Punctuated Anytime Learning for Evolving Multi-Agent Capture Strategies

H. Joseph Blumenthal and Gary B. Parker
Computer Science
Connecticut College
New London, CT 06320 USA
hjblu@conncoll.edu and parker@conncoll.edu

***Abstract*- The evolution of a team of heterogeneous agents is challenging. To allow the greatest level of specialization team members must be evolved in separate populations, but finding acceptable partners for evaluation at trial time is difficult. Testing too few partners blinds the GA from recognizing fit solutions while testing too many partners makes the computation time unmanageable. We developed a system based on punctuated anytime learning that periodically tests a number of partner combinations to select a single individual from each population to be used at trial time. We previously tested our method with a two agent box-pushing task. In this work, we show the efficiency of our method by applying it to the predator-prey scenario.**

I. Introduction

The objective of this work is to extend the range of our method of co-evolution by testing it in a problem involving four separate populations. The success of our method rests upon the GA's ability to produce a highly accurate solution while minimizing computation. We choose the predator-prey scenario [1] because it can easily accommodate teams of four agents, and it represents a simplified problem with respect to its possible applications. Additionally, by choosing a new task we expand the problems solved by our method. Cooperative behavior is an important field in robotics because robots that cooperate can often achieve much more than the sum of what they could do individually. Learning cooperative behavior for robots has been approached in several ways.

Luke and Spector developed a method for evolving a team of agents that act cooperatively [2]. As in our current research, Luke and Spector used the predator-prey scenario as the test-bed for their research. In their simulation four artificially intelligent agents representing "lions" attempt to trap the prey representing a "gazelle". Their method used genetic programming and represented the entire team of four lions as a single GP individual. Even though this method showed to be a good method for evolving heterogeneous behavior, this approach has the potential to limit specialization in the evolution. Evolving team members in a single chromosome compromises the ability of the GA to recognize suitable team members because a partner's score is overly influenced by the performance of other members of the team. Evolving team members in separate populations will tend toward specialization, so the evolutionary power of the GA can be concentrated on producing the most specialized individual possible.

Potter and De Jong created a method for evolving team members in separate populations with cooperative coevolutionary algorithms (CCAs) [3]. This method tests an individual's fitness by pairing it with a single individual from the other populations. This single individual used at trial time is the fittest individual from the last generation of training. Potter, Meeden, and Shultz employed this method to co-evolve agents to heard sheep into a corral [4]. The agents were then given the added challenge by introducing agents representing wolves which tried to attack the sheep. This method proved highly effective and the evolved agents moved the sheep into the corral while providing protection from the wolves.

While this method proved to be a successful means to evolve heterogeneous behavior, the CCA method still limits each individual's fitness calculation to being computed with only a single partner. In later work, Wiegand, Liles, and De Jong took an in depth look at the pertinent issues of co-evolution [5]. They concluded that the most influential factor in co-evolution is the collaborator pool size, or 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 approach 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 a learning system to be periodically updated throughout a simulated evolution [6]. The computer's internal model in simulation is updated or the GA's fitness evaluation is altered by measuring the robot's actual performance at certain consistently spaced numbers of generations called punctuated generations and entering those values into the GA.

Parker and Blumenthal adapted the concept of PAL to be applicable to evolving cooperative teams [7]. This method used the periodic nature of PAL to minimize the number of fitness evaluations required to evolve team members in separate populations. As previously stated, the most accurate method of fitness evaluation would be

to get an individual's fitness by pairing it with all possible partners at trial time. In order to reduce the number of fitness evaluations, at certain punctuated generations this method selected a single individual from each population as the best representative of the overall nature of their own population. This selected individual, referred to as an *alpha individual*, was used as a partner at trial times for evaluating the fitness of any individual in the opposing population. Employing this method of PAL, with *G* generations between each round of alpha selection, the number of fitness evaluations is reduced by a factor of *G*. Although these results showed that this method produces a highly accurate solution, it is still too computationally intensive to accommodate more than two populations.

Additional research showed that it is possible to further reduce computations during alpha selection by testing each individual's fitness with less than the entirety of the opposing population [8]. The chosen group used for alpha selection is referred to as the *sample* and the number of individuals in that sample is called the *sample size*. Upon realizing the success of this method, Parker and Blumenthal focused research on testing a number of differing sample sizes while ensuring that each perform an equivalent number of evaluations by changing the spacing between alpha selections during an evolution. It was found that the sample sizes of 4, 8, and 16 represent good candidates for consistent growth throughout an evolution. The previous research was tested by evolving a team of two robots to push a box to a target corner of the simulated area. The success of the PAL sampling method in solving this new problem expands the use of our method indicating its potential for general applicability. In this paper, we show the results of testing our sampling method using the predator-prey scenario with a team of four agents. The success of the PAL sampling method in solving this new problem expands the use of our method indicating its potential for general applicability, with an increase in the number of team members.

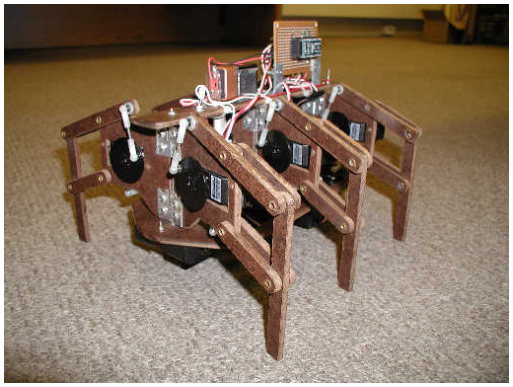

Figure 1. The ServoBot

II. Robot Simulation

All five robots simulated are modeled after ServoBots, which are inexpensive hexapod robots made of pressed wood with twelve hobby servos (Figure 1). 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) to produce and sustain forward motion. The BASIC Stamp II is capable of storing a sequence of timed activations to be repeated. These timed activations if sequenced correctly produce a *gait cycle* 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 list of controls for 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. Likewise, for the vertical servos a 0 corresponds to full down and a 1 corresponds to full lift. 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). Figure 2 also shows the corresponding twelve-bit activation. By this example, the number 001000000000 would lift the front left leg up, and 000001000000 would pull the second right leg backward.

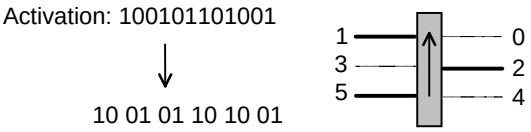

Figure 2: Diagram numbering the legs of the ServoBot and a sample twelve-bit activation.

Each activation is held by the controller for one pulse (approximately 25 msec). 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 [9]. 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. This near optimal gait cycle, which requires a sequence of 29 pulses, was learned by the CGA four our specific ServoBot.

Different degrees of turns varying from sweeping to sharp were then generated for our ServoBot by decreasing the total number of pulses sent to one side of the robot. If legs 1,3,5 were given all 29 pulses but legs 0,2,4 were

only given 14 pulses 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 the 14 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 ServoBot, for example the recorded "no turn" actually drifted left due to minor differences in the physical construction. These 31 performance values (measured in centimeters moved and degrees turned) were recorded and stored in a table.

III. The Predator-Prey Simulation

The task is to have four hexapod robots (the predators), starting in randomly placed positions, to capture a single hexapod robot (the prey), which starts in the middle. All five hexapod robots are placed in a colony space (simulated area measuring 500x500 units). The predators must capture the prey before it reaches the edge of the colony space. All of the tests done in simulation use agents 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 and have the exact same capabilities for motion. 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. 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 either the prey or any of the predators have taken 200 steps. 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.

IV. Evolutionary Methods

A. Cyclic Genetic Algorithms

A variation on the standard GA called a cyclic genetic algorithm (CGA) was used to develop our four heterogeneous cooperative agents [10]. A CGA differs from a regular GA in that the genes of the chromosome represent tasks to be completed. The tasks can be anything from a single action to a sub-cycle of actions. 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 can contain 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 executed. The genes can be arranged into repeated sequences and a chromosome can be arranged with single or multiple cycles or even the entire chromosome can be a cycle. In the case of multiple cycles, it is possible to switch from one to the other at any point. The evolved chromosome had two separate cycles. The first cycle determines the predator's actions before it first sees the prey, while the second determines its actions subsequently. 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.

$$(((T_1\ R_1)\ (T_2\ R_2)\ \ldots\ T_9\ R_9))\ \ \mathbf{((T_1\ R_1)\ (T_2\ R_2)\ \ldots\ (T_9\ R_9)))}$$

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, which controls the agent's movement after it first sees the prey.

B. Fitness Evaluation

The fitness score of a team of predators which fails to capture the prey is the number of steps taken in the round. Because it is possible for the prey to all together 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 the score awarded for a successful capture. With an equal number of rows and columns in the simulation 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 minimum 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 minimum, the fraction (*capt-dist* / *MAX-CAPT-DIST*) becomes increasingly smaller forcing the score to elevate drastically. 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.

$$\frac{NUM - COL - ROW}{\sqrt{\frac{capt - dist}{MAX - CAPT - DIST}}} \quad (1)$$

C. PAL For Evolving a Team

Punctuated anytime learning (PAL) was developed to strengthen offline genetic algorithms by capitalizing on Greffenstette and Ramsey's [11] dynamic anytime learning approach. Although PAL cannot allow for continuous updates of the computer's models, it updates its model every *G* generations, resulting in periods of accelerated learning. The generations in which the model is updated are referred to as punctuated generations. When applied to a single GA, PAL updates the computer's model every *G* generations by running tests on the actual robot and uses these results for fitness biasing in the GA [12] or in the co-evolution of model parameters [13].

Punctuated anytime learning is a fairly different concept when applied to co-evolving separate populations to form members of a team. The updated information that each population in the learning system receives is a more accurate representation of the overall nature of the other population. For ease of explanation, assume that the experiment has two populations, population A and population B. In this case, every *G* generations 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. The chosen most fit 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 possible 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.

D. Sampling Populations

The original adaptation of PAL was to perform alpha selection at punctuated generations by testing all members of one population with all members of the other populations. This method proved to be a powerful system for evolving teams. Although it was effective, this method remains too computationally expensive to accommodate more than two populations. In order to further reduce computation time, we tested the possibility of selecting alphas using less than the entire population, a sample of the population [8]. 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 to find an alpha individual to represent population B. The selected individuals from population A are referred to as the *sample*, and the number of chosen individuals is called the *sample size*.

In further research we tested a variety of sample sizes to investigate their merits [14]. The sample sizes tested were 1, 2, 4, 8, 16, 32, and 64. In order to fully examine the behavior of the different sample sizes, it was essential to ensure that each sample size performed an equivalent number of alpha evaluations during a test run of the co-evolution. To accomplish this, we staggered the punctuated generations such that the sample 1 performed alpha selection every generation, the sample 2 performed alpha selection every other generation, and so on, such that a sample 64 performed alpha selection every sixty-fourth generation. The results of the box-pushing tests are shown in Figure 4. Each curve is the average of five separate runs of the GA and the fitness of the best pair of agents were recorded after 0, 64, 128, 256, 512, 1024, 2048, 5120, and 10240 alpha evaluations.

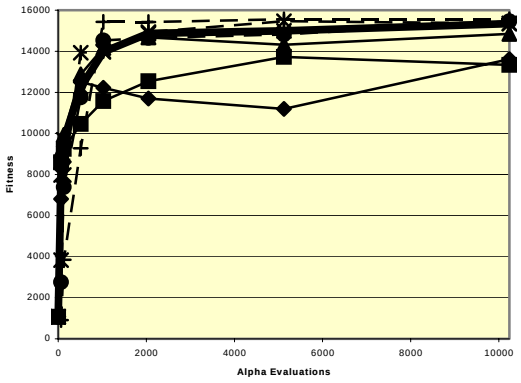

Figure 4. Results of a box pushing, shown through 10,240 alpha evaluations for sampling sizes 1, 2, 4, 8, 16, 32, and 64. Each curve is an average of five separate runs of the GA. Sample 8 is shown in hold and all higher sampling sizes are shown as a dashed line.

It can be seen from Figure 4 that the sample sizes of 8, 16, 32, and 64 on average reach a fitness of over 15000 out of a maximum possible fitness of 15625. When inspecting the earlier generations of training produced by testing the sample sizes, it became clear that the smaller

sample sizes performed better in the earlier generations while the higher sample sizes performed better in the later generations. This behavior of the sample sizes is intuitive because the smaller sample sizes select alpha more frequently causing initial accelerated growth and a larger sample size is needed for more accurate alpha selection to sustain growth for the later generations of training. Therefore, the sample sizes in the middle such as a sample 4, 8, or 16 represent good candidates for consistent growth throughout a single evolution.

To express mathematically how the system was refined to allow co-evolution of more than two populations 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 most accurate method of testing would be to compare all individuals in a population against all others in the opposing population for alpha selection every generation. A single generation of training would therefore require I^N evaluations. To reduce this level of computation, alphas are only selected at punctuated generations. If alpha selections occur every G generations, this reduces the evaluations by that factor of G. This solution cuts computation time to I^N/G. In order to further reduce computations, sampling is used. Using the previous parameters and adding the term S representing the sampling size, any given alpha selection requires only $N * (I * S^{N-1})/G$ trials. As previously stated, the most versatile sample sizes are those that sample frequently enough to provide an initial boost in fitness during the early generations, while getting a large enough sample for accuracy in the later generations. For these reasons we evolved the four predators using a sample 8 with alpha selection every 64^{th} generation. Assuming there are four populations with sixty four individuals each, a GA run with these parameters reduces computation by a factor of 128 compared to our previous method which tests all possible combinations of partners.

TABLE 1. RESULTS OF THE PREDATOR-PREY SCENARIO. THE VALUES SHOWN REPRESENT THE SCORE ACHIEVED BY THE TEAM OF FOUR ALPHA INDIVIDUALS SELECTED AT THAT PUNCTUATED GENERATION.

	0	64	128	256	512	1024	2048	5120	10240	20480
test1	43	49	58	62	70	73	71	447	479	416
test2	66	168	234	279	328	319	317	441	420	517
test3	57	113	292	257	473	483	482	513	498	528
test4	60	66	61	54	56	54	65	324	525	404
test5	54	77	50	230	120	349	460	547	483	448
test6	47	52	55	53	52	68	91	146	149	304
test7	47	54	52	82	101	133	455	516	688	653
test8	62	81	85	75	124	111	195	244	406	374
test9	48	134	256	173	234	169	190	331	338	359
test10	47	55	57	76	106	365	307	501	463	878
Avg.	53	85	120	134	166	212	263	401	445	488

V. RESULTS

To test our method we ran 10 different tests for 20480 alpha evaluations each. Table 1 shows the fitnesses of the alpha individuals selected at that punctuated generation. These scores were calculated by averaging 100 evaluations of the team of alpha individuals. Fitnesses were recorded after 0, 64, 128, 256, 512, 1024, 2048, 5120, 10240, and 20480 alpha evaluations. From looking at Table 1 we can see that our method was effective in learning the task. A fitness benchmark for an excellent solution in the simulation performance is to average a score of 500, the dimensions of the board or NUM-COL-ROW, because this correlates roughly to a capture at the maximum capture distance every round. In the predator-prey scenario it is often hard to gauge the evolved behavior simply considering fitness. Figure 5 shows a graph of the percentage of simulations during which the team of predators successfully capture the prey. The capture percentages for the 10 test runs of the GA shown in Table 1 are shown in Figure 5. The best team of predators capture the prey roughly 80% of the time, while the average capture rate of all 10 runs is approximately 65%.

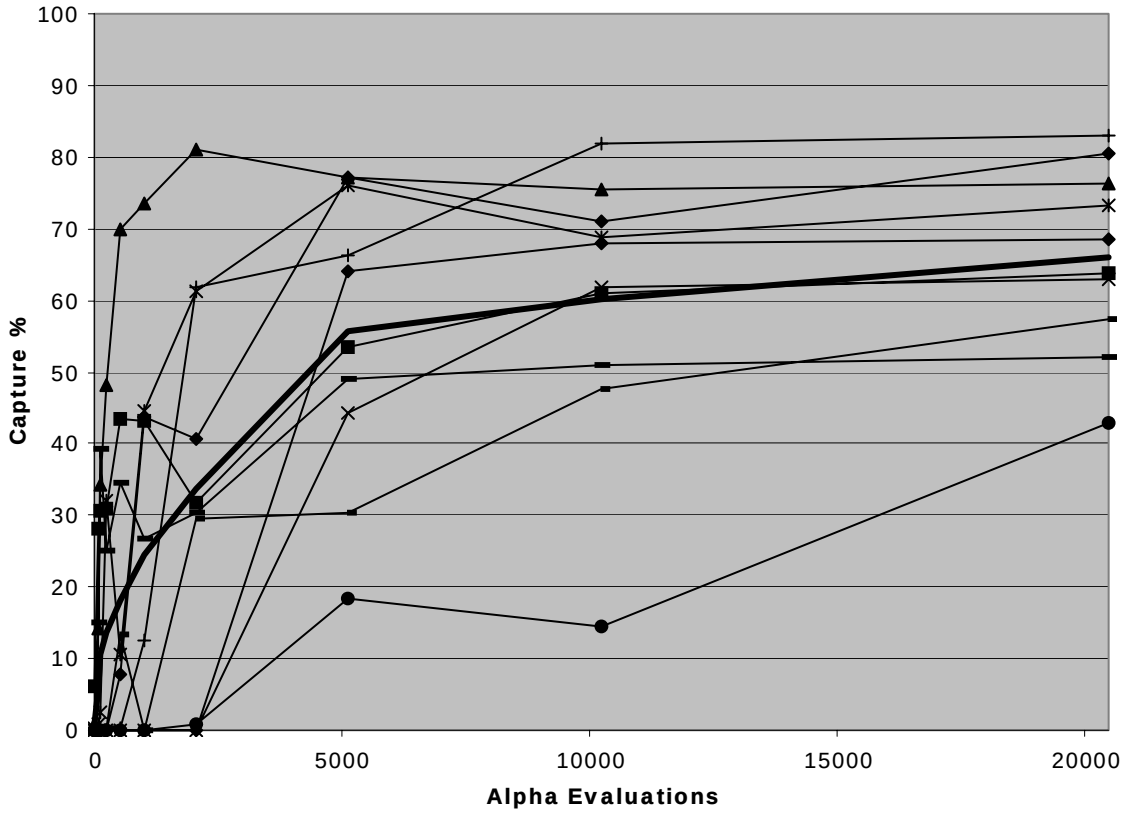

Figure 5. Results of the selected alpha individuals after 0, 64, 128, 256, 512, 1024, 2048, 5120, 10240, and 20480, alpha evaluations. The lines represent the number of successful captures of the team out of 100 chances. The average is shown in bold.

It is also important to examine the level of cooperation in the capture strategies. From studying the final solutions from the ten test runs of the GA we discovered that the most influential factor in deciding the capture strategy was the starting positions of the predators. We reasoned that this was because the prey can escape the non-bounded-box environment in only 22 gait cycles, forcing the predators to formulate a capture strategy highly dependent upon their initial placement. All of the evolved strategies employed at least three of the four predators to direct the motion of the prey. Figure 6 shows two different simulated runs of the GA. Each of the four predators are labeled with different letter ranging A-D and the prey is labeled with a P. The starting positions of all the agents are enclosed in a square and a solid line is drawn through predators when they first see the prey; the same is done for the prey when it first sees any of the four predators. The four predators show a white dot when they can see the prey. Each of the simulations was stopped a few gait cycles before the actual capture to clearly illustrate the coordinated strategy. Figure 6a is an example of a capture involving only three of the four predators. All four of the predators start on the same side of the prey and the prey has been randomly assigned to start moving away from their starting positions. Predator B chases the prey such that the other predators A and D can trap it from either side. The simulation shown in Figure 6b shows a four predator capture strategy. In this evolution the predators were assigned favorable starting positions and they all converge on the prey. The capture strategy shown is particularly interesting because after the prey is initially chased toward the bottom of the simulation, predator D halts its motion to slow the prey and direct it toward position where it is completely surrounded by all four of the predators. Though the capture strategies of the team vary greatly depending upon the starting positions of the simulation, the punctuated anytime learning method was able to adapt accordingly.

VI. Conclusions

The intent of our research is to show that the punctuated anytime learning method can evolve solutions for problems involving three or more populations and that it is applicable to the general class of learning

heterogeneous behavior. The method of sampling populations with a sample size of eight reduces the computation required by a factor of 128 and produces a highly accurate solution. The sampling method was very successful in evolving teams of predators that cohesively capture the prey. In future work we hope to observe these emergent behaviors on the ServoBots in a colony space at Connecticut College. We also hope to evolve coordination strategies using robots with different capabilities to test the level of specialization of behaviors.

REFERENCES

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

[2] Luke, S. and Spector, L., "Evolving Teamwork and Coordination with Genetic Programming," *Proceedings of First Genetic Programming Conference*. (1996), 150-156.

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

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

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

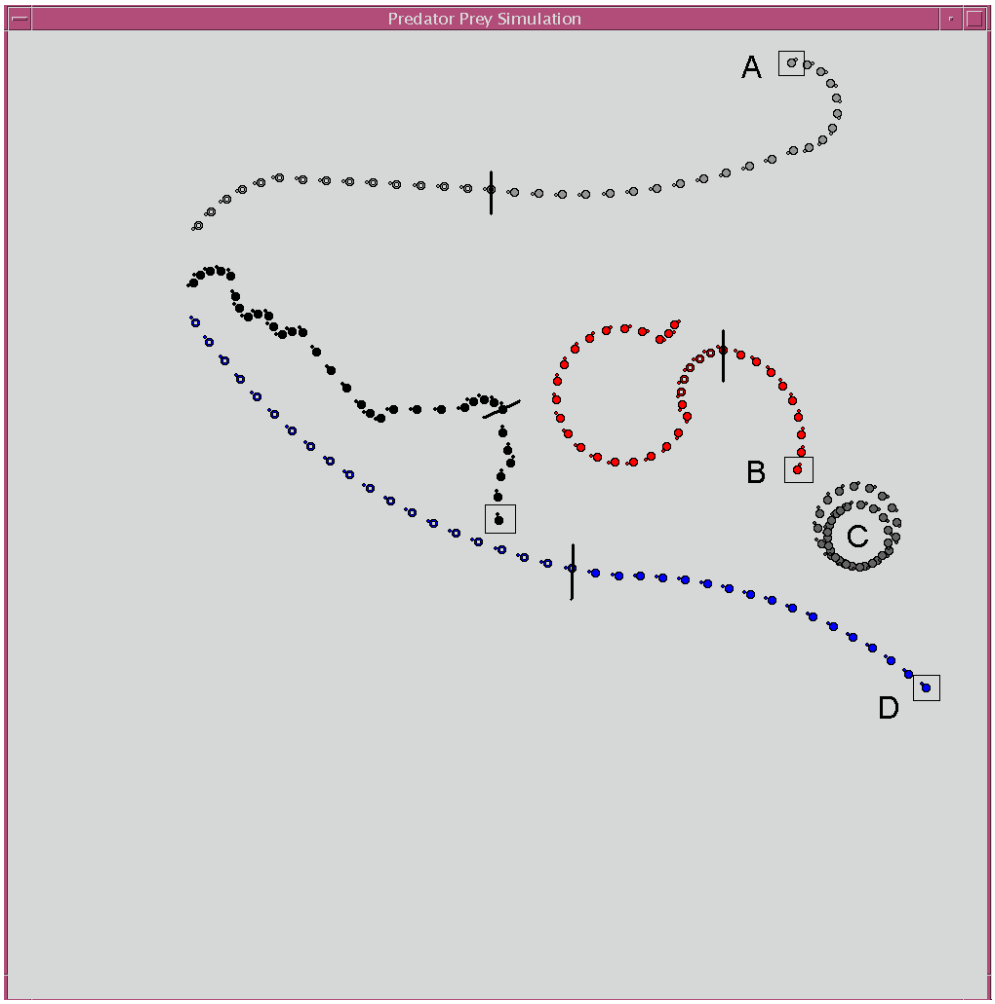

Figure 6a. A capture involving three predators

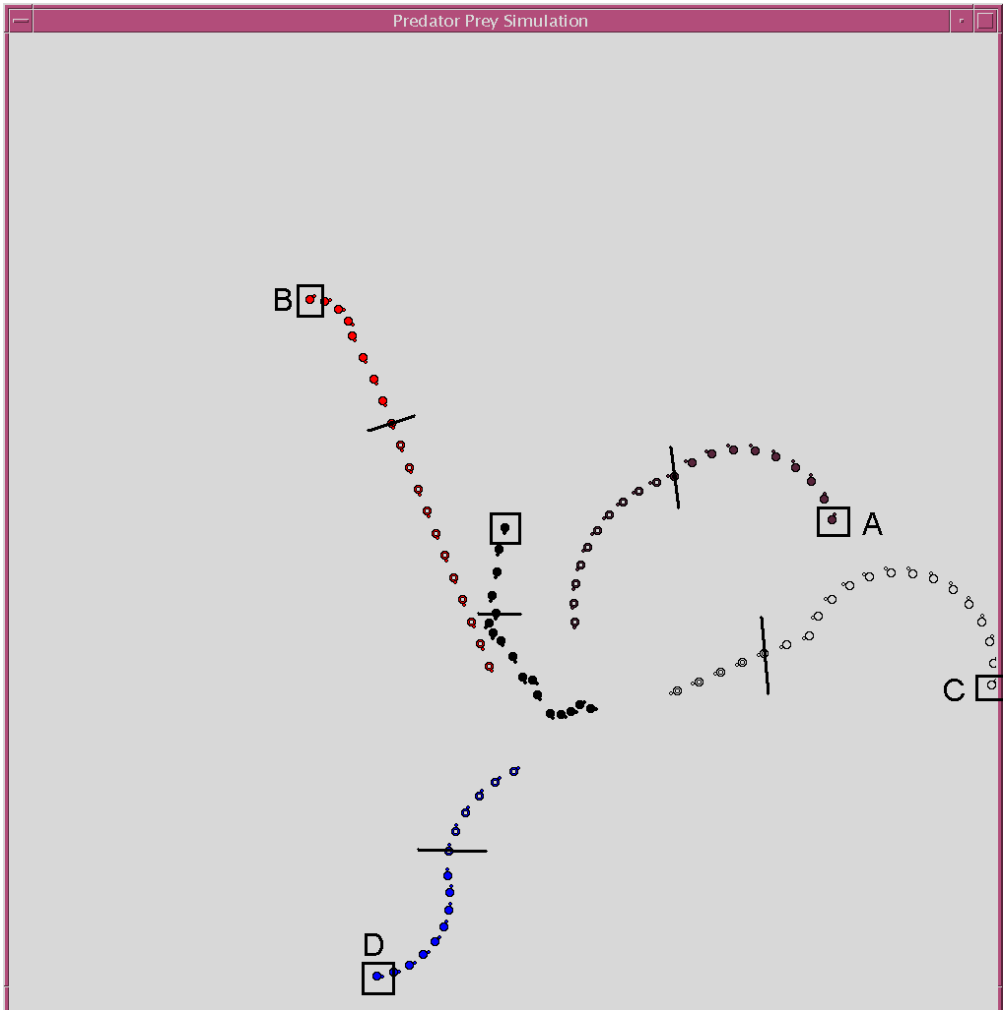

Figure 6b. A capture involving all four predators that surround the prey.

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

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

[8] Parker, Gary B. and Blumenthal, J., "Sampling the Nature of A Population: Punctuated Anytime Learning For Co-Evolving A Team," *Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE2002, Vol. 12)* (2002), 207-212.

[9] Parker, Gary B., "Evolving Cyclic Control for a Hexapod Robot Performing Area Coverage," *Proceedings of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2001).* (2001), 561-566.

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

[11] Grefenstette, J. J. and Ramsey, C. L., "An Approach to Anytime Learning. *Proceeding of the Ninth International Conference on Machine Learning*, (1992), 189-195.

[12] Parker, Gary B. and Mills, Jonathan W., "Adaptive Hexapod Gait Control Using Anytime Learning with Fitness Biasing," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99).* (1999), 519-524.

[13] Parker, Gary B., "Co-Evolving Model Parameters for Anytime Learning in Evolutionary Robotics," *Robotics and Autonomous Systems*, Vol. 33, Issue 1, (2000) 13-30.

[14] Parker, Gary B. and Blumenthal J., "Comparison of Sampling Sizes for the Co-Evolution of Cooperative Agents," *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003).* (2003), 536-543.