Multiagent Learning through Neuroevolution

Risto Miikkulainen, Eliana Feasley, Leif Johnson, Igor Karpov, Padmini Rajagopalan, Aditya Rawal, and Wesley Tansey

Department of Computer Science
The University of Texas at Austin, Austin, TX 78712, USA
{risto,elie,leif,ikarpov,padmini,aditya,tansey}@cs.utexas.edu

Abstract. Neuroevolution is a promising approach for constructing intelligent agents in many complex tasks such as games, robotics, and decision making. It is also well suited for evolving team behavior for many multiagent tasks. However, new challenges and opportunities emerge in such tasks, including facilitating cooperation through reward sharing and communication, accelerating evolution through social learning, and measuring how good the resulting solutions are. This paper reviews recent progress in these three areas, and suggests avenues for future work.

Keywords: Neuroevolution, neural networks, intelligent agents, games.

1 Introduction

Neuroevolution, i.e. evolution of artificial neural networks, has recently emerged as a powerful approach to constructing complex behaviors for intelligent agents [11,24]. Such networks can take a number of simulated or real sensor values as input, and perform a nonlinear mapping to outputs that represent actions in the world such as moving around, picking up objects, using a tool or firing a weapon. Recurrency in neural networks allow then to integrate information over time, and make decisions robustly even in partially observable domains where traditional value-function based reinforcement learning techniques [42] have difficulty. Neuroevolution has thus been useful in building intelligent agents for e.g. video games, board games, mobile robots, and autonomous vehicles [40,13,45,12,21,16,23].

Much of the work so far has focused on developing intelligent behaviors for single agents in a complex environment. As such behaviors have become more successful, a need for principled multiagent interactions has also risen. In many domains such as video games and robotics, there are actually several agents that work together to achieve a goal. A major part of being effective in such domains is to evolve principled mechanisms for interacting with other agents. Neuroevolution is a natural approach to multiagent systems as well: The evolving population provides a natural team setting, and neural networks allow implementing team sensing and interactions in a natural manner.

It turns out the multiagent perspective brings entirely new challenges and opportunities to neuroevolution research. This paper reviews recent progress in three of them: Setting up evolution so that effective collaboration emerges, combining evolution with learning within the team, and evaluating the team behaviors quantitatively.

First, how should evolution be set up to promote effective team behaviors. That is, when the team is successful, should the rewards be distributed among team members equally, or should individuals be rewarded for their own performance? Should the team members communicate explicitly to coordinate their behavior, or is it sufficient to rely on changes in the environment (i.e. stigmergy)? How much should collaboration be rewarded for it to emerge over simpler individual behaviors? Experiments illustrating these issues will be reviewed in section 2.1.

Second, being part of a team provides an opportunity not only for coordinating actions of several team members, but also of learning from one another in the team. How should just learning be best established? Should the population champion be used as a teacher, or is it better to learn from any successful behavior in the population, in an egalitarian fashion? If everyone is learning based on the same successful behaviors, how can diversity be maintained in the population? Is learning useful in driving evolution through the Baldwin effect, or is it more effective to encode the learned behaviors directly to the genome through Lamarckian evolution? Section 3 evaluates possible solutions to these issues.

Third, given that multiagent behaviors can be particularly complex, depending on interactions between the team members, the environment, and opponents, how can they be best characterized and evaluated? For instance in a competitive environment, can a tournament be set up to evaluate the strengths of teams quantitatively? Is there a single best behavior or are multiple roughly equally good different solutions possible? Are best behaviors shared by everyone on the team, or is it better to have different specialties, or even on-line adaptation? These issues are discussed in the context of a comprehensive NERO tournament in section 4.

2 Setting up Multiagent Neuroevolution

As described below in separate sections, prey capture by a team of predators is used as the experimental domain to study how reward structure and amount and coordination mechanism affect multiagent evolution. An advanced neuroevolution method of multi-component-ESP will be used to evolve the controller neural networks.

2.1 Predator-Prey Environment

A significant body of work exists on computational modeling of cooperation in nature. For instance, flocking behaviors of birds and schooling of fish have been modeled extensively using rule-based approaches [6,31,37]. Cooperative behavior of micro-organisms like bacteria and viruses has been modeled with genetic algorithms [22,33]. Ant and bee colonies have been the subject of many studies involving evolutionary computation as well [9,29,47]. Similarly, as a research setting to study how cooperation can best emerge in multiagent neuroevolution, predator-prey simulation environment was constructed to model hunting behaviors of hyenas. This environment provides immediate motivation and insight from nature; it is also easy to simulate with quantifiable results.

In this environment, a team of predators (hyenas) is evolved using cooperative coevolution to capture fixed-behavior prey (a gazelle or a zebra). The world in this simulation is a discrete toroidal environment with 100×100 grid locations without obstacles, where the prey and predators can move in four directions: east, west, north and south. They move one step at a time, and all the agents take a step simultaneously. To move diagonally, an agent has to take two steps (one in the east-west direction and one in the north-south direction). A predator is said to have caught a prey if it moves into the same location in the world as the prey. The predators are aware of prey positions and the prey are aware of predator positions. Direct communication among predators (in terms of knowledge of other predators' positions) is also introduced in some cases. In all other cases, the predator agents can sense only prey movements and have to use that to coordinate their actions (stigmergic communication). There is no direct communication among the prey. Each predator has as its inputs the x and y offsets of all the prey from that predator. In the case of communicating predators, they also get as input the x and y offsets to the other predators. When fitness rewards from prey capture are shared, all the predators gain fitness even when only one of them actually catches the prey. In cases with individual fitness, only the particular predator that captures the prey gets the reward.

There are two types of prey in the environment - a smaller prey (gazelle) that moves with 0.75 times the speed of the predator and a larger prey (zebra) that has the same speed as the predator. The prey behaviors in these experiments are hard-coded and do not evolve. Each prey simply moves directly away from the current nearest predator. The predators can therefore catch the smaller prey individually, but cannot catch the larger prey by just following the prey around, because their grid world is toroidal. The predators have to surround a zebra from different directions before they can catch it. In cases where both types of prey exist in the field simultaneously, the predators need to decide whether to catch the small prey individually or to coordinate and hunt the larger prey together. The larger prey give more reward than the smaller prey, and the relative reward amounts can be varied.

Thus, three parameters are progressively modified in these experiments: (1) whether only the individual actually catching the prey receives the fitness, or whether it is shared by all individuals, (2) whether the predators can observe one another or not (direct vs. stigmergic communication), and (3) the size of the fitness reward from catching a prey. These experiments are used to contrast the role of each of these parameters in the evolution of cooperation.

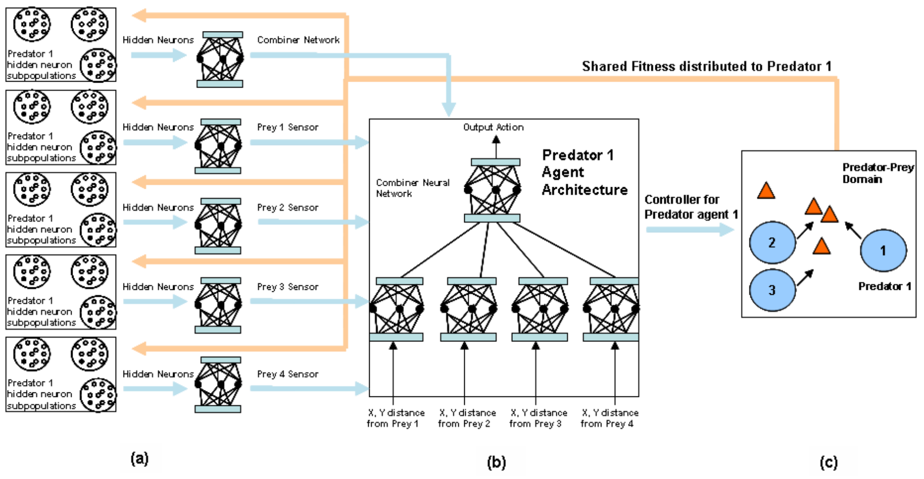
Predator 1 hidden neuron subpopulations
Hidden Neurons
Combiner Network
Prey 1 Sensor
Prey 2 Sensor
Prey 3 Sensor
Prey 4 Sensor
Shared Fitness distributed to Predator 1
Output Action
Combiner Neural Network
Predator 1 Agent Architecture
Controller for Predator agent 1
Predator-Prey Domain
Predator 1
X, Y distance from Prey 1
X, Y distance from Prey 2
X, Y distance from Prey 3
X, Y distance from Prey 4
(a)
(b)
(c)

Fig. 1. Multi-Component ESP in the predator-prey domain for predator agent in Experiment 1. A single predator agent (shown in (b)) is composed of five neural networks. Four of these sense one of the prey agents. Their outputs are given to a fifth combiner network that outputs the next move for that predator. Each network is evolved in a separate ESP process, where one subpopulation is evolved for each of the neurons in the network (a). The predator is evaluated in the domain simulation with prey and other predator agents (c). Its fitness is distributed equally among all the networks and among all the neurons that participated in it. In this manner, evolution can discover neurons and networks that cooperate well to form an effective agent.

2.2 The Multi-Component ESP Neuroevolution Method

Coevolution is defined as the simultaneous evolution of two or more individuals whose fitness is measured based on their interactions with each other [25]. In cooperative coevolution, the individuals have to evolve to cooperate to perform a task. They share the rewards and punishments of their individual actions equally. It turns out that it is often easier to coevolve components that cooperate to form a solution, rather than evolve the complete solution directly [15,26]. The components will thus evolve different roles in the cooperative task.

For example, in the Enforced SubPopulations (ESP) architecture [15], neurons selected from different subpopulations are required to form a neural network whose fitness is then shared equally among them. Such an approach breaks a complex task into easier sub-tasks, avoids competing conventions among the component neurons and makes the search space smaller. These effects make neuroevolution faster and more efficient.

Similarly, Multi-Component ESP extends this approach to evolve a team of agents (Figure 1). Each agent comprises multiple ESP-type neural networks to sense different objects in the environment. The team's reward from fitness evaluations is shared equally by the component networks of all the agents [30]. The cooperative coevolution approach has been shown to be effective when

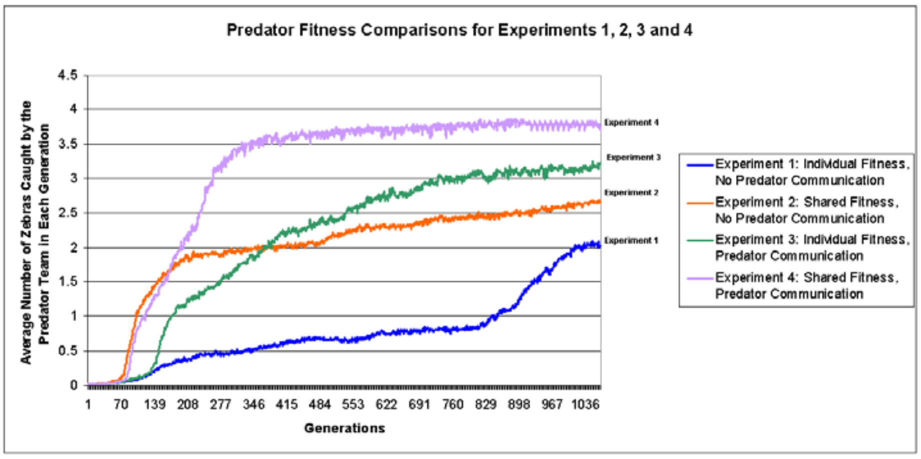

Fig. 2. Average number of prey (zebras) caught (out of four possible) in Experiments 1, 2, 3 and 4. The total number of prey caught by the three predators was averaged over 6000 trials for each generation. Cooperation is slow to evolve with individual rewards and without communication, and is less efficient (Experiment 1). Introduction of reward sharing results in faster and more effective evolution of cooperation (Experiment 2). Knowledge of positions of other predators makes it easier to evolve coordinated hunting strategies (Experiment 3). Evolution of cooperation is strongest when reward sharing and communication are combined (Experiment 4).

coevolving teams of agents. First, Yong and Miikkulainen [51] showed that a team of predators that share fitness can evolve to cooperate to catch prey with or without communication. In their experiments, without communication, the roles the predators evolve are more rigid but more effective; with communication, their roles are less efficient but more flexible. Second, Rawal et al. [30] showed that the Multi-Component ESP architecture can coevolve a team of predators with a team of prey. The individuals cooperate within the team, but the predator team competes with the prey team. Therefore, the Multi-Component ESP architecture will be used to evolve the predators in this paper as well.

In prior work, the outputs of the neural networks within a predator or prey agent were summed to get the final output action. However, preliminary experiments showed that including a combiner network to combine the outputs of these networks was more powerful and resulted in the emergence of more complex behaviors. Hence, this technique was used in this paper (Figure 1). The combiner network weights were evolved using the same technique as the other networks.

2.3 Experimental Setting Results

In the control experiment (Experiment 1), the predators neither communicate nor share fitness. Cooperation does not evolve initially and as a result, they

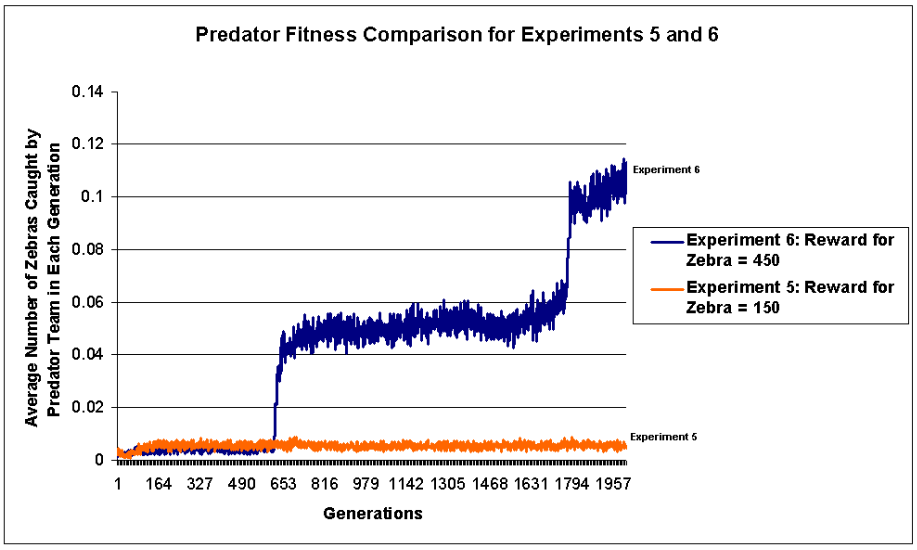

Fig. 3. Average number of zebras caught in Experiments 5 and 6. The total number of prey (out of one possible) caught by the three predators was averaged over 6000 trials for each generation. When the payoff on capturing a zebra is low with respect to the difficulty of catching it (Experiment 5), the predators prefer to hunt the easy-to-catch gazelles individually. When the net return for capturing the zebra is high enough (Experiment 6), the predators evolve to discover cooperative strategies to hunt it. Once it is caught, they continue by hunting gazelles.

rarely catch any zebras. On the other hand, adding reward sharing (Experiment 2) increases the number of prey caught as the predators efficiently evolve to cooperate over the early generations. The average number of zebras caught in each generation in Experiments 1 and 2 are contrasted in Figure 2.

Similarly, adding communication to predators with individual fitness in Experiment 3 results in the predators easily evolving to cooperate, leading to more prey captures (Figure 2). This effect is even stronger with both communication and fitness sharing enabled (Experiment 4; Figure 2), suggesting that these two factors affect different aspects of the evolution process, i.e. how easy it is to establish cooperation, and how worthwhile it is.

Experiments 5 and 6 were designed to answer the question: If there are both gazelles, which can be caught easily but give a lower fitness, and zebras, which need all the predators to cooperate to catch them but give higher fitness, which one is preferred? In Experiment 5, the predators prefer to hunt gazelles instead of evolving to cooperate to capture the zebra. The reward for catching the zebra is not large enough for cooperative behaviors to be selected during evolution. In contrast, in Experiment 6, it is large enough, and the predators slowly evolve to team up to capture this more difficult prey, thus verifying the hypothesis that net return is important in the evolution of cooperation (Figure 3). Interestingly, they are still able to hunt gazelles as well, but only do it when there are no zebras around even though zebras are still hard to catch. This result is important because it suggests that cooperative strategies include individual strategies as a special case.

2.4 Experimental Setting Conclusions

The experiments confirmed that predator coordination mechanism, reward structure, and net return on prey capture are important factors in the evolution of efficient cooperative hunting behaviors. When hyenas survive on gazelles, they do not need to cooperate. However, if the zebras are available and tasty enough, they will. These results are intuitive, but this is the first time easily replicable experiments were constructed to verify them. The same factors that were established to be important in the evolution of cooperation in this domain can be manipulated in more complex artificial environments to build interesting behaviors for other intelligent agents in the future.

3 Combining Evolution with Social Learning

After a brief motivation for social learning in multiagent neuroevolution, the robot foraging domain and the NEAT neuroevolution method are briefly described, followed by results answering the questions posed in section 1.

3.1 Motivation for Social Learning

Evolutionary algorithms (EAs) [14] evaluate agents either in isolation or in direct competition with a subset of the other members of the population. Social and cultural learning algorithms [32] extend EAs by enabling agents to leverage observations of other members of the population to improve their own performance during their lifetime. By learning from others without having to directly experience or acquire knowledge, social learning algorithms have been able to improve the learning rate of EAs in many challenging domains [8,17,46,1,7,48].

Traditionally in social learning algorithms, each agent is either a student or a teacher [28,2]. All actions of the teacher agents are considered to be good examples from which to learn, as they are derived from a high-fitness strategy (i.e. the teacher's policy). However, an agent with high overall fitness may not always choose good actions and agents with low overall fitness may actually perform well in some limited scenarios. Filtering potential observations based on their own merit may therefore be more appropriate and lead to both improved learning rate and stronger final strategies.

This paper presents Egalitarian Social Learning (ESL) as an alternative to the student-teacher paradigm. Agents in ESL are divided into subcultures at the start of each generation and can learn from any other agent in their subcultural group. Learning examples are determined by a user-defined acceptability function that filters out examples leading to low rewards. When an action is accepted, agents mimic it in order to learn a policy similar to that of the observed agent. ESL differs from other social learning algorithms in that the quality of a training example is measured by the reward received rather than the fitness of the agent generating the example.

3.2 The Foraging Domain

The domain used to evaluate ESL is a foraging world in which agents move freely on a continuous toroidal surface. The world is populated with various plants, some of which are nutritious and bear positive reward, while others are poisonous and bear negative reward. These plants are randomly distributed over the surface of the world. The foraging domain is non-competitive and non-cooperative; each agent acts independently of all other agents, with the exception of the teaching signals that pass between them. At the start of each generation, all individuals begin at the center of the world, oriented in the same direction, and confronted with the same plant layout and configuration. Every agent then has a fixed number time steps to move about the surface of the world eating plants— which happens automatically when an agent draws sufficiently close to one— before the evaluation is over.

Agents “see” plants within a 180° horizon via a collection discretized sensors. Each agent has eight sensors for each type of plant, with each sensor covering a different 12.5° sector of the 180° ahead of the agent. Agents cannot see other individuals or plants they have already eaten— all they can see is edible food. The strength of the signal generated by each plant is proportional to its proximity to the agent. Agents also have a sensor by which they can detect their current velocity. As agents can only turn up to 30° in a given timestep, knowledge of velocity is necessary for agents to accurately plan optimal trajectories (e.g. agents may need to slow down in order to avoid overshooting a plant). Each agent is controlled by an artificial neural network that maps from the agent’s sensor readings to the desired change in orientation and velocity.

Two separate configurations of the robot foraging world are used in the experiments. The first two experiments use a “simple” world where the toroidal surface is 2000 by 2000 units, with a single plant type of value 100 and 50 randomly distributed instances of the plant. In this world, the agents have a straightforward task of learning to navigate efficiently and gather as many plants as possible. The third set of experiments uses both the simple world and a second, more complex world to evaluate performance. The “complex” world has a surface of 500 by 500 units, with five different plant types of value -100, -50, 0, 50, and 100. For each plant type, 20 instances are created and randomly distributed across the surface. This world presents the agents with a more difficult task as they must efficiently gather nutritious food while simultaneously avoiding the poisonous food.

In all four experiments, 100 different agents are created in each generation. All networks are initialized with fully-connected weights with no hidden neurons and a learning rate of 0.1 is used when performing backpropagation. Agents automatically eat any plant within five units. Each evaluation lasts 1000 timesteps and the results for each experiment are the average of 30 independent runs. The acceptability function for all experiments is to learn from any action yielding a positive reward.

3.3 The NEAT Neuroevolution Method

NeuroEvolution of Augmenting Topologies (NEAT)[39] is an evolutionary algorithm that generates recurrent neural networks. Through a process of adding and removing nodes and changing weights, NEAT evolves genomes that unfold into networks. In every generation, those networks with the highest fitness reproduce, while those with the lowest fitness are unlikely to do so. NEAT maintains genetic diversity through speciation and encourages innovation through explicit fitness sharing.

In the foraging domain, NEAT is used to generate a population of individual neural networks that control agents in the world. The input to each network is the agent's sensors, and the outputs control the agent's velocity and orientation. The fitness of each network is determined by the success of the agent it controls—over the course of a generation, networks that control agents who eat a good deal of rewarding food and very little poison will have high fitness and those that control agents with less wise dietary habits will have low fitness.

In standard NEAT, the networks that are created do not change within one generation. To facilitate social learning, we must perform backpropagation [34] on the networks that NEAT creates in order to train agents on accepted examples. Since NEAT networks are recurrent, ESL enhances NEAT with backpropagation capabilities using the backpropagation through time algorithm [49].

The final fitness of each phenome, then, reflects the performance of the individual that used that phenome and elaborated on it over the course of a generation. This elaboration drives evolution in two alternate ways. In Darwinian evolution, the changes that were made to the phenome only affect selection and are not saved; in Lamarckian, the genome itself is modified.

3.4 Social Learning Results

Three experiments were performed: ESL was first applied to the entire population (without subcultures), and the best way to make use of learning (Darwinian vs. Lamarckian) determined. The effect of maintaining diversity through explicit subcultures was then evaluated. In the third experiment, ESL was compared to the traditional student-teacher model of social learning.

Figure 4 shows the results of applying a monocultural egalitarian social learning algorithm to the foraging domain in both the Lamarckian and Darwinian paradigms. The performance of both algorithms quickly converges, with Lamarckian reaching a higher-fitness solution than Darwinian evolution. In the context of *on-line* evolutionary learning algorithms, previous work [50] showed that Darwinian evolution is likely to be preferable to Lamarckian evolution in dynamic environments where adaptation is essential and the Baldwin effect [38] may be advantageous. However, as adaptation is not necessary for foraging agents

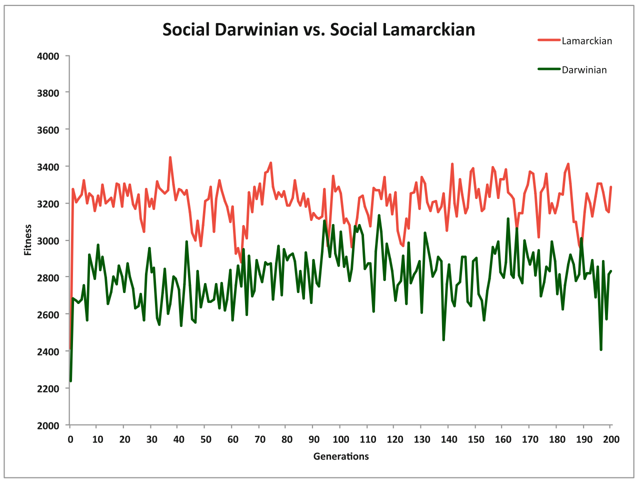

Fig. 4. The effects of Darwinian and Lamarckian evolution when using a monocultural variant of ESL. While both evolutionary paradigms converge rapidly Lamarckian evolution is more effective than Darwinian in the foraging domain. Consequently, Lamarckian evolution is the paradigm used in all remaining experiments.

(i.e. the rewards of each plant type are the same in every generation), in this experiment Lamarckian evolution outperforms Darwinian evolution. Nevertheless, in both cases performance converges to a lower score than that of simple neuroevolution.

On the other hand, monocultural Lamarckian social learning is likely to provide redundant information that may result in getting stuck in local optima. In order to address this problem, subcultural version of egalitarian social learning was designed to promote and protect diversity. At the start of each generation, the population is divided into 10 subcultures of 10 agents each, with each agent's subculture decided at random. During the evaluation, agents only teach and learn from other agents in their own subculture.

Figure 5 shows results comparing monocultural and subcultural learning. Subcultural learning not only reaches a higher peak than the monocultural method, but also arrives at this level of fitness more rapidly than the simple neuroevolution approach. When every mutated organism has the opportunity to train every other, as is the case in monocultural learning, the entire population may be negatively impacted by any one individual. By preventing agents that lead the population towards local optima from impacting the remainder of the population, subcultural learning provides safety and protection from premature convergence.

In the third set of experiments, subcultural ESL is compared to an on-line student-teacher learning algorithm inspired by the NEW TIES system [17].

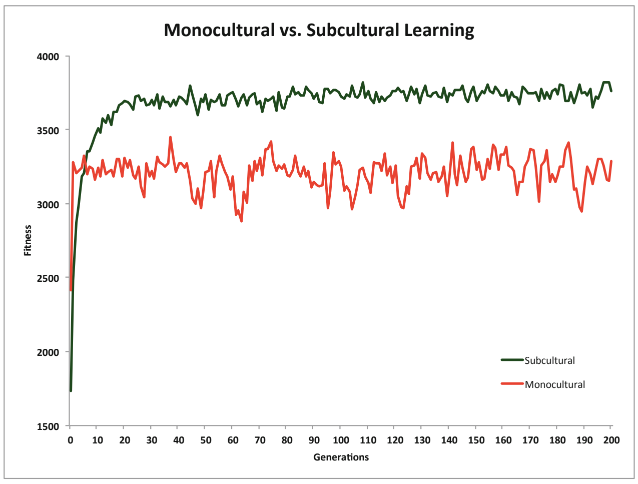

Fig. 5. Monocultural agents learning from the entire population and subcultural agents learning only from their subcultures. Subcultural agents outperform monocultural agents, converging to a much higher ultimate fitness.

The system utilizes a steady-state evolution in which at every timestep each agent probabilistically teaches the lowest-fitness member of the population within some radius, effectively forming geographical subcultures.

Figures 6 and 7 show the results of the subcultural ESL algorithm compared to the student-teacher variant of NEW TIES and simple neuroevolution. Subcultural ESL converges to a near-optimal solution faster than the student-teacher variant in both the simple and the complex world. While in the simple world (Figure 6) this speed-up is slight, in the complex world (Figure 7) the egalitarian approach is more than an order of magnitude faster, reaching a higher fitness by generation 50 than either the student-teacher or simple neuroevolution methods achieve by generation 500.

3.5 Social Learning Conclusions

Unlike traditional social learning algorithms that follow a student-teacher model, ESL teaches agents based on acceptable actions taken by any agent in its subculture. By constraining teaching samples to those from the same subcultural group, ESL promotes diversity in the overall population and prevents premature convergence. Experiments in a complex robot foraging domain demonstrated that this approach is highly effective at quickly learning a near-optimal policy with Lamarckian evolution. The results thus suggest that egalitarian social learning is a strong technique for taking advantage of team behaviors that exist in the evolving population.

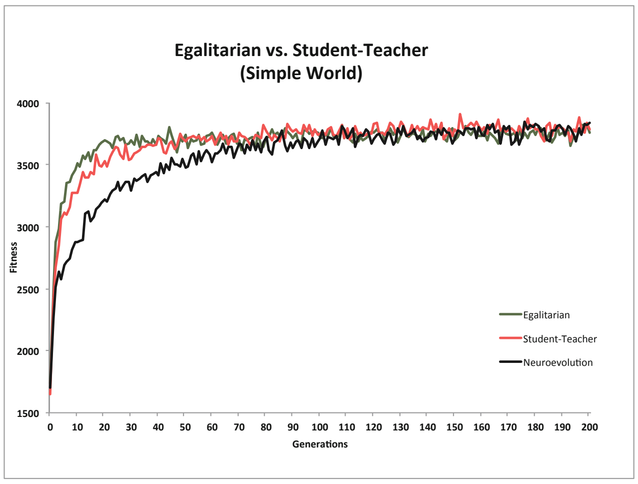

Fig. 6. ESL compared to simple neuroevolution, and student-teacher learning in the simple world. All strategies converge to solutions of similar quality, with egalitarian learning converging in the fewest evaluations.

4 Evaluating Multiagent Performance

4.1 Motivation

The NERO video game [40] was originally developed to demonstrate that neuroevolution could be a powerful tool for constructing solutions to open-ended design problems. A human player provides increasingly challenging goals, and a team of NPCs evolves to meet those goals, eventually excelling in the game. Complex behavior was demonstrated in a number of different challenge situations, such as running a maze, approaching enemy while avoiding fire, and coordinating behavior of small sub-teams. However, the final behavior of entire teams was never evaluated, so it is not clear how complex the behaviors could become in this process and what successful behavior in the game might actually look like. Also, it is not clear whether there is one simple winning strategy that just needs to be refined to do well in the game, or whether there are multiple good approaches; similarly, it is unclear whether winning requires combining individuals with different skills into a single team, or perhaps requires on-line adaptation of team composition or behaviors.

In any case, such evaluations are difficult for two reasons: (1) designing teams takes significant human effort, and covering much of the design space requires that many different designers participate; (2) evaluation of the resulting behaviors takes significant computational effort, and it is not clear how it can be best spent. This paper solves the first problem by crowd-sourcing, i.e. running a NERO tournament online. Students in the 2011 Stanford online AI course[1]

[1] `www.ai-class.com`

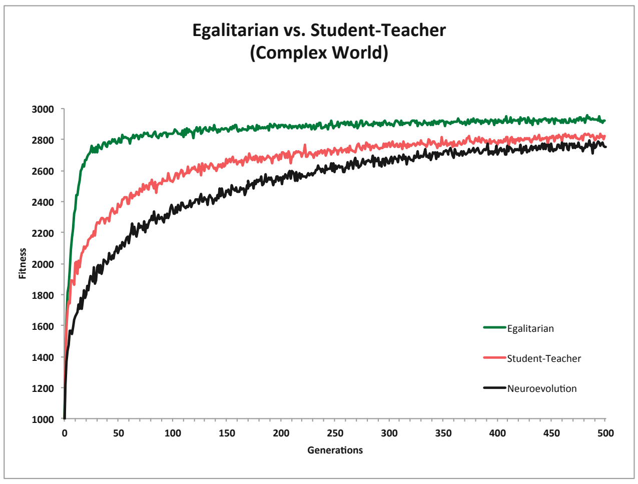

Fig. 7. ESL compared to simple neuroevolution, and student-teacher learning in the complex world. ESL is more than an order of magnitude faster, reaching a higher fitness by generation 50 than either comparison method achieves by generation 500.

were invited to participate. About 85 of them did, many spending considerable effort to produce good teams, thereby resulting in a wide selection of approaches and solutions. The second problem was solved by first testing out different tournament structures, and eventually running a comprehensive round robin tournament of 24,180 games in parallel in a Condor cluster. The results from the tournament were then used to identify complex and interesting behaviors that perform well on the task.

4.2 The NERO Domain

NERO [41] was originally developed as an experimental platform for training teams of agents to accomplish complex tasks based on the rtNEAT [39] method for evolving artificial neural networks. The rtNEAT method is a version of NEAT described in the previous section, with the difference that individuals are evaluated, reproduced, and replaced continuously instead of in generations. This approach allows running evolution in the background continuously in real-time without distracting the human player. The original NERO game was later extended into an open-source version called OpenNERO,[2] which is a general-purpose platform for AI research and education [20]. OpenNERO includes several different environments and AI methods in addition to the NERO game environment itself, but only the NERO environment in OpenNERO was used in this research.

Each NERO agent on a team has a fixed array of 15 sensors that detect agents on the same and opposite teams, placement of nearby walls, distance to

[2] `opennero.googlecode.com`

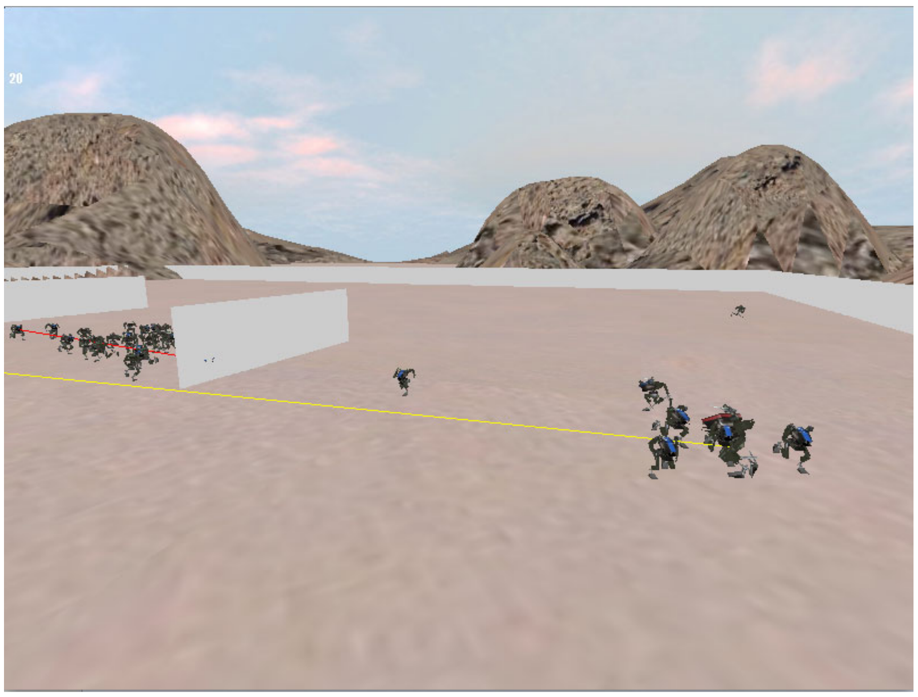

Fig. 8. A screenshot of a single NERO match. Two teams of agents are shown as bipedal robots in a playing arena with obstacles and boundaries. The teams start opposite each other on the two sides of the obstacle wall in the middle and have to get around this obstacle to damage opponents and earn points.

a flag (if present), current motion, damage to opponents, and damage to the agent itself. Agents control their movement on the field using a two-dimensional control signal $u =< \ddot{r}, \ddot{\theta} >$, where $\ddot{r}$ is the linear acceleration of the agent in the direction of the agent's current orientation θ, and $\ddot{\theta}$ is the agent's angular acceleration.

Training teams in OpenNERO is similar to NERO. The user can dynamically change the virtual environment by adding, scaling, rotating or removing walls, moving a flag, and adding or removing immobile enemy agents. The user can also change the way the fitness function is computed by adjusting a (positive or negative) weight on each of the different available fitness dimensions. The available fitness dimensions are *stand ground* (i.e. minimize $\dot{r}$), *stick together* (minimize distance to the team's center of mass), *approach flag* (minimize distance to a flag on the field, if present), *approach enemy* (minimize distance to the closest enemy agent), *hit target* (successfully fire at an enemy), and *avoid fire* (minimize accrued damage).

For the battle task, two teams—each consisting of 50 NERO agents—occupy a continuous, two-dimensional, virtual playing field of fixed size (see Figure 8).

The playing field contains one central obstacle (a wall), four peripheral obstacles (trees), and four walls around the perimeter to contain all agents in the same general area. Each NERO agent starts a battle with 20 hit-points. At each time slice of the simulation, each agent has the opportunity to fire a virtual laser at the closest target on the opponent's team that is within two degrees of the agent's current orientation. If an agent fires and hits an opponent, the opponent loses one hit-point. The score for a team is equal to the number of hit-points that the opponent team loses in the course of the battle.

A team of NERO agents can be serialized to a flat text file. The text file describes each of the 50 agents on a team. Agents that use rtNEAT serialize to a description of the genotype for each agent, and agents that use Q-learning serialize their (hashed) Q-tables directly to the file. Anyone was allowed to participate in the tournament by submitting online a serialized team of virtual agent controllers for the NERO battle task. The only difference between the teams was in the training of the controllers contributed by the competitors.

The OpenNERO code was extended for this tournament to allow teams to consist of mixtures of rtNEAT (neural network) and reinforcement learning (Q-learning) agents; this distinction is primarily interesting in the sense that rtNEAT agents search for control policies directly, while Q-learning searches in value-function space and then uses value estimates for each state to determine appropriate actions. For rtNEAT–based training, individuals within the population are ranked based on the weighted sum of the Z-scores over the fitness components. For Q-learning–based training, each fitness dimension is scaled to $[0, 1]$, and then a linear weighted sum is used to assign a total reward to each individual.

Both types of controllers could be submitted to the online tournament: artificial neural network controllers of arbitrary weight and topology, and hash tables approximating the value function of game states. The competitors could extend and/or modify the available OpenNERO training methods as well as create their own training environments and regimens. It was this training that determined the fitness of each team when pitted against other teams submitted to the tournament.

4.3 Evaluation Results

An online NERO tournament was run in December 2011. About 85 participants submitted 156 teams to the tournament. Of these, 150 teams contained neural network-controlled agents and 11 contained value table-controlled agents. Mixed teams were also allowed; four of the submitted teams contained mixed agent types. Because of the large number of teams, each game was played off-screen and limited to 5 minutes of game time. (In practice, good teams were able to eliminate all opponents in less than 5 minutes.) The team with the highest number of remaining hit points was declared the winner at the end of the match. Ties were extremely rare and were broken by a pseudo-random coin toss. The match-making script allowed matches to be run in parallel on a single machine or to be distributed to a Condor compute cluster [43].

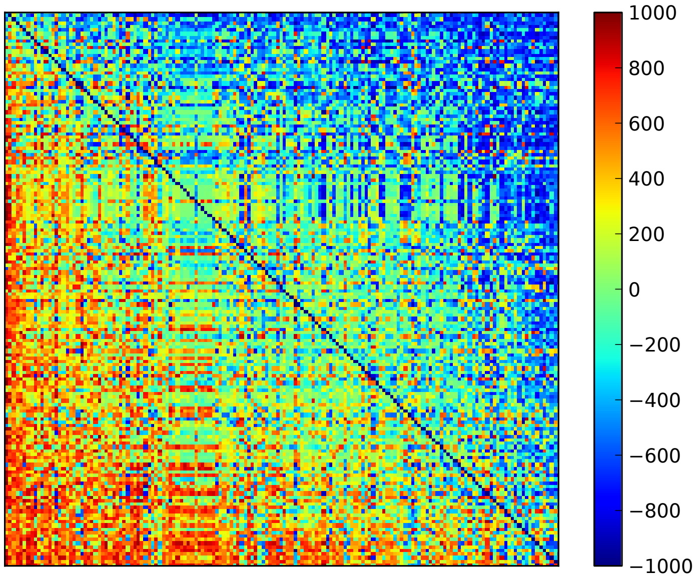

Fig. 9. Results from the round-robin NERO tournament. Teams are sorted by average score differential over all matches. Rows and columns in the matrix represent teams in the tournament, and colors represent score differentials for the respective individual matches between two teams. Red indicates victory by the row team, and blue indicates victory by the column team.

First, several double-elimination tournaments were run with the submitted teams. Repeated runs of the double-elimination tournament revealed that while the set of teams in the top 10 was generally consistent, their ranking was not stable. The teams were then placed in a round-robin tournament to evaluate the overall winner more accurately. In the round-robin tournament, each of the 156 submitted teams was matched against the 155 other teams. Each pair of teams was matched up twice (i.e. $k = 2$), allowing each team to play once as the blue team and once as the red team. This resulted in 24180 separate games, which were processed in parallel on approximately 100 computing nodes, which allowed the entire round-robin tournament to complete in less than 24 hours.

Figure 9 shows the complete results of the round-robin tournament. Black squares along the diagonal represent matches that were not played, blue squares indicate a win by the column team, and red squares indicate a win by the row team. One group of near-duplicate teams was submitted to the tournament; this shows up as the band of similar-colored games about one-third of the way through the matrix. The teams in the Figure are enumerated on both axes in order of increasing average match score differential.

Table 1. Top 10 teams by number of wins

Rank	Team	Total wins
1	synth.pop	137
2	synth_flag.pop	130
3	lolwutamidoing	126
3	me - Rambo	126
5	PollusPirata	125
6	Cyber-trout	124
7	CirclingBullies	123
8	SneakySnipers	121
8	Tut	121
10	coward1	120

Table 1 shows the top ten teams. Despite the large number of teams, no single competitor emerged that significantly outperformed all others. It is interesting to analyze why.

In NERO, agents can be "shaped" towards more complex behaviors by progressively changing the environment and the fitness function during training. This process can create teams of agents that perform specific tasks during a battle. Given the complexity of the environment and the task, many different strategies can arise in this process, and they can interact with each other in complex ways. Considering this potential complexity, evolved strategies in the tournament turned out to be surprisingly easy to analyze. Because fitness is evaluated similarly for each team member during training, teams generally consist of agents that perform similar actions in a given world state. In principle, multiple teams can be trained using different shaping strategies, and single agents from those teams then combined into one team by copying the appropriate parts of the serialized team files (as was suggested in the online tournament instructions). However, most teams submitted to the tournament did not (yet) take advantage of this possibility; instead, agents on a single team usually performed similar actions in response to each game state. It was therefore possible to characterize the most common strategies used in the tournament, as outlined below. The example teams and videos of games between then are available at the tournament website[3].

Pack: The most prominent strategy among winning teams was to train agents to move as a group toward the central wall, then follow the wall tightly to go around it, and then proceed towards the opponents on the other side. This strategy shows up in several of the top ten teams, but most notably in `synth.pop` and `me-Rambo`. These teams actively pursued their opponents by forming agents into a "pack" that had a lot of firepower and was therefore able to eliminate opponents effectively.

Backpedaling: A second successful strategy was almost exactly the opposite from the "pack" strategy: From the start, most agents backpedaled away from

[3] `code.google.com/p/opennero/wiki/TournamentResults2011`

the opponents and the central wall, and then took shots against the opponents from afar. Backpedaling preserves the ability of the agents to fire at opponents, while increasing the distance to the opposing team and maximizing view of the field. Backing up was often effective against the "pack" strategy because a team could eliminate the opponents one-by-one as they emerged around the edge of the wall. Examples of this strategy included `EvilCowards` and `SneakySnipers`.

Encircling: Some teams followed a third strategy, where all agents on the team would run to the far right or left of the wall in a wide arc, and that way try to encircle the enemy agents. Interestingly, although at the outset this strategy seems logical, and was indeed devastating to some opponents, it often lost to the first two strategies. Against the teams that adopted the "pack" strategy, agents following the "encircle" strategy were often not pointed toward their opponents, and thus could be fired upon without recourse. Similarly, teams following the "encircle" strategy tended to fail against the "backpedal" teams because the "encircle" agents again tended to be pointed away from enemy agents too often to fire successfully. Examples of encircling teams include `Caipirinha_01_FNM` and `Artificial Ignorance`.

Brownian Motion: Teams that used reinforcement learning agents tended to cluster in the middle of the playing field and move back and forth in Brownian motion. This behavior likely originated from a difficulty in acquiring sufficient data during training, and from a difficulty in approximating the value function for the task, resulting in agents that effectively chose random actions at each time step. However, sometimes this behavior was seen in rtNEAT teams as well, and it was at times surprisingly effective. When other teams approached the cluster, they were not always lined up to shoot—on the other hand, because of the Brownian motion, the shots were not always on target either. So the Brownian motion teams formed a firing squad that was difficult to approach, performing sometimes better than teams that employed a strategy for going around the wall. Examples of Brownian motion teams include `Peaceful Barbarians 1` and `The729Gang`.

Perhaps most interestingly, the strategies do not form a strict dominance hierarchy, but instead are highly cyclic. For instance, the third-place `me-Rambo` (a "pack" team) reliably defeats the first-place `synth.pop` (also a "pack" team), apparently due to subtle differences in timing. On the other hand, `synth.pop` wins over the 24th-place `EvilCowards` (a "backpedal" team), because the `synth.pop` pack splits into two and breaches the wall from both edges simultaneously. However, `EvilCowards` handily defeats `me-Rambo`, because agents in the `me-Rambo` train are eliminated one-at-a-time as they come around the wall!

There are many other similar cycles in the tournament graph as well, i.e. there is not a team in the tournament that is objectively better than all other teams. It actually seems that there is not even a single strategy that is better than the others: as e.g. the "pack" strategy becomes highly optimized, it also becomes more vulnerable to the "backpedal" counter-strategy. Such relationships may indeed be inherent and even desirable for complex games.

Based on these observations, a compelling next step might be to construct composite teams of individuals from several different teams. The idea is that such teams could perform well against a variety of strategies. Such an approach was already possible in the online tournament, but not extensively used. With more multi-objective evolutionary methods [35,36], it might also be possible to develop multi-modal behaviors that identify what strategy the opponent is using, and select a counter-strategy accordingly. It might also be possible in principle to adapt to opponents online, while the battle is taking place. Such extensions should result in more versatile multi-agent behavior; they will also make it even more difficult to analyze such behavior in the future.

4.4 Evaluation Conclusions

The results of the online NERO tournament demonstrate that multi-agent behavior can be evaluated quantitatively using tournaments. To fully characterize the behaviors, it is necessary to run round-robin tournaments: There may not be a single best strategy, but behaviors may instead be highly diverse, and perform differently against different opponents. This phenomenon may indeed be an inherent property of multiagent behavior in complex domains, and further computational tools may need to be developed to analyze it fully.

5 Discussion and Future Work

The experiments on cooperation raise an interesting issues about the nature of cooperation. For instance, the predators in Experiment 3 (individual rewards with communication) evolve cooperative hunting strategies efficiently, but they do not have any fitness incentive for cooperation. Instead, they use one another to improve individual fitness. Is this real cooperation? In biological literature, a cooperator is defined as an individual who pays a cost for another individual to receive a benefit [27]. This is a useful working definition in artificial settings as well. Thus in Experiment 3, though not all the predators gain by coordinating their behaviors, it is still considered cooperation.

Social learning is strongly motivated by biological analogy as well. The social intelligence hypothesis [5,19] and the cultural intelligence hypothesis [44] suggest that the need to handle complex social behaviors was the primary selection pressure driving the increase in brain size in primates and humans. These hypotheses are indeed supported by strong empirical evidence in recent years [18]. Further, egalitarianist philosophy advocates treating all individuals in a population as equals, regardless of such factors as background and status [3]. In hunter-gatherer societies, egalitarianism is a common paradigm for managing daily activities and organizing social structures [4]. It is likely that this lack of hierarchy and strict maintenance of equality has been pivotal in the development of human society and in separating humans from other primates [10]. It is interesting to see that the same conclusion follows from computational experiments on social learning: Egalitarianism promotes diversity, which in turn allows the population as a whole to achieve better performance. Given how difficult it may be to verify social and computational intelligence hypotheses directly, computational simulations may prove instrumental in testing and refining it further in the future.

Given that the NERO evaluation with 156 teams took significant supercomputing resources, it is useful to evaluate how this approach might be scaled up. Larger tournaments could be organized by using a hybrid structure; round-robin pools could be run in parallel to identify the proper seeds for top-ranking teams, and then a double-elimination tournament could be used to identify the overall winner. Thanks to the independence of individual matches in round-robin tournaments and within each level of a knockout tournament, it should be possible to scale up to even larger tournaments by running games on more compute nodes or carefully designing a tournament structure to optimize use of computing resources.

On the other hand, the tournament also showed that machine-learning games, where neuroevolution of multiagent behavior play a central role, may indeed be a viable game genre in the future. Several approaches to the game were identified in the tournament, none of them dominating all others. This is precisely what makes such games interesting: There is room for innovation and creativity, and the outcomes often turn out to be surprising. Using such games as a platform, it may also be possible to make significant research progress in multi-agent systems and intelligent agents in general.

6 Conclusion

Multiagent systems can be seen as the next frontier in constructing intelligent behavior through neuroevolution. This paper reviewed three challenges and opportunities in such systems: manipulating the rewards, coordination, and return; combining social learning with evolution; and evaluating performance through tournaments. Significant interactions and complexity were observed in each case, leading to the conclusion that the research is still in the beginning stages, but also that the technology is a good match with the opportunities.

Acknowledgments. Special thanks to Marc Wiseman and Kay Holekamp for their help with hyena data, all participants of the OpenNERO online tournament, Sebastian Thrun and Peter Norvig for allowing us to run the tournament in their online AI class, and Ken Stanley and Adam Dziuk for help with OpenNERO. This research was supported in by Google, Inc. and NSF under grants DBI-0939454, IIS-0915038, and IIS-0757479.

References

1. Acerbi, A., Nolfi, S.: Social learning and cultural evolution in embodied and situated agents. In: IEEE Symposium on Artificial Life, ALIFE 2007, pp. 333–340. IEEE (2007)

2. Acerbi, A., Parisi, D.: Cultural transmission between and within generations. Journal of Artificial Societies and Social Simulation 9(1) (2006)
3. Arneson, R.: Egalitarianism. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Stanford University (2009)
4. Boehm, C.: Hierarchy in the forest: The evolution of egalitarian behavior. Harvard Univ. Pr. (2001)
5. Byrne, R., Whiten, A.: Machiavellian Intelligence: Social Expertise and the Evolution of Intellect in Monkeys, Apes, and Humans. Oxford University Press, USA (1989)
6. Czirok, A., Vicsek, T.: Collective behavior of interacting self-propelled particles. Physica A 281, 17–29 (2000)
7. de Oca, M., Stutzle, T., Van den Enden, K., Dorigo, M.: Incremental social learning in particle swarms. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41(2), 368–384 (2011)
8. Denaro, D., Parisi, D.: Cultural evolution in a population of neural networks. In: Marinaro, M., Tagliaferri, R. (eds.) Neural Nets Wirn 1996, pp. 100–111. Springer, Newyork (1996)
9. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics–Part B 26(1), 29–41 (1996)
10. Erda, L., Whiten, A., Mellars, P., Gibson, K.: Egalitarianism and Machiavellian intelligence in human evolution. McDonald Institute for Archaeological Research (1996)
11. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: From architectures to learning. Evolutionary Intelligence 1, 47–62 (2008)
12. Floreano, D., Urzelai, J.: Evolutionary robots with on-line self-organization and behavioral fitness. Neural Networks 13, 431–4434 (2000)
13. Fogel, D.B.: Blondie24: Playing at the Edge of AI. Morgan Kaufmann, San Francisco (2001)
14. Fogel, L., Owens, A., Walsh, M.: Artificial intelligence through simulated evolution. John Wiley (1966)
15. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior, 317–342 (1997)
16. Gomez, F., Miikkulainen, R.: Active guidance for a finless rocket using neuroevolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 2084–2095. Morgan Kaufmann, San Francisco (2003)
17. Haasdijk, E., Vogt, P., Eiben, A.: Social learning in population-based adaptive systems. In: IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence), pp. 1386–1392. IEEE (2008)
18. Herrmann, E., Call, J., Hernández-Lloreda, M., Hare, B., Tomasello, M.: Humans have evolved specialized skills of social cognition: The cultural intelligence hypothesis. Science 317(5843), 1360 (2007)
19. Humphrey, N.: The social function of intellect. Growing Points in Ethology, 303–317 (1976)
20. Karpov, I.V., Sheblak, J., Miikkulainen, R.: OpenNERO: A game platform for AI research and education. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (2008)
21. Kohl, N., Stanley, K.O., Miikkulainen, R., Samples, M., Sherony, R.: Evolving a real-world vehicle warning system. In: Proceedings of the Genetic and Evolutionary Computation Conference (2006)

22. Kubota, N., Shimojima, K., Fukuda, T.: Virus-evolutionary genetic algorithm - coevolution of planar grid model. In: Proceedings of the Fifth IEEE International Conference on Fuzzy Systems (FUZZ IEEE 1996), pp. 8–11 (1996)
23. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. Nature 406, 974–978 (2000)
24. Miikkulainen, R.: Neuroevolution. In: Encyclopedia of Machine Learning. Springer, Berlin (2010)
25. Mitchell, M., Thomure, M.D., Williams, N.L.: The role of space in the success of coevolutionary learning. In: Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems, pp. 118–124 (2006)
26. Moriarty, D.E., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution. Evolutionary Computation 5(4), 373–399 (1997)
27. Nowak, M.A.: Five rules for the evolution of cooperation. Science 314, 1560–1563 (2006)
28. Parisi, D.: Cultural evolution in neural networks. IEEE Expert 12(4), 9–14 (1997)
29. Pérez-Uribe, A., Floreano, D., Keller, L.: Effects of Group Composition and Level of Selection in the Evolution of Cooperation in Artificial Ants. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 128–137. Springer, Heidelberg (2003)
30. Rawal, A., Rajagopalan, P., Miikkulainen, R.: Constructing competitive and cooperative agent behavior using coevolution. In: IEEE Conference on Computational Intelligence and Games (CIG 2010) (August 2010)
31. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. In: Computer Graphics (SIGGRAPH 1987 Conference Proceedings), vol. 21(4), pp. 25–34 (1987)
32. Reynolds, R.: An introduction to cultural algorithms. In: Proceedings of the Third Annual Conference on Evolutionary Programming, pp. 131–139. World Scientific (1994)
33. Roeva, O., Pencheva, T., Tzonkov, S., Arndt, M., Hitzmann, B., Kleist, S., Miksch, G., Friehs, K., Flaschel, E.: Multiple model approach to modelling of escherichia coli fed-batch cultivation extracellular production of bacterial phytase. Electronic Journal of Biotechnology 10(4), 592–603 (2007)
34. Rumelhart, D., Hinton, G., Williams, R.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
35. Schrum, J., Miikkulainen, R.: Evolving agent behavior in multiobjective domains using fitness-based shaping. In: Proceedings of the Genetic and Evolutionary Computation Conference (2010)
36. Schrum, J., Miikkulainen, R.: Evolving multimodal networks for multitask games. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2011), pp. 102–109. IEEE, Seoul (2011)
37. Seno, H.: A density-dependent diffusion model of shoaling of nesting fish. Ecol. Modell. 51, 217–226 (1990)
38. Simpson, G.: The baldwin effect. Evolution 7(2), 110–117 (1953)
39. Stanley, K., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
40. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the NERO video game. IEEE Transactions on Evolutionary Computation 9(6), 653–668 (2005)

41. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the NERO video game. IEEE Transactions on Evolutionary Computation 9(6), 653–668 (2005)
42. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
43. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurrency - Practice and Experience 17(2-4), 323–356 (2005)
44. Tomasello, M.: The cultural origins of human cognition. Harvard Univ. Pr. (1999)
45. Valsalam, V., Miikkulainen, R.: Evolving symmetry for modular system design. IEEE Transactions on Evolutionary Computation 15, 368–386 (2011)
46. Vogt, P., Haasdijk, E.: Modeling social learning of language and skills. Artificial Life 16(4), 289–309 (2010)
47. Waibel, M., Floreano, D., Magnenat, S., Keller, L.: Division of labour and colony efficiency in social insects: effects of interactions between genetic architecture, colony kin structure and rate of perturbations. In: Proceedings of the Royal Society B, vol. 273, pp. 1815–1823 (2006)
48. Watkins, C., Dayan, P.: Q-learning. Machine Learning 8(3), 279–292 (1992)
49. Werbos, P.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10), 1550–1560 (1990)
50. Whiteson, S., Stone, P.: Evolutionary function approximation for reinforcement learning. The Journal of Machine Learning Research 7, 877–917 (2006)
51. Yong, C., Miikkulainen, R.: Coevolution of role-based cooperation in multi-agent systems. IEEE Transactions on Autonomous Mental Development (2010)