SAMPLING THE NATURE OF A POPULATION: PUNCTUATED ANYTIME LEARNING FOR CO-EVOLVING A TEAM

GARY B. PARKER
Connecticut College
New London, CT 06320

H. JOSEPH BLUMENTHAL
Connecticut College
New London, CT 06320

ABSTRACT
Evolving agents to function as cooperative members of a team is a difficult problem. In order for individuals in a team to best display heterogeneous behavior, their evolution must take place in separate populations to promote specialization. To allow the genetic algorithm to find a robust solution, the fitness evaluations must ensure that each individual is paired with team members that best represent the nature of their partners' populations. In previous work, we employed a method of punctuated anytime learning to ensure the integrity of the fitness evaluation. This was very successful, but was still considered to be too computationally expensive. In this work, we use a sampling method to maintain the quality of the solutions while significantly decreasing the time for computation. We chose a box pushing task to show the success of our method.

INTRODUCTION

The objective of our work is to develop a robust and computationally inexpensive method of co-evolving two separate populations to produce cooperative behavior in two specialized members of a team. 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 (1996) researched methods of increasing specialization and cooperation in teams. They considered the whole team as a single genetic programming individual, instead of evolving the agents in separate populations.

Potter and De Jong (1994) focused on methods to optimize co-evolution in their paper that was a theoretical discussion of ways to break down a cooperative task into parts and evolve a population for each sub-task. The method described in their work called cooperative co-evolutionary algorithms (CCAs), allows heterogeneous control systems to evolve by defining an efficient means of computing an individual's fitness. The individuals of each population are each tested while teamed with the best fit individuals from the generation of the other populations. This method of selecting one individual from each population with which to evaluate each member of a population allowed each population to develop specialized individuals of the teams. In addition, this method insured that an individual is not discounted by getting teamed with a poorly evolved partner. Although successful in completing the task and an excellent method for evolving cooperative teams, the CCA method still limits each individual's fitness calculation to being computed with a partner that was selected by comparison with only one individual from the opposing populations.

Wiegand, Liles, and De Jong (2001) took a more in depth look at the factors that effect the co-evolution of a team. In their work they examine collaborator selection issues. Their conclusion was that the most influential factor on the success of the co-evolution is the collaboration pool size. They also noted that as the collaborator pool size increases, so does the computational cost of the evaluations. Taking this to the extreme, the most successful co-evolution would be to test each individual against every other possible partner. A task requiring *N* partners, with *I* individuals in each population of the genetic algorithm, would require $N \times I^N$ comparisons at any given set of fitness evaluations; not an acceptable solution.

In previous work in Evolutionary Robotics, we developed Punctuated Anytime Learning (PAL), a method to allow for the learning system to be periodically updated throughout simulated evolution (Parker and Larochelle, 2000). The computer's model in simulation is updated or the GA fitness evaluation altered by measuring the robot's actual performance at certain consistently spaced number of generations called punctuated generations and entering those values into the GA. This same concept of PAL was adapted to be applicable to evolving cooperative teams (Parker and Blumenthal, 2002). This method used the periodic nature of PAL to minimize the number of fitness evaluations required to evolve team members in separate populations. As already discussed, 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 our method selected a single individual from each population as the best representative of the nature of their population. This selected individual, refered 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 our method of PAL, with N generations between each round of alpha selection, the number of fitness evaluations are reduced by a factor of approximately N. Our results showed that this method was very successful at co-evolving separate partners. However, it was still considered to be too computationally intense for teams of more than two members.

In this paper we suggest a computationally cheaper method of periodically finding a single individual to represent the nature of a population. Instead of using the whole population as in previous work, this new method uses a sampling of individuals to find the new alphas. This proved to be a successful method to decrease computation time and maintain accuracy in the solution.

PROBLEM DESCRIPTION

The task is to have two hexapod robots starting from one corner of an enclosed square area walk to and push a box that is in the middle of the area to the opposite corner. The scenario from which the task has been abstracted is a colony space in the Connecticut College Robotics Lab. The colony space is approximately an 8x8 ft area. In this area, the two ServoBot robots and a square cardboard box can be placed. The problem is for the pair to act cooperatively to force the box into the opposing corner from which the robots started. The tests, done in simulation, use agents that model actual robots.

The robots simulated in the experiment are ServoBots. These are inexpensive hexapod robots with two servos per leg, one oriented in a vertical capacity and the other oriented in a horizontal capacity, giving two degrees of freedom per leg. The ServoBot is controlled by a BASIC Stamp II, which is capable of individually addressing each of the twelve servo actuators (two on each leg) to produce and sustain a gait cycle. A gait cycle 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. The BASIC Stamp is capable of storing a sequence of timed activations to be repeated. A single activation represents the simultaneous movement of the twelve servos. Different degrees of turns can be generated by decreasing the number of activations producing thrust sent to one side of the robot (Parker, 2001). The effect of these 15 left turns, 15 right turns, and a straight applied to the robot were measured and stored in a table.

The simulated environment used for evolving the agents was an abstraction of the colony space in the lab. The simulated area measured 250x250 units. Both robots were represented as circles with a diameter of 6 units but the robots were treated as single points for rules of contact with the box. The box was represented as a square measuring 18 x18 units. In each trial, both the robots and the box were placed in consistent starting positions. The first robot started on the point (10,5) and faced parallel to the x-axis, while the second robot started in the mirrored position (5,10) but faced parallel to the y-axis. The box started in the middle of the environment at the point (125,125).

Each robot's ability to push the box on its own (without aid from its partner) was affected by an endurance factor. The endurance factor starts at zero and increases with each consecutive non-aided push. With F representing the would be full force of the robot push acting singly, and E representing the endurance factor, the force the robot may apply to the box is given by the quotient $F/2E$ (with $E = 0$ the force is F). This cuts their pushing power in half after each gait cycle. Whenever as both robots push the box simultaneously, both of their endurance factors are reset to zero. In the simulation, both robots move simultaneously, and a trial ends when either each robot has taken 200 steps or one of the three (the two robots or the box) moves out of the simulated area.

METHOD

A type of evolutionary computation called a cyclic genetic algorithm (CGA) was used to develop our two heterogeneous cooperative agents (Parker, 2001). A CGA is similar to a regular GA, except that in the CGA the 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. The CGA was used for evolving our agents because it is designed for learning cyclic behavior and it allowed for our incremental learning approach. The set of actions to get each agent to the box was one cycle and each agent's behavior after touching the box was defined in the second cycle of their CGA chromosome. During a trial, as soon as a robot touches the box, the controller would switch from the first to the second cycle, at the completion of the current gait cycle. 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.

Individuals were selected stochastically for breeding based on their fitness score and standard operators were used for the CGAs. The evolution was done in two different stages. The incremental learning approach was employed because the problem can be easily broken down into two smaller tasks with the first requiring no cooperative behavior. For the first increment, two completely separate populations were evolved; one for each robot. They were evolved using an identical method except for the robot's starting positions. For population A, the starting point was (10,5) facing down the x-axis, the fitness of an individual was either the value of the box's y coordinate after the trial finished or zero if the individual failed to move the box positively in the y direction. For population B starting at (5,10) facing down the y-axis, the individual's score was computed the same as for an individual in population A, except the robot was charged with moving the box positively in the x-direction to receive a non-zero score. Only the first cycle of the CGA chromosome was evolved during the first increment of learning. A population of these chromosomes learned for each team member during this first increment was used to evolve each team in the second increment. The first cycles remained unchanged while the second cycles for each chromosome were randomly generated to create start populations for the second increment of learning. The second increment of the learning process was more complex, involved team coordination, and required co-evolution using PAL and a CGA. The fitness score of a team was decided by the product of the box's final (x,y) coordinate position, or zero if the team failed to move the box toward the target corner of the area in the x or y position from the box's starting point (125,125).

When PAL is applied to co-evolving two populations the updated information that each population receives during the learning is a more accurate representation of the overall nature of the opposing population. Assume that the experiment has two populations, population A and population B. In this case, every N generations, some chosen number of individuals in population A are tested against all individuals in population B. The chosen individuals from population A are refered to as the sample, and the number of chosen individuals is called the sample size. The purpose of this process is to find the fittest individuals 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 best method of evolution would therefore be to select new alpha individuals for each generation. However the process of selecting the two alphas requires significant computation. Assuming there exists I individuals in each population and the sample size is S, the computer must perform $2 \times (I \times S)$ trials for each selection of the pair of alphas. In order to avoid that level of computation, new alpha individuals are only selected at punctuated generations.

RESULTS

When first considered sampling sizes, it was thought that for populations of 64 individuals, a good sampling rate would be the square root of 64. The graph Figure 1, shows a comparison of sampling rates of 64 and 8, with new alpha selection every 100 generations. Each data series is the average of five separate runs of each sampling size at selected generations. The dashed line represents the sample 8 runs, and the solid line represents the sample 64 runs. The graph shows that the sampling size of 64 is the most accurate method. Though the sample 64 tests outperformed the sample 8 tests, the graph shows that the sample 8 is capable of producing a viable solution. The experiments with the sample 8 runs showed that it was an effective method to both develop emergent behavior and reduce computation time,

Further tests were done to consider several sampling rates. The frequency of the punctuated generations was increased to every 20. Figure 2 shows a comparison between the sampling sizes of 2, 4, 8, 16, 32. Each data series plotted is a specific sampling size's performance averaged over five separate tests. The best fitness at specific generations is shown for each sampling size. The lines on the graph represent the sampling rates in ascending order starting from bottom to top, with the flattest learning curve as the sample 2 and the sharpest learning curve as the sample 32. The data series shown in bold is the sample 8 test. As can be seen from the graph, there is a gap in performance between the sampling rates of 2, 4, and 8. However, a difference between the success of the sampling rates of 8, 16, and 32, is only present at the 300th generation. Therefore, we believe that the sampling rate which best strikes the balance between sampling size and computation time is 8.

CONCLUSIONS

The results from experiments involving PAL with population sampling and CGAs are very encouraging. The original method of evolution with a sampling size of 64 (the whole population) yielded a near optimal solution for the box pushing task. The goal of minimizing computation time by reducing the sample size used for alpha selection while maintaining the accuracy of the solution was successful. Through experiments involving comparisons between different sampling sizes, we conclude that the sample size of 8 is well suited for efficiently producing a near optimal solution for this application.

REFERENCES

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

Parker, Gary B. and Blumenthal, J., 2002, "Punctuated Anytime Learning For Evolving A Team." *World Automation 2002 Congress Proceedings.*

Parker, Gary B., 2001, "Learning Control Cycles for Area coverage with Cyclic Genetic Algorithms." *Proceeding of the 2nd WSES International Conference on Evolutionary Computation,* pp. 283-289.

Parker, G. B. and Larochelle, K. J., 2000, "Punctuated Anytime Learning For Evolutionary Robotics." *Proceedings of the World Automation Congress (WAC2000), Volume 10, Robotic and Manufacturing Systems*, pp. 268-273.

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

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

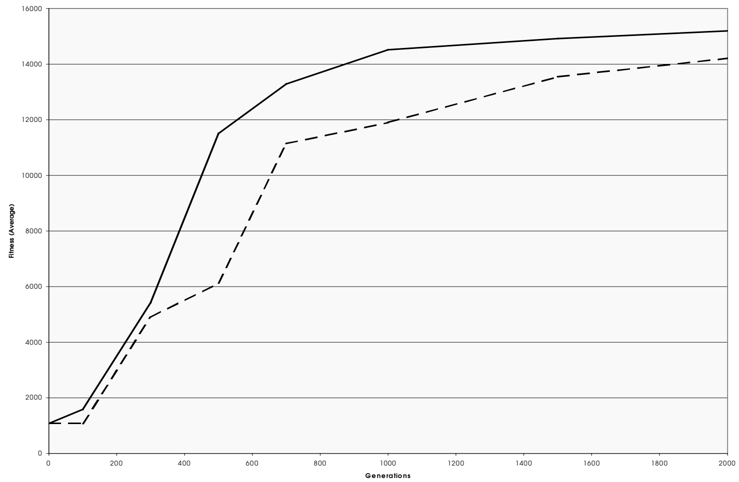

Figure 1: Comparison between sampling rates of 64 (solid line) and 8 (dashed line) for the box pushing task. The average fitness of the best team for each sampling rate is shown. The best fitness from five runs for each sampling rate was recorded at generations 0, 100, 300, 500, 700, 1000, 1500, and 2000.

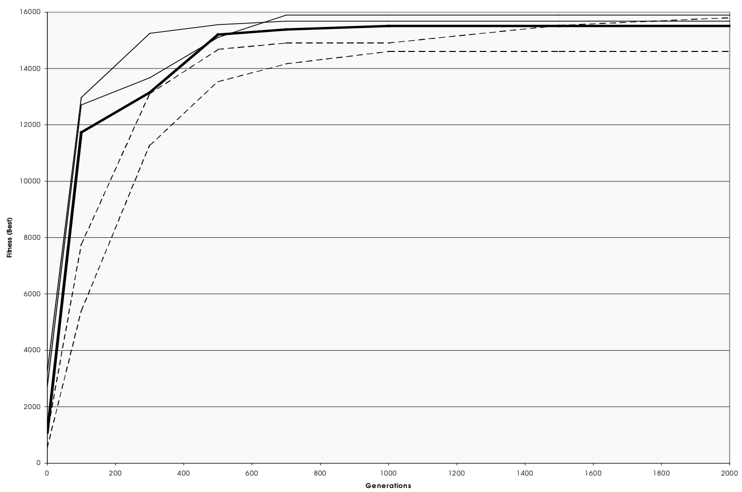

Figure 2: Comparison between sampling rates of 2, 4, 8, 16, 32, with alpha selection every 20 generations. The sampling rate of 2 and 4 are dashed lines, 8 is shown in bold.