Co-Evolving Team Capture Strategies for Dissimilar Robots

H. Joseph Blumenthal and Gary B. Parker

Computer Science
Connecticut College
New London, Ct 06320 USA
hjblu@conncoll.edu and parker@conncoll.edu

Abstract

Evolving team members to act cohesively is a complex and challenging problem. To allow the greatest range of solutions in team problem solving, heterogeneous agents are desirable. To produce highly specialized agents, team members should be evolved in separate populations. Co-evolution in separate populations requires a system for selecting suitable partners for evaluation at trial time. Selecting too many partners for evaluation drives computation time to unreasonable levels, while selecting too few partners blinds the GA from recognizing highly fit individuals. In previous work, we employed a method based on punctuated anytime learning which periodically tests a number of partner combinations to select a single individual from each population to be used at trail time. We began testing our method in simulation using a two-agent box pushing task. We then expanded our research by simulating a predator-prey scenario in which all the agents had the exact same capabilities. In this paper, we report the expansion of our work by applying this method of team learning to five dissimilar robots.

Introduction

The objective of our work is to show that our system of co-evolution can learn effective capture strategies that capitalize on the strengths of robots with different capabilities. To test the success of our method we choose the predator-prey scenario (Benda, Jagannathan, and Dodhiawalla 1985) because it can represent a simplified version of real world problems to be solved using teams of agents. Such applications include search and rescue missions, scientific data collection, and toxic waste cleanup. Cooperative behavior is an important area of research because robots that cooperate can often achieve much more than the sum of what they could individually. Learning cooperative behavior for robots has been approached in several ways.

Luke and Spector focused their research on testing different implementations of systems for evolving members of a team (Luke and Spector 1996). Specifically, they tested several methods of communication between agents and breeding among members of a single population. As in our current research, Luke and Spector used the predator-prey scenario as the test-bed to compare their various implementations. They used a continuous simulated environment in which four artificially intelligent agents representing "lions" attempt to capture a prey representing a "gazelle". Their method of evolution uses genetic programming and considers each team a single GP individual. Using this method of representation all of the team learning can take place within a single population. This system of learning was found to be highly successful in producing heterogeneous members of a team. Additionally, their implementation is computationally inexpensive due to the evolution within a single population. However, if the controller being evolved for each individual is sufficiently large, the overall size of the chromosome may preclude convergence. In addition, evolving team members in a single population has the potential to limit the level of specialization amongst team members. Evolving the agents in a single chromosome could compromise the GAs ability to recognize suitable team members because a partner's score is overly influenced by the performance of other members of the team to which it is bound within the chromosome. Evolving team members in separate populations is advantageous because the evolutionary power of the GA can focus on producing the most specialized individuals possible.

Potter and De Jong created a system for evolving team members in separate populations. This method called cooperative coevolutionary algorithms (CCAs) tests an individual's fitness by pairing it with a single individual from the other populations (Potter and De Jong 1994). This selected individual used for evaluation is randomly selected for the first generation of training and in subsequent generations the selected individual is the fittest offspring from the previous generation of training compared to others within its own population. Potter, Meeden, and Schultz used this method to co-evolve agents herding sheep into a corral (Potter, Meeden, and Schultz 2001). The herding task was then made more challenging by introducing agents representing wolves that attempted to attack the sheep. Even with this added complexity the method proved a highly successful means of evolving a team.

Wiegand, Liles, and De Jong took an in depth look at the pertinent issues of co-evolution (Wiegand, Liles, and De Jong 2001). They concluded that the most influential factor in co-evolution is the collaborator pool size, the number of combinations of partners tested at trial time. They also note that as the collaborator pool size increases so does the computation required to find a solution to any

given problem. Taking this observation to the extreme, the most accurate method of co-evolution would be to test every possible combination of partners every generation. For a task requiring N partners with I individuals in each population, any round of training would require I^N evaluations; a method too computationally expensive for practical use.

Parker developed Punctuated Anytime Learning (PAL) to allow for an offline learning system to be periodically improved throughout a simulated evolution (Parker 2002). The computer's internal model is updated or the GA's fitness evaluation is adjusted by measuring the robot's actual performance at certain consistently spaced numbers of generations called punctuated generations. These values are used to co-evolve the model parameters of the simulation being used by the GA or they are used to bias all of the fitnesses of the individuals of the GA population.

Parker and Blumenthal adapted the concept of PAL to be applicable to evolving teams (Parker and Blumenthal 2002a). This method employed the periodic nature of PAL to reduce the number of fitness evaluations required to evolve team members in separate populations. Taking into account the observations made by Wiegand, Liles, and De Jong, the method periodically tests every possible partner combination to select a single individual from each population as the best representative of the overall nature of their own population. This selected individual was used as a partner for evaluating the fitness of any individual in the opposing population. Using this method of PAL, the number of fitness evaluations is greatly reduced. This method produces a highly accurate solution, but it is still too computationally intensive to accommodate problems requiring multiple members in a team.

Striving to minimize computation time, additional research showed that it was feasible to select an appropriate representative through a random sampling of the opposing population (Parker and Blumenthal 2002b). In our previous research we tested every possible partner combination, but using the random sampling method we were able to test only a portion of the total number of collaborators to allow for further computational reduction.

While the original PAL co-evolution method was tested using a team of two robots to push a box to a target corner of a simulated area, the implementation of the random sampling method the flexibility to accommodate a greater number of populations. The PAL sampling method was tested using a predator-prey scenario in which four agents attempt to capture a fifth, where all five agents were modeled after the same robot (Blumenthal and Parker 2004). The success of the predators showed that the learning system could accommodate a problem requiring four separate populations.

In this paper, we expand our work by applying this method to dissimilar robots. The actual performance values from five distinct robots were measured and used as agents in the predator-prey scenario. The goal of this research is to show that our system of co-evolution can produce highly specialized behavior by capitalizing on the unique strengths of each team member. Additionally, the problem is more challenging since the robot from which the prey is modeled is more agile and capable than any of the four predators.

PAL With Sampling Populations

Parker developed punctuated anytime learning (PAL) to strengthen offline genetic algorithms. It is a modification of Greffenstette and Ramsey's dynamic anytime learning approach (Greffenstette and Ramsey 1992). Although PAL does not allow the learning system to be updated continuously, it updates every G generations, resulting in a period of accelerated fitness growth. The generations in which the learning system is updated are referred to as "punctuated" generations. When PAL is applied to a single GA, it updates the learning system every G generations by running tests on the actual robot, measures its performance, and uses these results for fitness biasing in the GA or in the co-evolution of model parameters (Parker 2002).

Punctuated anytime learning is a fairly different concept when applied to co-evolving separate populations to form a heterogeneous team. The updated information that the learning system receives in punctuated generations is a more accurate representative of the overall nature of each of the populations. For ease of explanation, assume that the experiment has two populations, population A and population B. In this case, every G generations, all of the individuals in population A are tested against all individuals in population B. The purpose of this process is to find the fittest individual from each population to evolve with the other population in future generations. The chosen representative individual from each population will be referred to as the "alpha individual". The generations in which the computer finds new alphas are called "punctuated" generations. In non-punctuated generations, the alpha individuals selected from the last punctuated generation are paired with perspective team members in the other population for fitness evaluation. This method not only ensures consistency within a generation of training, it also decreases the total number of evaluations required to find an accurate solution.

Although this method was effective, it remains too computationally expensive. In order to further reduce computation time, we tested the possibility of selecting alphas using less than the entire population. Assuming that the experiment has two populations, population A and population B, every G generations, some chosen number of individuals in population A are randomly selected and tested against all individuals in population B. The selected individuals from population A are referred to as the *sample*, and the number of chosen individuals is called the *sampling size*. Our tests (Parker and Blumenthal 2002b) involved using a sample size of 8, which was thought to be a good starting point as it is the square root of our original sample size of 64. We found the sample size of 8 to be both accurate and swift in alpha selection.

Tests were done on the sample sizes of 1, 2, 4, 8, 16, 32, and 64 to determine their relative strength (Parker and Blumenthal 2003). In order to ensure proper testing of the characteristics of the different sample sizes, we staggered the punctuated generations such that the sample 1 performed alpha selection every generation, sample 2 selected new alphas every other generation, up to the sample 64 which selects the new alphas every 64^{th} generation. This staggering was essential to provide each sample size with equal numbers of alpha evaluations. We tested our sample sizes using a box-pushing problem in which two robots are charged with the task of moving a box from the center of a square space to a target corner. The maximum fitness of these trials was 15,625. The performance of the fittest combination of partners for each of the seven different sampling rates were recorded through 10240 alpha evaluations. Fitnessses were recorded at 0, 64, 128, 256, 512, 1024, 2048, 5120, and 10240 alpha evaluations. The tests showed that all seven sample sizes reached reasonably accurate solutions and the sample 64 ultimately reached the highest average fitness of any sample size by the 10240^{th} alpha evaluation. The lower sample sizes showed relatively superior performance in the earlier generations of training, but inferior performance in later generations. The opposite was true for the higher sampling sizes where they were inferior to lower sizes in the earlier generations, but better in the later generations of training.

Although the lower sample sizes did not produce the best end result, they were better in the initial stages of learning. This is rather intuitive because by the sixty-fourth generation, the sample one has evolved with sixty-four different set of alpha individuals while the sample sixty-four has evolved with only one set of alphas. Samples 4 and 8 had accelerated growth past the sample sizes of 1 and 2, because these sample sizes lack the ability to represent the true nature of a population with so few comparisons for alpha selection. Therefore, we select a sample 8 as a good candidate for consistent growth throughout the entire period of training.

To express mathematically the computational reduction achieved by this union of methods, we let G represent the number of generations between alpha selections, I represent the number of individuals in a single population, and N represent the number of populations. The method in which we compared all individuals in a population against all others in the opposing population required I^N evaluations each generation. When the alpha selection is done only at punctuated generations the number of evaluations is reduced by that factor of G. Further reductions are necessary to accommodate the co-evolution of more than two populations. In order to further reduce computations, sampling is used. Using the previous parameters and adding the term S representing the sampling size, any given evolution requires an average of only $N * (I * S^{N-1})/G$ evaluations per generation.

Simulation of Robot Performance

There are two different kinds of robots modeled in the simulation, the ServoBot and the Hex II (produced by Lynxmotion, Inc.) All five agents were modeled after existing hexapod robots in the lab, each with distinct capabilities. The four predators were abstracted from the ServoBots while the prey was abstracted from the Hex II robot. The ServoBot is an inexpensive hexapod robot made of masonite, a hard pressed wood. The Hex II is a hexapod robot which has a body constructed of yellow plastic. Although the Hex II is lacking in some durability, a characteristic that makes us prefer our own robots, the Hex II is the fastest and most capable of any of the five robots modeled. All five robots modeled were hexapod robots each with twelve hobby servos to provide locomotion. Figure 1 shows a picture of a ServoBot and a Hex II robot.

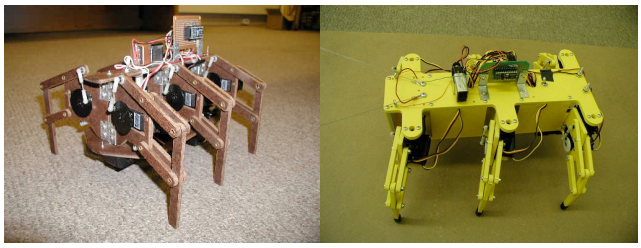

Figure 1: The robot shown on left is the ServoBot and the robot shown on the right is the Hex II robot.

The movements of the servos are coordinated by a central controller, a BASIC Stamp II capable of individually addressing each of the twelve servo actuators (two on each leg) in parallel to produce and sustain forward motion. The BASIC Stamp II is capable of storing a sequence of activations (pulses) to be sent to the servo every 25 msec. These activations, if sequenced and repeated correctly, produce a *gait cycle*, which is defined as the timed and coordinated motion of the legs of a robot such that the legs return to the positions from which they began the motion. Each activation represents the simultaneous movement of the twelve servos. The sequence which controls the movement of the twelve servos is represented in the controller as a twelve-bit number. Each bit represents a single servo with a 0 or a 1. For the horizontal servos, a 1 indicates full back and a 0 indicates full forward. Similarly, for servos oriented in a vertical capacity, a 1 corresponds to full lift and a 0 corresponds to full down. Therefore, each pair of bits can represent the motion of one leg, each bit controlling one servo, corresponding to one of the two degrees of freedom. The pairs of bits are ordered to their represented leg as 0 to 5 with legs 0,2,4 being on the right from front to back and 1,3,5 being on the left from front to back. Figure 2 shows a twelve-bit activation and the corresponding movement of the six legs.

With this method of representation, a cyclic genetic algorithm (CGA), which is discussed in a later section, can be used to evolve an optimal gait cycle for a specific ServoBot. The gait cycle used in our simulation was a

tripod gait, in which three legs provide thrust while three legs are repositioning to provide thrust on the next set of activations.

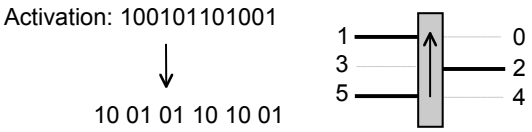

Figure 2: Legs 1,2,5 are moving back while staying down and legs 0,3,4 are moving forward while they lift.

By varying the number of pulses to each side of the robot, we were able to generate degrees of turns varying from sweeping to sharp. These were generated for our robots by decreasing the total number of pulses sent to the thrust producing legs of one side of the robot. If legs 1,3,5 were given 30 pulses during thrust, but legs 0,2,4 were only given 10 pulses during thrust, the result would be a right turn due to the reduced thrust generated by the left legs (0,2,4) throughout the duration of the gait cycle. The effects of each of 15 left and right turns, plus no turn, were measured as they were performed by the ServoBot being tested. These turns are unique to the particular robot. For example, the recorded "no turn" drifted to one side on one robot and to the other side on another robot. These variances are due to minor differences in the physical construction of each robot. The 31 performance values (measured in centimeters moved and degrees turned) were recorded and stored in a table for each robot.

Cyclic Genetic Algorithms

A type of evolutionary algorithm called a cyclic genetic algorithm was used to evolve all four predators coordination strategy (Parker and Rawlins 1996). A CGA is similar to a regular GA, but it differs in that the CGA chromosome can represent a cycle of tasks. These tasks can be anything from a single action to a sub-cycle of tasks. Using this method of representation, it is possible to break up a chromosome into multiple genes with each gene acting as a cycle. Each gene or sub-cycle contains two parts, one part representing an action or set of actions, and the second part representing the number of times that action is to be repeated. The genes can be arranged into a repeated sequence and a chromosome can be arranged with single or multiple cycles. In the case of multiple cycles, it is possible to switch from one to the other at any point.

Individuals were selected stochastically for breeding based on their fitness score and standard operators were used for the CGAs. The CGA was perfectly fit for evolving our agents because it is designed for learning cyclic behavior such as encircling the prey to keep it from escaping the area. Our CGA chromosome has two parts, the first is their motion before they first see the prey (stalking or searching behavior) and the second defines their motions after they first see the prey, which coordinates the pursuit and capture with the other agents.

The CGA chromosome had two cycles containing nine genes each. Every gene contained two 5-bit numbers, one representing a gait cycle with 31 possible turns or a 0, which indicated that it was to stand still, and the other representing the repetitions of that gait cycle. The scheme representation of the chromosome is shown in Figure 3.

$(((T_1\ R_1)\ (T_2\ R_2)\ \ldots\ (T_9\ R_9))$ **$((T_1\ R_1)\ (T_2\ R_2)\ \ldots\ (T_9\ R_9)))$**

Figure 3: Scheme representation of the CGA chromosome where T is a specific turn and R is the number of repetitions of that turn. The genes which appear in bold represent the second cycle.

The Predator-Prey Scenario

The chosen task to prove the effectiveness of our method is to have five hexapod robots participate in the pursuit domain of the predator-prey scenario. Four of the robots in the simulation are the predators and the fifth is the prey. The predators' goal, starting from randomly assigned positions, is to chase and capture the prey, which starts in the center of the colony space (simulated area measuring 500x500 units). The predators must capture the prey before it escapes by reaching the edge of the colony space or all the prey has taken 150 steps.

The prey attempts to evade the predators using the nearest predator first algorithm (NPF), which simply moves the prey in the opposite direction of the nearest predator. If the prey does not see any predators, three out of four times a random number from 0 to 31 is generated to determine its next gait cycle and the other one fourth of the time the prey remains stationary to simulate a feeding behavior. This means that the prey never follows the same path twice because of the random gait generation. To have a successful capture, a predator must move within 12 units of the prey, a distance equivalent to approximately twice the size of the agents themselves.

The fitness score of a team of predators that fail to capture the prey is the number of steps taken in the round. Because it is possible for the prey to escape the environment, the predators are rewarded for keeping the prey within the simulation for as long as possible. In the event of a capture, the fitness of a team is the number of steps in the round plus a bonus for the capture, which is derived from the distance of the capturing predator to the prey. Equation 1 shows how the score is calculated for a successful capture. With an equal number of rows and columns in the simulation space this number is represented by *NUM-COL-ROW*, the maximum distance that can be between a predator and a prey for it to be considered a capture is *MAX-CAPT-DIST*, and the distance of the capturing predator to the prey is *capt-dist*. If a member of the team of predators captures the prey at the maximum allowable capture distance, the team is given a score of *NUM-COL-ROW* because the denominator turns to 1. However, if the distance of the capture is less than the

maximum, the fraction (*capt-dist* / *MAX-CAPT-DIST*) becomes increasingly smaller forcing the score to elevate drastically. Equation 1 shows the fitness function as implemented in the program. We decided to focus our fitness function on the distance of the capture because in order to achieve a consistently small capture distance a team is forced to immobilize the prey.

$$\frac{NUM\_COL\_ROW}{\sqrt{\frac{capt\_dist}{MAX\_CAPT\_DIST}}} \quad (1)$$

At the beginning of any GA run, the prey is placed in the center of the simulated area at the point (250,250) with a random heading from 0 to 360 degrees. The four predators are assigned random starting headings and coordinates except that no predator can start within sight of the prey. These assigned random starting positions are held throughout every generation of training. The simulation is a non-bounded-box environment, meaning that any of the predators or the prey can step out of the simulation at anytime. In the event that a predator steps out of bounds, it is automatically removed from the round. The simulation ends if the prey moves out of bounds, is captured, or the prey has taken 150 steps.

All of the tests done in simulation use agents with different capabilities that model existing robots in the lab. All five agents in the simulation were represented as circles and could see for 125 units in any direction. Both the predators and prey move simultaneously. In previous work, the predators and the prey had the same capabilities. In this scenario, the prey is far more capable than any of the four predators. Where the prey can move a possible forward distance of 16.5 cm the fastest predator can move only 13.3 cm. The prey can also out maneuver the predators. The prey has the advantage of being able to turn more sharply than the predators. The most capable of the predators is predator A. This predator can move forward 13.3 cm. While predator A can move forward the greatest distance in a single gait cycle, predator B is more maneuverable. Predator B can move almost 27 degrees with a full left turn and 24 degrees with a full right turn. Predator B can turn approximately 2 degrees more in either direction than any other agent. The predator C, is the least capable of any agent in the simulation. It can move .1 cm further straight than predator B, but can turn left 3 degrees less than any other. Predator D has fairly average capabilities all around, it can move forward 12 cm, turn left 20 degrees, and right 17 degrees.

Results

To test our method we ran 10 different tests for 20480 generations each. These scores were calculated by averaging 100 evaluations of the team of alpha individuals. We chose to record average scores for each team of alphas because the prey's motion in any given evaluation is random. These fitnesses were averaged and recorded after 0, 64, 128, 256, 512, 1024, 2048, 5120, 10240, and 20480 generation of training. We chose to use a sample 8 with punctuated generations every 64th generation. The sample 8 provides a candidate for consistent growth throughout the entire evolution. A graph of the fitnesses of each of the 10 runs is shown in Figure 4. In all ten test runs, the four predators were able to cooperate to surround the prey. The lowest fitness achieved by any team of alphas by generation 20480 was 150. This means that on average the team of predators was able to successfully stop the prey from escaping the area. It is easy to see that the fitness of one test far outperforms all the other runs of the GA. The best run of the GA achieved a fitness score of almost 800. This corresponds to capturing the prey 70% of rounds tested. We attribute this to the fact that the random placing generated for the predators started them in a highly advantageous formation.

One way to interpret the fitness values in this scenario is to consider that a team with a score of 500, the dimensions of the board or NUM-COL-ROW, is averaging a capture of the prey at the maximum capture distance every round. This goal is very difficult for the predators to attain because the prey is much more capable than any of the predators. Taking a look at Figure 5, it is easier to see the fitness growth in the earlier generations of training. The runs of the GA show a nice fitness curve with the exception of tests 4 and 6. However, from looking at the table, you can see that their fitness rebounds in the next recorded generation. Although these tests have a temporary lapse in fitness the average line is still a smooth increasing curve. Table 1 shows the fitnesses of all ten runs of the GA. As can be seen the average score of the four alphas at generation 20480 is 342 which is a high level of cooperation considering the disadvantages given to the predators in the simulation.

Another important way to interpret the results is to look at the capture strategies themselves. For our tests of the punctuated anytime learning method to be considered a success the learning system must capitalize on the individual strengths of each of the four distinct predators. Figure 6 shows a capture from one of the 10 test runs of the GA. The four predators are marked A, B, C, and D. All five agents starting positions are marked with a square around them. When the predators are within their range of sight of the prey, they appear with a white dot in the middle. A black line is drawn where they first spot the prey; the location where they switched from the first cycle of the CGA to the second. The prey is originally colored black, but when it is evading a predator it turns white. There is one area where predator A seems to cut through the prey, but this is the trail of where the prey started in the first steps of the round.

All four predators in Figure 6 start by finding a position to surround or immobilize the prey. Predator C holds the most defensive position, probably due to its starting position and the fact that it is the least capable of all four

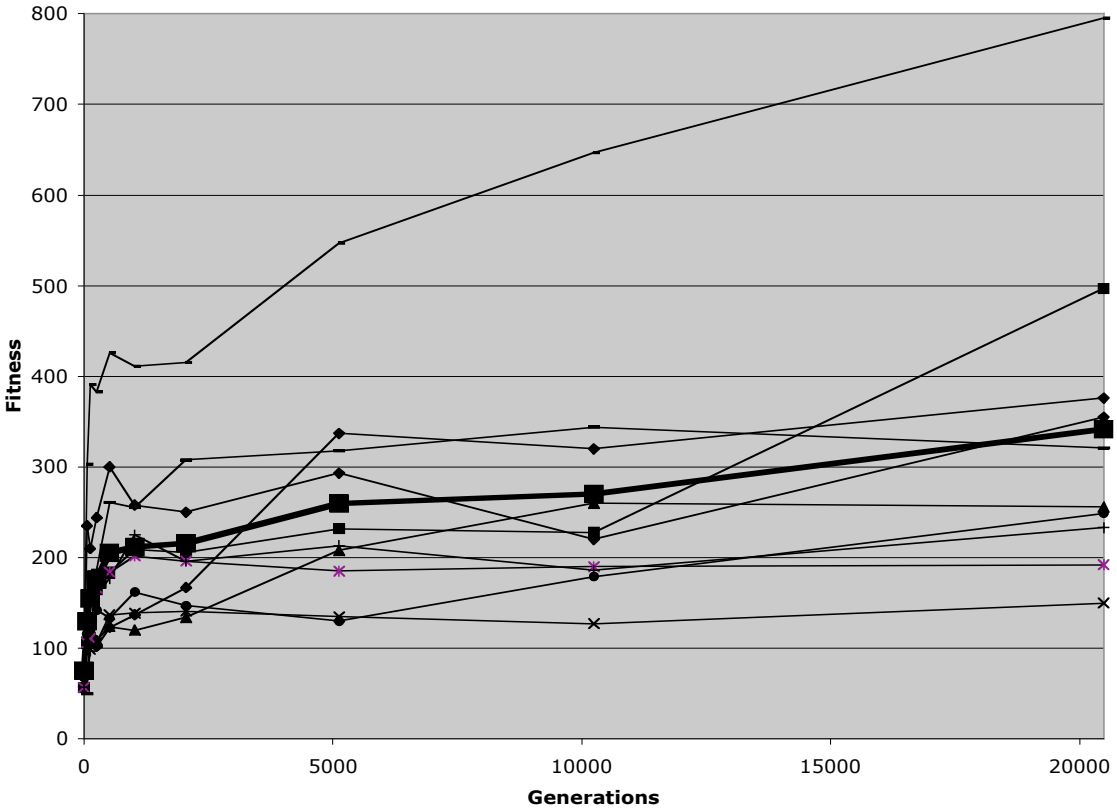

Figure 4: Plotted above are the 10 independent runs of the GA, with the average shown in bold.

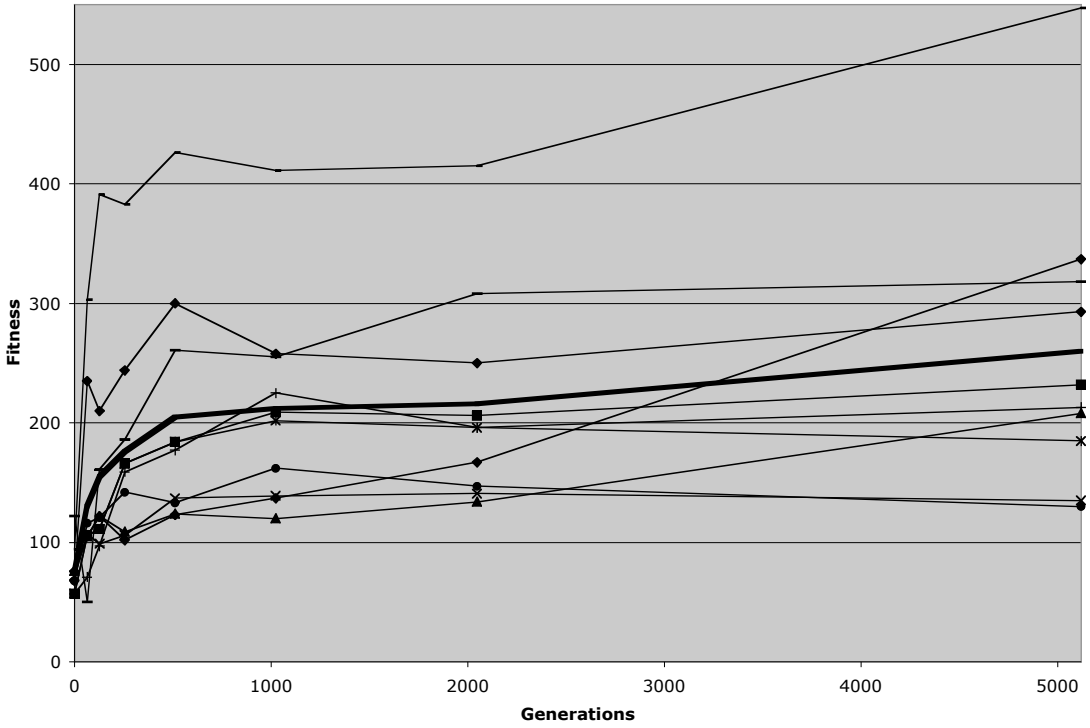

Figure 5: This graph shows the first 5120 generations of the ten runs of the GA. The average is shown in bold.

Generations	0	64	128	256	512	1024	2048	5120	10240	20480
test1	68	235	210	244	300	258	250	293	220	355
test2	57	106	111	166	184	209	206	232	228	497
test3	76	105	122	109	124	120	134	208	260	256
test4	72	107	99	106	137	139	141	135	127	150
test5	57	106	111	166	184	202	196	185	190	192
test6	68	116	121	142	133	162	147	130	179	249
test7	57	71	97	159	177	225	196	213	186	233
test8	94	303	391	383	426	411	415	547	647	795
test9	122	50	161	186	261	255	308	318	344	321
test10	76	105	122	102	123	137	167	337	320	376
Average	75	130	155	176	205	212	216	260	270	342

Table 1: Shown are the 10 averaged and recorded fitness values that comprise the graphs Figure 4 and 5.

robots. Predator D evolved (0 9) as the first gene of the second cycle, meaning that it pauses for nine steps when it sees the prey. This behavior shows specialization because it is working with Predator C to isolate the prey from the entire right side of the screen. Not only is the prey in position for predators A and B, but it eliminates the chance that the prey can escape from the right side of the screen avoiding the capture. Predator C evolved (24 29) as the first gene in the second cycle. In effect, predators C and D are achieving the same end, except that predator C holds position with a tight turn.

All four predators in Figure 6 start by finding a position to surround or immobilize the prey. Predator C holds the most defensive position, probably due to its starting position and the fact that it is the least capable of all four robots. Predator D evolved (0 9) as the first gene of the second cycle, meaning that it pauses for nine steps when it sees the prey. This behavior shows specialization because it is working with Predator C to isolate the prey from the entire right side of the screen. Not only is the prey in position for predators A and B, but it eliminates the chance that the prey can escape from the right side of the screen avoiding the capture. Predator C evolved (24 29) as the first gene in the second cycle. In effect, predators C and D are achieving the same end, except that predator C holds position with a tight turn.

The most interesting facet of the capture strategy is how predators A and B coordinate their final attack on the prey. As previously stated predator B is the most maneuverable (sharpest turning radius) of the predators and the predator A is fastest moving straight forward. In the strategy you can tell that predator B is making a sharp turn to get the prey is proper position while predator A makes a more direct path ultimately capturing the prey. This team of four predators was able to capture the prey approximately 68% percent of all rounds tested. The punctuated anytime learning method was able to successfully use the unique strengths of the predators to form a cohesive capture strategy.

Conclusions

The intent of our research was to show that the punctuated anytime learning method can evolve solutions for robots with different capabilities. Additionally, we wanted to show that the predators can still be successful if given a relative disadvantage in respect to the prey. The sampling method was able to achieve this end while reducing the computation time required by a factor of 128 compared to our original method. Our system of punctuated anytime learning with sampling of populations evolved accurate and computationally inexpensive solutions to the given predator-prey scenario. In the future we hope to expand upon this idea of different capabilities by attaching different sensors to each of the four robots in addition to their pre-existing unique locomotion capabilities.

References

Benda, M., Jagannathan, V., and Dodhiawalla R. 1985: On optimal cooperation of knowledge sources, Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA.

Blumenthal, H. J. and Parker, G. B. 2004. Punctuated Anytime Learning for Evolving Multi-Agent Capture Strategies. In *Proceedings of the Congress on Evolutionary Computation*, 1820-1827. (CEC 2004)

Grefenstette, J. J. and Ramsey, C. L. 1992. An Approach to Anytime Learning. In. *Proceeding of the Ninth International Conference on Machine Learning,* 189-195.

Luke, S. and Spector, L. 1996. Evolving Teamwork and Coordination with Genetic Programming. In *Proceedings of First Genetic Programming Conference*, 150-156.

Parker, G. B. 2002. Punctuated Anytime Learning for Hexapod Gait Generation. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2664-2671. (IROS 2002).

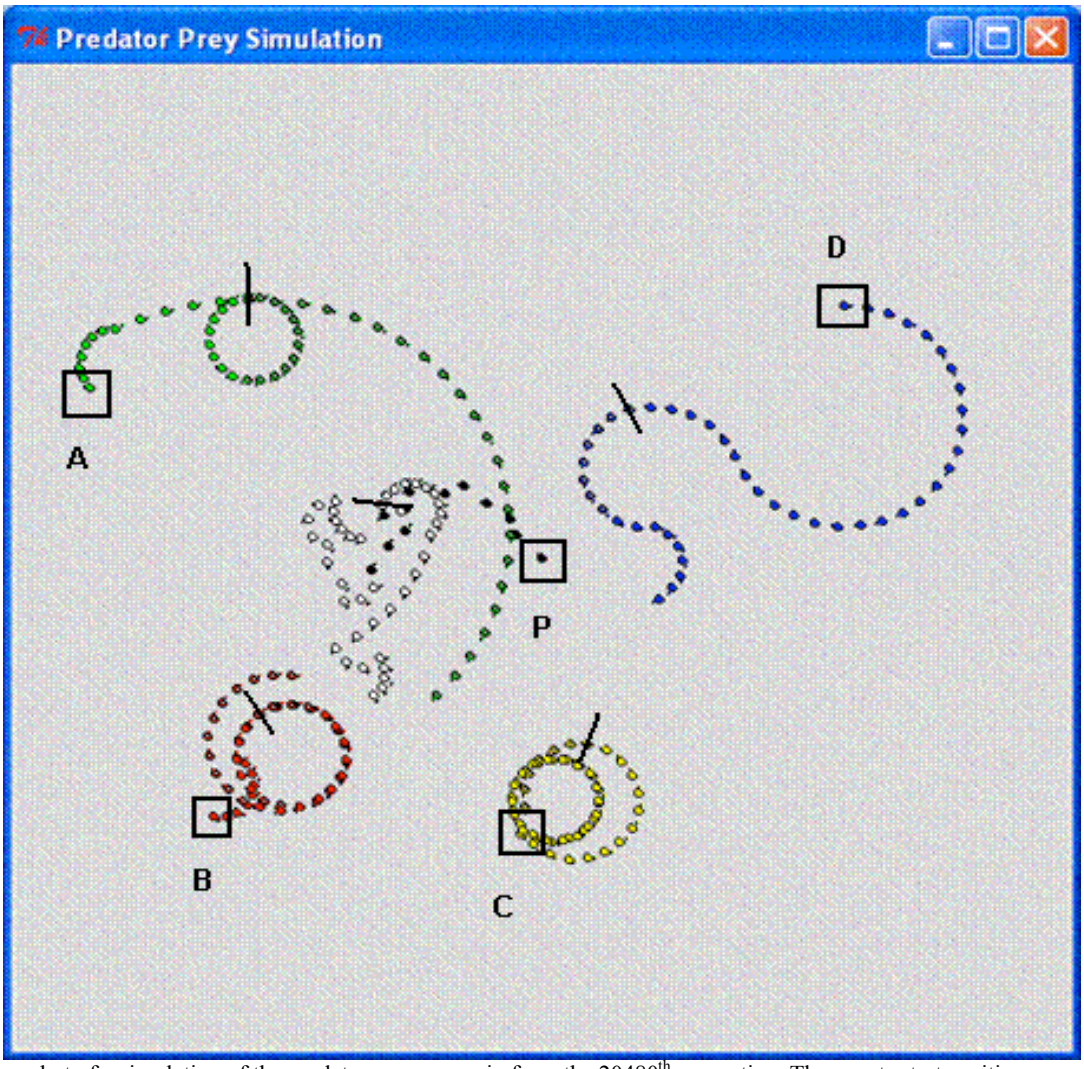

Figure 6: A snapshot of a simulation of the predator-prey scenario from the 20480th generation. The agents start positions are marked with a square and a line is drawn through the predators when they first see the prey. The prey turns from black to white when it is being chased and is marked with a black line the first time it spots a predator.

Parker, G. B. and Blumenthal H. J. 2003. Comparison of Sampling Sizes for the Co-Evolution of Cooperative Agents. In *Proceedings of the 2003 Congress on Evolutionary Computation*, 536-543. (CEC 2003).

Parker, G. B. and Blumenthal, H. J. 2002. Sampling the Nature of A Population: Punctuated Anytime Learning For Co-Evolving A Team. In *Intelligent Engineering Systems Through Artificial Neural Networks,* Vol. 12 207-212. (ANNIE 2002).

Parker, G. B. and Blumenthal, H. J. 2002. Punctuated Anytime Learning for Evolving a Team. In *Proceedings of the World Automation Congress,* Vol. 14*, Robotics, Manufacturing, Automation and Control*, 559-566. (WAC2002).

Parker, G. B. and Rawlins, G. J.E., 1996. Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots. In *Proceedings of the World Automation Congress* Volume 3, Robotic and Manufacturing Systems, 617-622. (WAC '96).

Potter M. A. and De Jong K. A. 1994. A Cooperative Coevolutionary Approach to Function Optimization. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature,* 249-257.

Potter, M. A., Meeden L. A., and Schultz A. C. 2001. Heterogeneity in the Coevolved Behaviors of Mobile Robots: The Emergence of Specialists. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence*. (2001).

Wiegand R. P., Liles W. C., and De Jong K. A. 2001. An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference,* 1235-1245. (GECCO 2001).