, 1996. ISBN 3-540-60923-7.

[238] S. Sen and M. Sekaran. Individual learning of coordination knowledge. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):333–356, 1998.

[239] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, 1994.

[240] Y. Shoham, R. Powers, and T. Grenager. On the agenda(s) of research on multi-agent learning. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[241] R. Smith and B. Gray. Co-adaptive genetic algorithms: An example in othello strategy. Technical Report TCGA 94002, University of Alabama, Department of Engineering Science and Mechanics, 1993.

[242] L. Spector and J. Klein. Evolutionary dynamics discovered via visualization in the breve simulation environment. In *Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems*, pages 163–170, 2002.

[243] L. Spector, J. Klein, C. Perry, and M. Feinstein. Emergence of collective behavior in evolving populations of flying agents. In E. Cantu-Paz *et al*, editor, *Prooceedings of the Genetic and Evolutionary Computation*

Conference (GECCO). Springer-Verlag, 2003.

[244] R. Steeb, S. Cammarata, F. Hayes-Roth, P. Thorndyke, and R. Wesson. Distributed intelligence for air fleet control. In A. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 90–101. Morgan Kaufmann Publishers, 1988.

[245] L. Steels. A self-organizing spatial vocabulary. *Artificial Life*, 2(3):319–332, 1995.

[246] L. Steels. Emergent adaptive lexicons. In P. Maes, editor, *Proceedings of the Simulation of Adaptive Behavior Conference*. MIT Press, 1996.

[247] L. Steels. Self-organising vocabularies. In *Proceedings of Artificial Life V*, 1996.

[248] L. Steels. The spontaneous self-organization of an adaptive language. In S. Muggleton, editor, *Machine Intelligence 15*. Oxford University Press, Oxford, UK, 1996.

[249] L. Steels. Synthesising the origins of language and meaning using co-evolution, self-organisation and level formation. In J. Hurford, C. Knight, and M. Studdert-Kennedy, editors, *Approaches to the Evolution of Language: Social and Cognitive bases*. Edinburgh University Press, 1997.

[250] L. Steels. The puzzle of language evolution. *Kognitionswissenschaft*, 8(4):143–150, 2000.

[251] L. Steels and F. Kaplan. Collective learning and semiotic dynamics. In *Proceedings of the European Conference on Artificial Life*, pages 679–688, 1999.

[252] P. Stone. Layered learning in multiagent systems. In *Proceedings of National Conference on Artificial IntelligenceAAAI/IAAI*, 1997.

[253] P. Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Carnegie Mellon University, 1998.

[254] P. Stone and R. Sutton. Keepaway soccer: A machine learning testbed. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*, pages 214–223. Springer, 2002. ISBN 3-540-43912-9.

[255] P. Stone and M. M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[256] N. Sturtevant and R. Korf. On pruning techniques for multi-player games. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 201–207, 2000.

[257] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 832–839, 1997.

[258] N. Suematsu and A. Hayashi. A multiagent reinforcement learning algorithm using extended optimal response. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 370–377, 2002.

[259] D. Suryadi and P. J. Gmytrasiewicz. Learning models of other agents using influence diagrams. In *Preceedings of the 1999 International Conference on User Modeling*, pages 223–232, 1999.

[260] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[261] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[262] J. Svennebring and S. Koenig. Trail-laying robots for robust terrain coverage. In *Proceedings of the International Conference on Robotics and Automation (ICRA-03)*, 2003.

[263] P. 't Hoen and K. Tuyls. Analyzing multi-agent reinforcement learning using evolutionary dynamics. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, 2004.

[264] M. Tambe. Recursive agent and agent-group tracking in a real-time dynamic environment. In V. Lesser and L. Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems (ICMAS-95)*. AAAI Press, 1995.

[265] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. In M. N. Huhns and M. P. Singh, editors, *Readings in Agents*, pages 487–494. Morgan Kaufmann, San Francisco, CA, USA, 1993.

[266] P. Tangamchit, J. Dolan, and P. Khosla. The necessity of average rewards in cooperative multirobot learning. In *Proceedings of IEEE Conference on Robotics and Automation*, 2002.

[267] G. Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[268] G. Tesauro and J. O. Kephart. Pricing in agent economies using multi-agent Q-learning. *Autonomous Agents and Multi-Agent Systems*, 8:289–304, 2002.

[269] S. Thrun. Learning to play the game of chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 1069–1076. The MIT Press, Cambridge, MA, 1995.

[270] K. Tumer, A. K. Agogino, and D. H. Wolpert. A bartering approach to improve multiagent learning. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 386–393, 2002.

[271] K. Tuyls, K. Verbeeck, and T. Lenaerts. A selection-mutation model for Q-learning in multi-agent systems. In *AAMAS-2003 — Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[272] W. Uther and M. Veloso. Adversarial reinforcement learning. Technical Report CMU-CS-03-107, School of Computer Science, Carnegie Mellon University, 2003.

[273] H. Van Dyke Parunak. Applications of distributed artificial intelligence in industry. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed AI*. John Wiley & Sons, 1996.

[274] L. Z. Varga, N. R. Jennings, and D. Cockburn. Integrating intelligent systems into a cooperating community for electricity distribution management. *International Journal of Expert Systems with Applications*, 7(4):563–579, 1994.

[275] J. Vidal and E. Durfee. Agents learning about agents: A framework and analysis. In *Working Notes of AAAI-97 Workshop on Multiagent Learning*, 1997.

[276] J. Vidal and E. Durfee. The moving target function problem in multiagent learning. In *Proceedings of the Third Annual Conference on Multi-Agent Systems*, 1998.

[277] J. Vidal and E. Durfee. Predicting the expected behavior of agents that learn about agents: the CLRI framework. *Autonomous Agents and Multi-Agent Systems*, January 2003.

[278] K. Wagner. Cooperative strategies and the evolution of communication. *Artificial Life*, 6(2):149–179, Spring 2000.

[279] X. Wang and T. Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems (NIPS-2002)*, 2002.

[280] R. Watson and J. Pollack. Coevolutionary dynamics in a minimal substrate. In E. Cantu-Paz *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2001.

[281] R. Weihmayer and H. Velthuijsen. Application of distributed AI and cooperative problem solving to telecommunications. In J. Liebowitz and D. Prereau, editors, *AI Approaches to Telecommunications and Network Management*. IOS Press, 1994.

[282] M. Weinberg and J. Rosenschein. Best-response multiagent learning in non-stationary environments. In *AAMAS-2004 — Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[283] G. Weiß. Some studies in distributed machine learning and organizational design. Technical Report FKI-189-94, Institut für Informatik, TU München, 1994.

[284] G. Weiß. *Distributed Machine Learning*. Sankt Augustin: Infix Verlag, 1995.

[285] G. Weiß, editor. *Distributed Artificial Intelligence Meets Machine Learning : Learning in Multi-Agent Environments*. Number 1221 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1997.

[286] G. Weiß. Special issue on learning in distributed artificial intelligence systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3), 1998.

[287] G. Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[288] G. Weiß and P. Dillenbourg. What is 'multi' in multi-agent learning? In P. Dillenbourg, editor, *Collaborative learning. Cognitive and computational approaches*, pages 64–80. Pergamon Press, 1999.

[289] G. Weiß and S. Sen, editors. *Adaptation and Learning in Multiagent Systems*. Lecture Notes in Artificial Intelligence, Volume 1042. Springer-Verlag, 1996.

[290] M. Wellman and J. Hu. Conjectural equilibrium in multiagent learning. *Machine Learning*, 33(2-3):179–200, 1998.

[291] J. Werfel, M. Mitchell, and J. P. Crutchfield. Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, 4(4):388, November 2000.

[292] B. B. Werger and M. Mataric. Exploiting embodiment in multi-robot teams. Technical Report IRIS-99-378, University of Southern California, Institute for Robotics and Intelligent Systems, 1999.

[293] G. M. Werner and M. G. Dyer. Evolution of herding behavior in artificial animals. In *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, 1993.

[294] T. White, B. Pagurek, and F. Oppacher. ASGA: Improving the ant system by integration with genetic algorithms. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 610–617, University of Wisconsin, Madison, Wisconsin, USA, 22-25 1998. Morgan Kaufmann.

[295] S. Whiteson and P. Stone. Concurrent layered learning. In *AAMAS-2003 — Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[296] R. P. Wiegand. *Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, Department of Computer Science, George Mason University, 2003.

[297] R. P. Wiegand, W. Liles, and K. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In E. Cantu-Paz *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1235–1242, 2001.

[298] R. P. Wiegand, W. Liles, and K. De Jong. Analyzing cooperative coevolution with evolutionary game theory. In D. Fogel, editor, *Proceedings of Congress on Evolutionary Computation (CEC-02)*, pages 1600–1605. IEEE Press, 2002.

[299] R. P. Wiegand, W. Liles, and K. De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In Poli et al. [201], pages 231–248.

[300] R. P. Wiegand and J. Sarma. Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In *Parallel Problem Solving from Nature — PPSN-2004*. Springer, 2004.

[301] M. Wiering, R. Salustowicz, and J. Schmidhuber. Reinforcement learning soccer teams with incomplete world models. *Journal of Autonomous Robots*, 1999.

[302] A. Williams. Learning to share meaning in a multi-agent system. *Autonomous Agents and Multi-Agent Systems*, 8:165–193, 2004.

[303] E. Wilson. *Sociobiology: The New Synthesis*. Belknap Press, 1975.

[304] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[305] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems-11*, pages 952–958, Denver, 1998.

[306] D. H. Wolpert, K. R. Wheller, and K. Tumer. General principles of learning-based multi-agent systems. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 77–83, Seattle, WA, USA, 1999. ACM Press.

[307] M. Wooldridge, S. Bussmann, and M. Klosterberg. Production sequencing as negotiation. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96),*, 1996.

[308] A. Wu, A. Schultz, and A. Agah. Evolving control for distributed micro air vehicles. In *IEEE Computational Intelligence in Robotics and Automation Engineers Conference*, 1999.

[309] H. Yanco and L. Stein. An adaptive communication protocol for cooperating mobile robots. In *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, pages 478–485, 1993.

[310] N. Zaera, D. Cliff, and J. Bruten. (not) evolving collective behaviours in synthetic fish. Technical Report HPL-96-04, Hewlett-Packard Laboratories, 1996.

[311] B. Zhang and D. Cho. Coevolutionary fitness switching: Learning complex collective behaviors using genetic programming. In *Advances in Genetic Programming III*, pages 425–445. MIT Press, 1998.

[312] J. Zhao and J. Schmidhuber. Incremental self-improvement for life-time multi-agent reinforcement learning. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4*, pages 516–525, Cape Code, USA, 9-13 1996. MIT Press. ISBN 0-262-63178-4.