Cooperative Coevolution of Multi-Agent Systems

Chern Han Yong and **Risto Miikkulainen**
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
compute,risto@cs.utexas.edu

Technical Report AI01-287

Abstract

In certain tasks such as pursuit and evasion, multiple agents need to coordinate their behavior to achieve a common goal. An interesting question is, how can such behavior best be evolved? When the agents are controlled with neural networks, a powerful method is to coevolve them in separate subpopulations, and test together in the common task. In this paper, such a method, called Multi-Agent ESP (Enforced Subpopulations) is presented, and demonstrated in a prey-capture task. The approach is shown more efficient and robust than evolving a single central controller for all agents. The role of communication in such domains is also studied, and shown to be unnecessary and even detrimental if effective behavior in the task can be expressed as role-based cooperation rather than synchronization.

1 Introduction

In multi-agent problem solving, several agents work together to achieve a common goal. Due to their distributed nature, multi-agent systems can be more efficient, more robust, and more flexible than centralized problem solvers. To be effective, the agents need to interact, and they need to behave cooperatively rather than greedily to accomplish a common objective. The central issue is how such cooperation can be best established. First, should the agents be implemented as a diverse set of autonomous actors, or should they be coordinated by a central controller? Second, if the agents are autonomous, what kind of communication is necessary for them to cooperate effectively in the task?

This paper explores these questions in the context of machine learning, where a team of neural networks is evolved using genetic algorithms to solve a cooperative task. The Enforced Subpopulations method of neuroevolution (ESP [7, 8]), which has proven highly efficient in single-agent reinforcement learning tasks, is first extended to multi-agent evolution. The method is then evaluated in a pursuit-and-evasion task where a team of several predators must cooperate to capture a fast-moving prey.

The main contribution is to show how the different ways of encoding, evolving, and coordinating a team of agents affect performance. We demonstrate two interesting results: first, a central-controller neural-network that is evolved to control the entire team performs substantially worse than a set of autonomous neural networks each evolved cooperatively to control a single predator. This result counteracts the common-sense notion that a centralized controller is useful for a team. Moreover, the agents do not even need to communicate to behave cohesively: it is sufficient that they coevolve to establish compatible roles in the team. In fact, communicating teams (where each team member sees all the other team members) consistently performed worse than non-communicating teams (where they each only see the prey)! These surprising results are due to niching in coevolution, which is especially strong in ESP. Instead of searching the entire space of solutions, coevolution allows identifying a set of simpler subtasks, and optimizing each team member separately and in parallel for one such subtask. In the end, each agent knows what to expect from the other agents, and explicit communication is not necessary.

We will begin with a brief review of related work in cooperative coevolution, multi-agent learning, and the predator-prey domain. The multi-agent Enforced Subpopulations method is then described, followed by its experimental evaluation. A discussion of future prospects of this approach concludes the paper.

2 Background and Related Work

Coevolution in Evolutionary Algorithms refers to maintaining and evolving individuals for different roles in a common task, either in a single population or in multiple populations. In competitive coevolution, these roles are adversarial in that one agent's loss is another one's gain. In cooperative coevolution, however, the agents share the rewards and penalties of successes and failures. The kinds of problems that can best utilize cooperative coevolution are those in which the solution can be naturally modularized into subcomponents that interact or cooperate to solve the problem. Each subcomponent can then be evolved in its own population, and each population contributes its best individual to the solution. For example, Gomez and Miikkulainen [7] developed a method called Enforced Subpopulations (ESP) to evolve populations of neurons to form a neural network. A neuron was selected from each population to form the hidden-layer units of a neural network, which was evaluated on the problem; the fitness was then passed back to the participating neurons. In the multi-agent evolution developed in this paper, we use ESP to evolve each neural network, but also require that the neural networks cooperate. The ESP method and our extension to it are discussed in more detail in Section 4.

Potter and De Jong [21] outlined an architecture and process of cooperative coevolution that is similar to that of ESP, and we also use it partially in our approach. While Potter and De Jong focus on methodological issues such as how to automatically decompose the problem into an appropriate number of subcomponents and roles, our focus is on understanding cooperation itself, including the efficiency of the different models of team control and the role of communication.

In a series of papers, Haynes and Sen explored various ways of encoding, controlling, and evolving predators that behave cooperatively in the predator-prey domain [11, 10, 12]. In the first of these studies [11], Genetic Programming was used to evolve a population of strategies, where each individual was a program that represented the strategies of all predators in the team. The predators were thus said to be homogeneous, since they each shared the same behavioral strategy. In follow-up studies [10, 12], they developed heterogeneous predators: each chromosome in the population was composed of k different programs, each one representing the behavioral strategy of one of the k predators in the team. They reported that the heterogeneous predators were able to perform better than the homogeneous ones. We take a further step in the direction towards increasing heterogeneity by evolving a controller for each predator in separate populations using cooperative coevolution. Also, because ESP has been shown to be powerful in various control tasks [7, 8], we use ESP to evolve neural network controllers, rather than Genetic Programming to evolve program controllers.

The role of communication in cooperative behavior has been studied in several Artificial Life experiments [26, 4]. These studies have shown that communication can be highly beneficial, allowing the communicating individuals to outperform the non-communicating ones. However, most of these studies did not take into account the cost of communication—-such as the energy expenditure in signaling, or the danger of attracting predators, or the complexity of the apparatus required. There also exists other forms of cooperative strategies that do not involve communication. For example, Wagner [24] suggested that even in domains where communication does contribute towards solving a task, communicative traits may still not evolve if they involve a significant cost. Other kinds of cooperative strategies may evolve instead, depending on other factors such as the nature of the task, population density, and availability of resources. This idea is especially relevant to our work, in which agents that do not have communicative abilities still manage to evolve cooperative behavior.

Balch [2] examined the behavioral diversity learned by robot teams using reinforcement learning. He found that when the reinforcement was local, i.e. applied separately to each agent, the agents within the team learned identical behavior; global reinforcement shared by all agents, on the other hand, produced teams with more heterogeneous behavior. This result provides a useful guideline for evolving cooperating agents: rewarding the whole team for good behavior privileges cooperation even when some agents do not contribute as much as others, while rewarding individuals induces more competitive behavior as each individual tries to maximize its own reward at the expense of the good of the entire team. Our work diverges from Balch's in focus and implementation: we study the behavioral diversity and niching of evolved teams

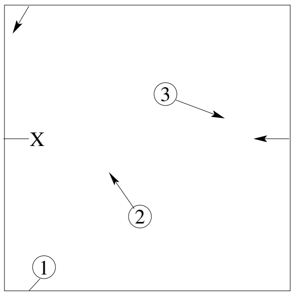

Figure 1: **The Predator-Prey Domain.** The environment is a 100×100 toroidal grid, with one prey (denoted by "X") and three predators (denoted by "1", "2" and "3"). The arrows indicate current direction of movement.

with respect to global vs. local control and communication, and instead of reinforcement learning we use evolutionary learning on neural networks, which tends to give more malleable and efficient performance [8].

In sum, the coevolutionary approach seems to be a good match for multi-agent tasks. The predator-prey domain is a simple but effective domain to test this hypothesis, as will be discussed next.

3 The Predator-Prey Domain

The prey capture task used in this paper is a special case of pursuit-evasion problems [15]. Such tasks consist of an environment with one or more preys and one or more predators. The predators move around the environment trying to catch the prey, and the prey tries to evade the predators. Pursuit-evasion tasks are interesting because they are ubiquitous in the natural world, and offer a clear objective that requires complex coordination with respect to the environment, other agents with the same goal, and adversarial agents [7]. They are therefore challenging for even the best learning systems, allow measuring success accurately, and allow analyzing and visualizing the strategies that evolve.

We use this domain to test various approaches to evolution and control. There is one prey and three predators in the environment (Figure 1). The prey is controlled by a simple algorithm; the predators are controlled by neural networks. The goal is to evolve the neural networks to form an efficient team for catching the prey. The different approaches and techniques are compared based on how long it takes for the team to catch the prey, and what kind of strategies they use.

The environment is a 100×100 square toroid without obstacles or barriers (the 100×100 square is referred to as the "map" below). All agents can move in 4 directions: N, S, E, or W. The prey moves as fast as the predators, and always directly away from the nearest predator. The prey starts at a random location of the map, and the predators all start at the bottom left corner. If the predators have not caught the prey in 150 moves, the trial is terminated and counted as a failure.

Constrained this way, it is impossible to consistently catch the prey without cooperation. First, since the predators always start at the bottom left corner, behaving greedily would mean that they chase the prey as a pack in the same direction. The prey would avoid capture by running in away in the same direction. Because it is as fast as the predators, and the environment is toroidal, they would never catch it (see Figure 10 for an illustration of this scenario). On the other hand, should the predators behave randomly, there is little chance for them to coordinate an approach, let alone maneuver into a capture. The time limit is chosen so that the predators can travel from one corner of the map to the other. This way they have enough time to move to surround the prey, but it is not possible for them to just mill around and bump into the prey eventually.

The prey capture task has been widely used to test multi-agent behavior. For example Benda [5], as well as Haynes and Sen [10, 12], used this domain to assess the performance of different coordination systems. In their variant of the task, the predators were required to surround the prey to catch it. The main difficulty in this task lies in coordinating the predators to occupy the positions of the capture configuration: the prey tends to move either randomly or at a slower speed than the predators, thus allowing the predators to catch up with it easily. In our domain, on the other hand, it is enough for one predator to move onto the prey's position for a successful capture. However, the prey moves as fast as the predators, and always away from the nearest predator; there is thus no way to catch the prey simply by chasing it. The main difficulty that our

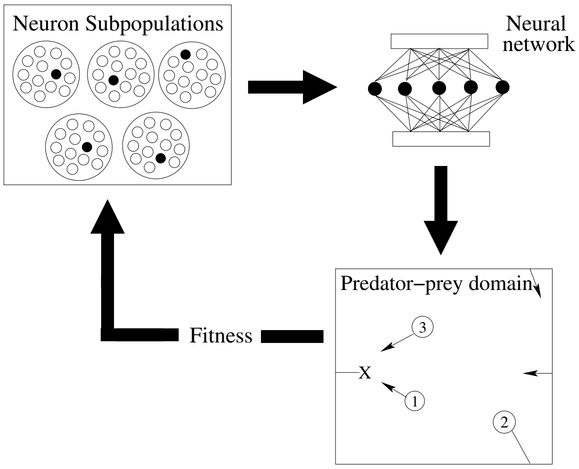

Figure 2: **ESP architecture.** Each population contributes a neuron to form the neural network, which is then evaluated in the domain. The fitness is passed back to the participating neurons. This is also the scheme used to evolve the central-controller neural network that controls all three predators simultaneously. See Figure 4 for a detailed architecture of this neural network.

predators face is coordinating the chase so that in the end the prey has nowhere to go, which requires a more long-term strategy.

4 The Multi-Agent ESP Approach

The Enforced Subpopulations Method (The ESP[1]; [7, 8]) is an extension of Symbiotic, Adaptive Neuro-Evolution (SANE; [17, 18, 16]). SANE is a method of neuro-evolution that evolves a population of neurons instead of complete networks. Neurons are selected from the population to form the hidden layer of a neural network, which is evaluated on the problem. The fitness is then passed back to all the partaking neurons of the network equally. ESP extends on SANE by allocating a separate population for each hidden layer neuron of the network; a number of neuron populations are thus evolved simultaneously (Figure 2). It is thus a cooperative coevolution method of evolving neural networks: each neuron population tends to converge to a role that results in the highest fitness when the neural network is evaluated. This way, ESP decomposes the problem of finding a successful network into several smaller subproblems, resulting in more efficient evolution.

In several robot control benchmark tasks, ESP was compared to other neuro-evolution methods such as SANE, GENITOR [28], and Cellular Encoding [9, 29], as well as to other reinforcement learning methods such as Adaptive Heuristic Critic [3, 1], Q-learning [25, 20], and VAPS [14]. ESP turns out to be consistently the most powerful, solving problems faster, and solving harder problems [8]. It therefore forms a solid foundation for an extension to multi-agent systems evolution.

In this paper, ESP is adapted to allow for the simultaneous evolution of multiple agents. We evaluate two approaches of encoding and controlling agents: the central-controller approach and the autonomous, cooperating controllers approach. Each of these entails a different method of ESP evolution.

In the central-controller approach, all three predators are controlled by a single neural network (Figure 4). Since there is only one network, this system is implemented using the usual ESP method (Figure 2). In the distributed control approach, each predator is controlled by its own network (Figure 5)—there are thus three autonomous networks that need to be evolved simultaneously. During each cycle, each network is formed using the usual ESP method. These three networks are then evaluated together in the domain as a team, and the resulting fitness for the team is distributed among the neurons that constitute the three networks (Figure 3).

[1] ESP neuroevolution software, as well as sofware for SANE, is available at www.cs.utexas.edu/users/nn/pages/software/software.html

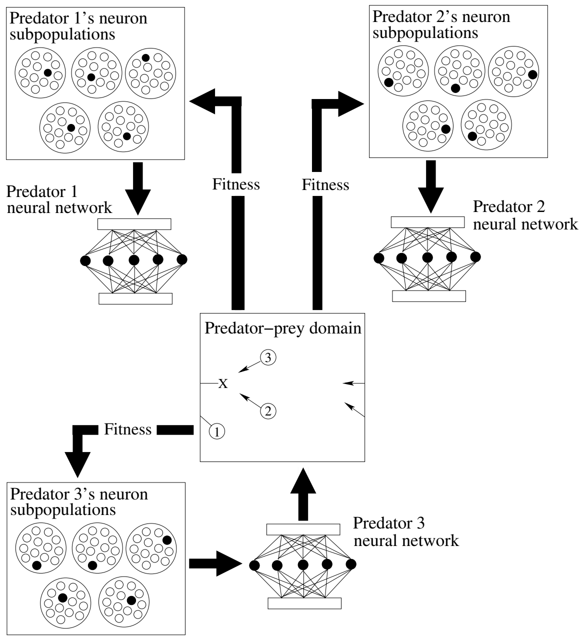

Figure 3: **Multi-agent ESP architecture.** Each predator is controlled by its own neural network, formed from its own subpopulations of neurons. The three neural networks are formed and evaluated in the domain at the same time as a team, and the fitness for the team is passed back to all participating neurons. See Figure 5 for a detailed architecture of the individual networks.

In both the central-controller and the autonomous-controllers approach, the agents were evolved in a series of incrementally more challenging tasks. Such incremental evolution, or shaping, has been found to facilitate learning of complex domains, where direct evolution in the goal task would result in inadequate, mechanical strategies [7, 22, 6]. Evolution proceeds through five stages: in the easiest task the prey is stationary, and in each subsequent task the prey moves at a faster speed, until in the last task it moves as fast as the predators. When a team manages to solve the current task consistently, the next harder task is introduced. In our domain, incremental learning is particulary useful because it gives the predators an opportunity to close in on the stationary or slow-moving prey of the easier tasks and gain experience in catching the prey at close proximity. Placing the predators into the final task right from the start does not give them sufficient exposure to maneuvering close to the prey, as it is difficult to approach the prey to begin with. Incremental learning is therefore used to give evolution more experience in the necessary skills that would otherwise be hard to develop.

The fitness function consists of two components, depending on whether they prey was captured or not:

$$f = \begin{cases} \frac{d_0 - d_e}{10} & \text{if prey not caught} \\ \frac{200 - d_e}{10} & \text{if prey caught} \end{cases}$$

where d_0 is the average initial distance from the prey, d_e is the average final distance from the prey. This fitness function was chosen to satisfy four criteria:

1. If the prey is caught, we should not privilege or deprivilege teams based on the starting scenarios—that is, the initial distance from the prey must not be a factor. However, we should privilege teams if their ending positions are good—that is, if all predators are near the prey.

2. If the prey is not caught, then we should take into account the distance covered by the predators, wherein the initial distance from the prey is a factor.

3. Since a successful strategy has to involve surrounding or sandwiching the prey between two or more predators, at least one predator must travel the long distance of the map and approach the prey from

the furthest direction. Thus the time taken for each capture tends to be about the same, and should not be a factor in the fitness function.

4. The fitness function should have the same form throughout the different stages of incremental evolution, making it simple and convenient to track progress.

The neuron chromosomes are concatenations of the real-valued weights on the input and output connections of the neuron. As is usual in ESP, burst mutation through delta-coding [27] on these weights is used as needed to avoid premature convergence. If progress in evolution stagnates because the populations have converged, the populations are re-initiated according to a Cauchy distribution around the current best solution. Burst mutation typically takes place at task transitions as well as in prolonged evolution in difficult tasks [7, 8]. However, it only happened a couple of times in our simulations, which were not that difficult for ESP to solve.

5 Experiments

In this section, we conduct two experiments to test our main hypotheses: first, that cooperative coevolution of autonomous controllers is more effective than evolving a central controller (Section 5.1), and second, that the agents controlled by autonomous neural networks can learn to cooperate even without any communication between them, and indeed learn more powerful cooperative behavior in a shorter time compared to communicating agents (Section 5.2). In addition to quantitative comparisons of performance, Section 5.3 describes and compares actual example behaviors learned in the three approaches. Finally, in two more experiments, we will test the robustness of the solutions, and will verify that cooperation indeed is necessary for the task.

In each experiment, there are 30 subpopulations of neurons. In the central-controller approach the neural network has 30 hidden-layer neurons, one from each subpopulation; in the distributed approach each predator's neural network has 10 hidden-layer neurons. Each subpopulation consists of 100 neurons. During each evolutionary cycle, 1,000 trials are run wherein the neurons are randomly chosen from their subpopulations to form the neural network(s). In each trial, the team is evaluated six times; the prey starts in a random location each time, while the predators always start in the bottom-left corner. The fitnesses over the six evaluations are averaged, and assigned to all the neurons that constitute the network(s). After the trials, the top 25% of neurons are recombined using 1-point crossover. The offspring replaces the bottom bottom 50% of the neurons, and they are then mutated with a rate of 0.4 on one randomly-chosen weight on each chromosome, by adding a Cauchy-distributed random value to it.

Note that the environment is stochastic only in the prey's starting location, and this is the only factor that determines the course of action taken by the predators. In order to test these strategies comprehensively, we implemented a suite of benchmark problems. The map was divided into nine 33×33 subsquares; in each trial, each team was tested nine times, with the prey starting at the center of each of these squares in turn. Such an arrangement provides a sampling of the different situations, and allows estimating the general effectiveness of each team. A team that manages to catch the prey in seven out of the nine benchmark cases is considered to have learned the task reasonably well; a team that catches the prey in all nine benchmark cases is considered to have completely solved the task, and indeed such a team usually has a 100% success rate in random, general scenarios.

5.1 Standard Evolution of a Central Controller vs. Cooperative Coevolution of Autonomous Controllers

In this section we compare the results of evolving a single neural network that controls the entire team against coevolving three separate neural networks, each controlling a single predator. We shall make the comparison in terms of the number of evolutionary cycles needed for the neural network(s) to learn to solve the task reasonably well; in Section 5.3, the actual the predator behaviors that emerge are described and compared. The network architectures are shown in Figures 4 and 5, and they are evolved according to the scheme outlined in Section 4 above (Figures 2 and 3). Five simulations of each technique were run for 400 evolutionary cycles each.

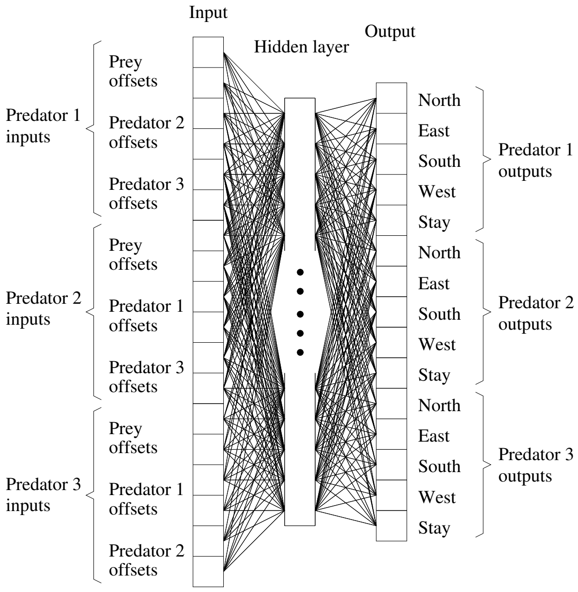

Figure 4: **Central controller network for all three predators..** This neural network receives the relative x and y offsets of the prey and the other predators from the perspective (i.e. location) of all three predators, and outputs the movement decisions for all three predators. This way it acts as the central controller for the whole team. The chromosome size of each hidden layer unit is 33 (18 inputs + 15 outputs).

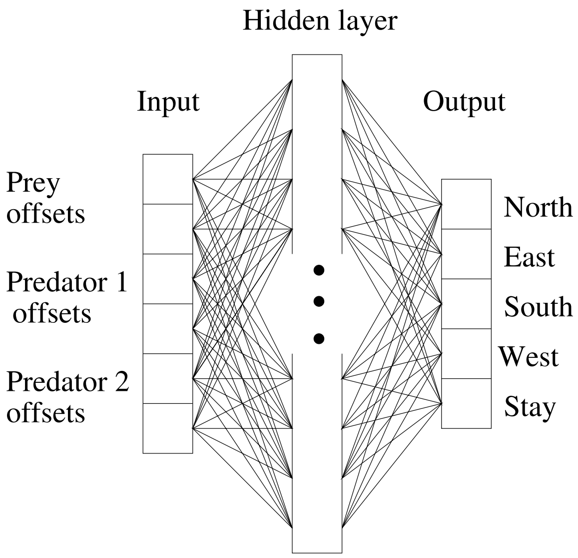

Figure 5: **Controller for a single autonomous cooperating predator.** This neural network autonomously controls one predator; three such networks are simultaneously evolved in the task. The network receives the relative x and y offsets of the prey and the other two predators as its input. The chromosome size of each hidden layer unit is 11 (6 inputs + 5 outputs).

Control	**Generations to Solve 7 Benchmark Cases out of 9**		**Teams that Evolved to Solve 9 Benchmark Cases out of 9**
	Mean	**Standard Deviation**	
Central	231	116	0 out of 5 simulations
Distributed	87	22	5 out of 5 simulations

Table 1: **Learning performance of standard evolution of a central controller versus cooperative coevolution of multiple cooperative controllers.**

Communication	**Generations to Solve 7 Benchmark Cases out of 9**		**Teams that Evolved to Solve 9 Benchmark Cases out of 9**
	Mean	**Standard Deviation**	
With	87	22	5 out of 5 simulations
Without	18	7	5 out of 5 simulations

Table 2: **Learning performance of cooperative coevolution of autonomous controllers with and without communication.**

Table 1 shows the mean and standard deviation of the number of evolutionary generations needed for each approach to learn to solve the task reasonably well—that is, to be able to catch the prey in at least seven benchmark cases out of nine. It also lists the number of teams that learned to solve the task completely—that is, in all nine benchmark cases.

On average, the coevolution of the three neural network controllers was almost three times as fast as the evolution of a centralized controller in finding a reasonable solution (the difference is statistically significant with $p > 0.95$). Furthermore, the single neural network was unable to evolve to the level of expertise required to solve all nine benchmark cases within the 400 evolutionary cycles, whereas the cooperating neural networks were able to do it every time. These results provide convincing evidence that a cooperative coevolution is more powerful than a standard centralized approach in this task.

5.2 Cooperative Coevolution With vs. Without Communication

In the previous section, we saw how separating the control of each agent into disjoint autonomous networks allows for more powerful evolution. Even though the controllers no longer receive direct information about what the other agents see, the domain is still completely represented in the predator's own and the prey's offsets. In this section we reduce the available information by preventing the predators from seeing each other. This way the agents will have to act entirely autonomously, without any direct communication between them. The objective is to find out whether communication is necessary for cooperative behavior to evolve.

The modified network architecture is shown in Figure 6. The predator no longer sees the relative x and y offsets of the other predators, only the offsets of the prey. Such networks were evolved with the same coevolutionary multi-agent ESP method as the communicating networks of Figure 5. Again, five simulations of each system were run for 400 evolutionary cycles.

The learning performance of the communicating and non-communicating controllers is given in Table 2. Somewhat surprisingly, the non-communicating system learned reasonable behavior four times faster on average (the difference is statistically significant with $p > 0.99$). Both systems also learned to solve the task completely without exception. These results show that explicit communication is not necessary for cooperative behavior to emerge in this task; in fact, since it is not necessary, it is more efficient to do away with it entirely. Let us next analyze examples of evolved behaviors to gain insight into why this is the case.

5.3 Analyses and Comparisons of Evolved Behaviors

In this section, we try to characterize the behaviors that emerge from each approach, and to point out their differences. We first describe the emergent behavior of a team of the autonomous cooperative controllers without communication, then do the same for the communicating team, and then compare these two. Finally,

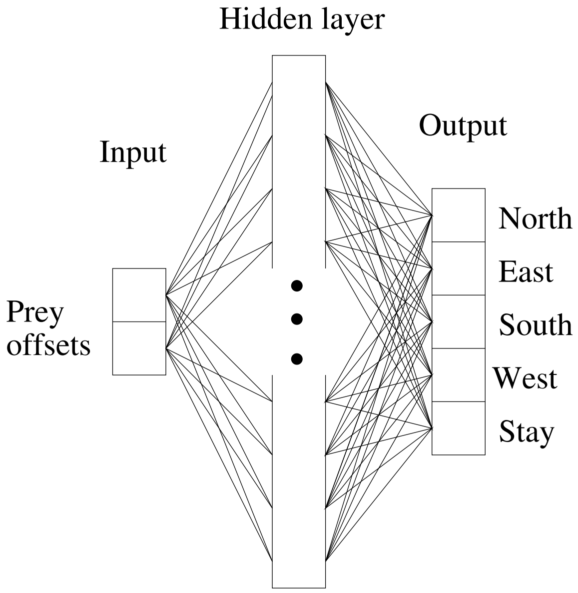

Figure 6: **A non-communicating autonomous controller.** This neural network receives the prey x and y offsets as its inputs. Therefore, it controls a single predator without knowing where the other two predators are (i.e. there is no communication between them). The chromosome size of each hidden layer unit is 7 (2 inputs + 5 outputs).

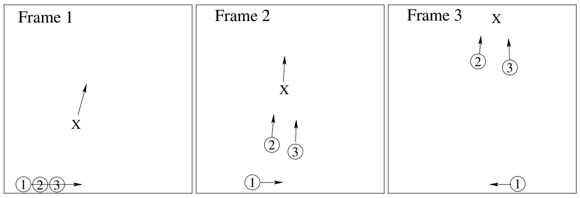

Figure 7: **A sample strategy of a non-communicating team.** The predators 2 and 3 are Chasers, pursuing the prey towards predator 1, which acts as a Blocker.

we will discuss the behavior of the team with a centralized controller compared to the other two.

The main result is that evolution without communication produces teams that evolve specific and rigid roles for each team member, and that utilize a single effective strategy in all cases. On the other hand, evolution with communication tends to produce teams with more flexible (although less effective) agents able to employ two or more different strategies.

Figure 7 illustrates one such successful strategy for a non-communicating team. This strategy involves 2 different roles, which we call the Chaser and the Blocker; predator 1 is a Blocker, while predators 2 and 3 are Chasers. The Blocker only moves in a horizontal direction, moving into and staying on the same vertical axis as the prey; the Chasers pursue the prey vertically, upwards or downwards depending on where the prey is. The first frame in Figure 7 shows the initial positions of the agents and the prey. Predator 1, the Blocker, moves right to get onto the prey's vertical axis; predators 2 and 3 do the same, while the prey flees from them. In frame two, the Chasers are more or less in the same vertical column as the prey, and start chasing it upwards; the Blocker (predator 1) simply keeps on the prey's vertical axis. In frame three, the prey has been trapped between the Blocker and the Chasers, who move in for the capture. Notice that this strategy requires no communication between predators: as long as the Blocker keeps itself on the prey's vertical axis, and the Chasers chase the prey vertically, the prey will always be caught.

Another successful strategy that sometimes emerges involves only Chasers. If the Chasers go after the prey in opposite directions, i.e. one upwards, another downwards, they can sandwich the prey between them and capture it without help from the Blocker. Again, communication is not necessary, only the implicit knowledge that there will be an opposite Chaser in the team. Since the networks are evolved in separate

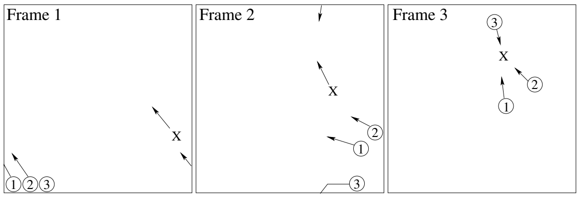

Figure 8: **A strategy of a communicating team.** This strategy shows more versatility, starting with two Chasers and a Blocker, but ending with opposite Chasers.

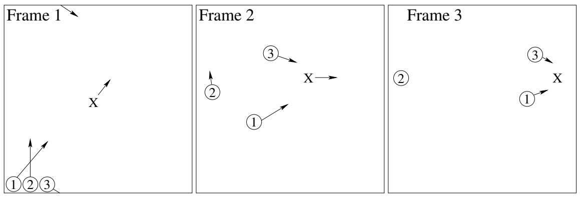

Figure 9: **Another strategy of the same communicating team as in Figure 8.** This time there is a Blocker and two Chasers throughout, but the movement is horizontal.

populations, it is possible to draw successful teams either by combining Chasers with opposite directions, or by combining Chasers and Blockers. Sometimes both kinds of teams can be formed from the same population; they constitute divisions of the high-level problem into useful general-purpose subtasks, which can therefore be found very robustly.

Figures 8 and 9 illustrate the behavior of an evolved communicating team. Two different strategies are shown because this team actually displays both of them, and also their combinations and variations, depending on the prey's starting location and the location of the other predators at each timestep. Figure 8 illustrates behavior similar to the Chaser-Blocker strategy. The first frame is a snapshot of the starting position. Predators 1 and 2 are the Chasers, and they start pursuing the prey upwards. Predator 3 is the Blocker, and it moves left onto the prey's vertical axis. At this point, however, it starts chasing the prey downwards, in Frame 2, until the prey is trapped between all three predators in Frame 3. Already this strategy is more versatile as those of the non-communicating teams, as a combination of Blocking and opposite Chasers.

Figure 9 illustrates another strategy employed by the same team. In the first frame, predators 1 and 3 start moving toward the prey diagonally upwards and downwards, respectively, while predator 2 moves upwards until it is horizontal with the prey. By the second frame, predators 1 and 3 have started chasing the prey horizontally until it is trapped between them and predator 2. This strategy is again similar to the Chaser-Blocker strategy, except this time the prey is chased horizontally instead of vertically, and the chase includes diagonal movement as well.

Although each strategy is similar to those of non-communicating teams, what is significant here is that they are employed by one and the same team. This team can also use combinations of these strategies, depending on the situation, for example by starting with one and finishing with the other. Thus, each predator does not have a specific role it has to perform rigidly, but modifies the strategy depending on the situation. Each predator behaves not only according to the prey's relative location, but also observes the other predators in deciding how to act. This way, their strategy is more versatile, but also less efficient. Whereas the non-communicating teams resemble e.g. players in a well-trained soccer team, where each player knows what to expect from the others in each play, the behavior of the communicating teams is similar to a pickup team where each player has to constantly monitor the others to determine what to do. Such players can perhaps play with many other kinds of players, but not as efficiently.

Of course we have to be somewhat cautious and not attribute undue intelligence and intention to neural networks that simply manage to adapt to each others' behavior; however, the difference in the behavior of the two approaches is striking: the noncommunicating team employs a single, efficient, failproof strategy in which each team member is evolved into a specific and rigid role, while the communicating team adaptively employs variations and combinations of two (or more) strategies.

Communication	Average Number of Benchmark Cases Solved		
	Prey moves 3 steps each time and in random direction 20% of time	**Prey moves 3 steps each time and in random direction 50% of time**	**Prey always moves right only**
With	2.7	2	0
Without	4.3	4.3	1.3

Table 3: **Adaptation of communicating and non-communicating teams to novel prey behavior.** The noncommunicing teams are robust as long as their basic strategy is valid; however, they cannot cope if the prey employs a consistently different strategy. The results were averaged over the three teams of each approach.

In contrast, none of the evolutions of the centrally-controlled teams were able to produce a solution to solve all nine benchmark cases. The cases on which they failed were often characterized by a mechanical strategy: all predators and the prey would make a step in the same direction, and since their relative locations remain unchanged, they would make the same move again and again untill time ran out. However, they behaved similarly to the non-communicating team in that each team would try to utilize a single strategy in all cases, only less cohesively and indeed often failing in the task.

5.4 How robust are the solutions?

Although the non-communicating networks work together like a well-trained soccer team, soccer (like most interesting real-world tasks) is somewhat unpredictable. For example, a player from the other team may intercept a pass, in which case the team members will have to quickly adapt their strategy to cope with the new situation. To determine how the non-communicating team can deal with such unpredictability, we conducted a test in which three teams, evolved until they could solve all 9 benchmark cases, were pitted against a prey that behaved differently from those encountered during evolution. For comparison, we also did the same test for communicating teams. Since the non-communicating teams' predators act according to rigid roles, we expected that they would not be able to adapt as well as the communicating teams' more flexible agents.

The results, summarized in Table 3, are surprising: the non-communicating teams are more robust against unpredictable preys than the communicating ones. Apparently, the first two prey behaviors, which are noisy versions of the original behavior, are still familiar enough so that the rigid roles are effective: the teams still catch the prey about 50% of the time. All the agents have to do is track the occasional erratic movement, otherwise their strategy can remain the same. The communicating teams, however, have a narrower margin of adaptable situations, particularly because their agents tend to switch strategies and roles based on the current state of the map, and thus get easily confused by the unexpected prey actions. In the third case, where the prey always moves right, both teams are unable to adapt. This behavior is consistently novel, and the agents are evolved not to expect it.

In sum, teams that have delegated rigid and specific roles to its members may be more tolerant to noisy or unusual situations, as long as the basic strategy is still valid.

5.5 Is Coevolution Necessary?

Although the performance of cooperative coevolution looks convincing, it does not necessarily mean that coevolution is essential for the task. Perhaps it is possible to evolve good predators individually, and just put them together into the domain? In this section we demonstrate experimentally that such an approach is not sufficient, and the agents indeed must be evolved together to solve the task.

We took a single predator without communication inputs (as shown in Figure 6) and evolved it alone in the prey-capture domain with incremental learning using the standard ESP method as described in Section 4 and Figure 2. The predator was allowed to evolve until it could no longer improve its fitness. This process was repeated twice, each time with a new predator, to produce three independent but capable predators. These three predators were then put into the same environment and evaluated in the prey capture task.

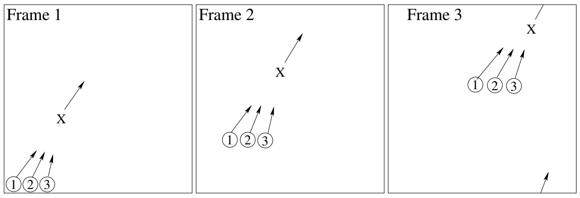

Figure 10: **A strategy of three individually evolved predators placed on the same environment..** The predators chase the prey together in the nearest direction but are unable to catch it

The results clearly support coevolution. When a predator evolves alone, it is never able to catch the prey, since the prey moves at the same speed as the predator. It learns to chase the prey but is never able to reduce the distance between them, and is only capable of preventing the prey from increasing this distance. When the 3 individually evolved predators are put together against the prey, they all chase the prey in the nearest direction, and are unable to catch it at all—the prey keeps running and maintains the distance (Figure 10). In other words, coevolution is necessary in this task to evolve successful cooperative behavior.

6 Discussion

In Section 5.1 we found that evolving a central controller took almost three times as long as coevolving autonomous cooperating controllers up to a reasonable level of performance. Furthermore, the centralized approach was never able to achieve the level of expertise and generality needed to solve all benchmark cases, whereas the cooperative approach did so every time. Cooperative coevolution appears able to decompose the task into simpler roles, thereby making it easier to search for a solution.

Such decomposition is a special case of speciation in evolutionary systems. Speciation has been widely used to maintain diversity in evolution. Using various techniques such as islands and fitness sharing [23, 19, 13], separated populations are encouraged to diverge, allowing more efficient search of the solution space. If these separated populations are further evaluated jointly and rewarded with a global fitness, they tend to converge to heterogenous policies that work well together. This is the driving mechanism behind cooperative coevolution. Thus, the observed cooperation between predators is a necessary consequence of cooperation between populations during evolution.

On the other hand, the central-controller evolution should also be able to generate cooperating agents—after all, the agents are not autonomous, but integral parts of a single decision making system. This system merely has to coordinate its components, much the same way as an infant learns to coordinate his/her legs to walk. Therefore, in cooperative coevolution the agents learn to cooperate indirectly through correlated fitness, whereas in the centralized approach the agents are directly evolved to cooperate. It is thus somewhat surprising that the indirect learning is more efficient. Furthermore, the central controller should perform better because all predators are always completely synchronized and coordinated by a single neural network; there can never be any surprises.

However, such theoretical advantages of the central controller are accompanied by nontrivial costs. The central controller must perform more computations than the three autonomous controllers together. The central controller has 30 neurons with 33 weights each, whereas the disjoint-controllers have 30 neurons with 11 weights each (compare Figures 4 and 5); thus, the centralized evolution has a drastically larger search space to explore. The central controller coordinates the three agents by linking them together with these extra weights, whereas the disjoint controllers coordinate more abstractly through niching: each populations specializes to a useful subtask that can be explored more efficiently, and combinations of subtasks will most often lead to good solutions. Therefore, the practical difficulty for a central network evolution to optimize all three agents at once overwhelms the theoretical advantages of a more coordinated and centralized control.

Section 5.2 revealed the unexpected result that predators who cannot see each other evolve to cooperate and solve the task about four times faster than predators that can. This is surprising because such communication allows each predator to make a decision based on where the other predators are, as well as where the prey is, and should theoretically allow for more complex strategies to evolve. However, as discussed in Section 5.3, we found that the non-communicating team always employed a single strategy where each agent has a rigid and specific role, whereas the communicating team tended to utilize variations and combinations of two or more strategies and the roles are not as well delineated. During evolution, each non-communicating subpopulation converges towards optimizing specific functions such that, as long as each agent performs its role right, the team solves the task successfully, even though the team members are entirely invisible to one another. Evolution without communication thus places strong evolutionary pressure on each predator to perform its assigned role meticulously. The assignment of roles appears to take place through simultaneous adaptive niching: as one agent begins to converge to a particular behavior, the other agents that behave complementarily are rewarded, and themselves begin to niche into such roles; this in turn yields a higher fitness, and all predators begin to converge into cooperative roles.

However, not all multi-agent domains may be as efficiently solved by non-communicating agents. For example in the predator-prey domain where a capture configuration is necessary (as used by Benda [5] and Haynes and Sen [10, 12]) it would be very difficult for the agents to guess the exact locations of the other agents to achieve successful capture. On the other hand, if the agents can let other agents know where they are, they can effectively coordinate their positions. Such a comparison leads to a potentially useful distinction between communication-based cooperative behavior on one hand, and role-based cooperative behavior on the other. In the former, agents cooperate by synchronizing their actions, for example by letting the others know which capture position has already been taken; in the latter, agents cooperate by taking on well-known roles. Our domain is strictly role-based, in that communication is not necessary at all. In such domains, communication is actually a source of noise that diverts teams from the best solution. It is also interesting to note that the capture-configuration domain encourages homogeneously-behaving agents that tend to share the same strategies of discovering unoccupied capture positions and occupying them, whereas our domain encourages heterogeneous-behaving agents to develop and perform specific and different roles. Other domains that involve role-based cooperative behavior may include controlling elevators to most efficiently serve a building, or controlling agents that search the web for information. Such tasks constitute a most interesting direction for future work.

7 Conclusion

The experiments reported in this paper show that evolving several autonomous, cooperating neural networks to control a team of agents is more efficient and robust than evolving a single centralized controller. We proposed an efficient and natural method for such multi-agent cooperative coevolution, called Multi-Agent ESP. Furthermore, a class of problems was identified, called role-based cooperative problems, where communication is not necessary for success, but may actually make evolution less effective. Identifying such problems is still somewhat difficult, although it appears many real world tasks fall in this category. Applying the approach to such tasks, as well as studying ways to deal with novel and changing environments are the main directions of future work in this area.

Acknowledgments

This research was supported in part by the National Science Foundatoin under grant IIS-0083776. We thank Bobby Bryant for insightful suggestions on how to test cooperation and communication, and Tino Gomez for suggestions on adapting ESP to multi-agent systems.

References

[1] Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9:31–37.

[2] Balch, T. (1997). Learning roles: Behavioral diversity in robot teams. In *AAAI Workshop on Multiagent Learning*.

[3] Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846.

[4] Batali, J. (1994). Innate biases and critical periods: Combining evolution and learning in the acquisition of syntax. In Brooks, R. A., and Maes, P., editors, *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, 160–171. Cambridge, MA: MIT Press.

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

[6] Elman, J. L. (1991). Incremental learning, or The importance of starting small. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 443–448. Hillsdale, NJ: Erlbaum.

[7] Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

[8] Gomez, F., and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Denver, CO: Morgan Kaufmann.

[9] Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

[10] Haynes, T., and Sen, S. (1996). Co-adaptation in a team. *International Journal of Computational Intelligence and Organizations*, 1(4).

[11] Haynes, T., and Sen, S. (1996). Evolving behavioral strategies in predators and prey. In Weib, G., and Sen, S., editors, *Adaptation and Learning in Multi-Agent Systems*, Lecture Notes in Computer Science, 113–126. Springer Verlag, Berlin.

[12] Haynes, T., and Sen, S. (1997). Crossover operators for evolving a team. In *Proceedings of Genetic Programming 1997: The Second Annual Conference*, 162–167.

[13] Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

[14] Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999). Learning finite state controllers for partially observable environments. In *Proceedings of the 15th International COnference of Uncertainty in Artificial Intelligence*.

[15] Miller, G., and Cliff, D. (1994). Co-evolution of pursuit and evasion i: Biological and game-theoretic foundations. Technical Report CSRP311, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK.

[16] Moriarty, D. E. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin. Technical Report UT-AI97-257.

[17] Moriarty, D. E., and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

[18] Moriarty, D. E., and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, 5:373–399.

[19] Muhlenbein, H., Schomisch, M., and Born, J. (1991). The parallel genetic algorithm as a function optimizer. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann.

[20] Pendrith, M. (1994). On reinforcement learning of control actions in noisy and non-Markovian domains. Technical Report UNSW-CSE-TR-9410, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia.

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

[22] Savage, T. (1998). Shaping: The link between rats and robots. *Connection Science*, 10:321–340.

[23] Smith, R. E., Forrest, S., and Perelson, A. S. (1992). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149.

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

[25] Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[26] Werner, G. M., and Dyer, M. G. (1991). Evolution of communication in artificial organisms. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Proceedings of the Workshop on Artificial Life (ALIFE '90)*, 659–687. Reading, MA: Addison-Wesley.

[27] Whitley, D., Dominic, S., and Das, R. (1991). Genetic reinforcement learning with multilayer neural networks. In Belew, R. K., and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 562–569. San Francisco, CA: Morgan Kaufmann.

[28] Whitley, D., Dominic, S., Das, R., and Anderson, C. W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284.

[29] Whitley, D., Gruau, F., and Pyeatt, L. (1995). Cellular encoding applied to neurocontrol. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, 460–469. San Francisco, CA: Morgan Kaufmann.