Enhancing Embodied Evolution with Punctuated Anytime Learning

Gary B. Parker, *Member IEEE*, and Gregory E. Fedynyshyn

***Abstract*—This paper discusses a new implementation of embodied evolution that uses the concept of punctuated anytime learning to increase the complexity of tasks that the learning system can solve. The basic idea is that there is one population of chromosomes per robot rather than one chromosome per robot and reproduction between robots involves a combination of two entire populations of chromosomes instead of the recombination of two single chromosomes. The embodied evolution with punctuated anytime learning system is compared with embodied evolution alone and evolutionary computation alone, as the three methods are used to solve a common problem. The results show that this new learning system is superior to the other methods for evolving colony robot control.**

I. Introduction

THE concept of multi-robot systems is important to the field of robotics because multiple robots can outperform individual robots in terms of quality and efficiency and can perform tasks that single robots cannot. For example, when surveying an area, advantages to having a colony of robots are that a team of robots can survey the area in parallel, the mission would not necessarily fail of one or more robots fail, and teams of small inexpensive robots can be less expensive than one very expensive robot [1]. Some examples of tasks that are performed efficiently by multi-robot teams are large area searching, cleaning of hazardous waste, and object transportation.

Evolutionary computation (EC) is a powerful tool, borrowing concepts from heredity and natural selection, for solving a wide range of problems, such as optimization or classification. When applied to robotics, EC is proficient at evolving behaviors that would be laborious to program and offers a means of adaptability for robots. However, there are issues concerned with the use of EC as a learning system for robots.

One of the big issues with using EC to learn robot control is where the learning should take place. Learning on a simulation of the robot in its environment [2] is the fastest, but requires a very accurate simulation and does not naturally allow the system to adapt to chances in the robot. Evolving the solution entirely on the robot [3] requires no simulation, but takes significant time and energy since every potential solution needs to be tested on the robot. A third method [4] does most of the learning in simulation, but allows for the later generations to be done on the actual robot. Even with fewer generations tested on the robot, there is much loss of time and energy and the robot still has a very limited system for adapting to changes.

To deal with the issue of where the evolution should take place, *punctuated anytime learning* (PAL) was developed [5,6]. The learning is done on a simulation, with periodic tests performed on the actual robot. In this way, the tests on the robot are minimized while allowing the learning system to adapt to changes in the robot's capabilities and the environment. PAL was shown to be effective in adapting learned gaits for a hexapod robot where the robot's capabilities were changing over time.

In this paper, we consider the use of EC to learn behaviors for a colony of robots. There has been interesting previous work in this area. Wu, Schultz, and Agah used EC to learn control for a colony of micro air vehicles. The main focus of their work was to develop a system to learn rule sets for controlling the behavior of a team of micro air vehicles that were continuously conducting surveillance of an area [1]. The learning mechanism used EC to evolve a rule set based on a simulated environment, which could later be transferred to the actual vehicles.

In an effort to have a system where the evolution can take place on the robot as opposed to in simulation, a group from Brandeis University developed the concept of *embodied evolution* (EE) [7,8]. The idea behind EE is to have a large population of robots, which are able to reproduce (i.e., share genetic information) with one another, evolve in their task environment without the help of EC running on a simulated model. Embodied Evolution is defined by the Brandeis group as "evolution taking place within a population of real robots where evaluation, selection, and reproduction are carried out by and between the robots in a distributed, asynchronous, and autonomous manner" [7].

They evolved an artificial neural network on each robot to search an area for a light source. When any two robots in the task environment come within a certain range of each other, they would transmit their genetic information to each other. As in traditional EC, the fitness of a robot was determined by how well it performed its task. However, with EE, fitness (better referred to as energy) is constantly changing. As a robot searched the task environment, its energy was slowly decremented. Whenever a robot successfully found the light source, its energy was increased substantially. The more proficient the robot was at finding the light source, the higher fitness it attained.

Manuscript received March 7, 2007.

G.B. Parker is the Director of Computer Science at Connecticut College, New London, CT, USA (e-mail: parker@conncoll.edu)

G.E. Fedynyshyn is a student in Computer Science at Connecticut College (e-mail: gefed@conncoll.edu).

The concept of EE was a good use of the existent colony of robots to act as the individuals of a population of evolving solutions. However, EE as originally described is not practical for the complicated tasks that an actual colony of robots will be employed to accomplish. A colony of robots starting with randomly generated control solutions for a difficult task would use significant time and energy before they even approached a useful level of productivity. In addition, problems that are difficult to solve typically take large populations of individuals for the evolution to be effective. In the traditional EE model, each robot contains a single chromosome. Offspring are produced when two robots' genes are crossed to make new chromosomes for the next generation. In the previous works presenting EE [7,8], eight robots were used to complete the relatively simple task of finding a light source. Although effective with this simple task, we believe EE is not suitable for solving complex problems due to the EC requirements for large population sizes and slow initial learning curves.

In order to use the positive aspects of the EE concept without the burden of contending with small population sizes necessitating simple problems, we employ the concept of PAL. Evolution takes place in simulation for each individual robot. However, as robots reproduce in an EE sense, based on their actual performance, entire populations of chromosomes are used to produce a new population for the mating pairs' offspring. In this case, EC on a simulation can produce reasonable solutions in the initial learning phases and EE reproduction will strengthen the solutions produced based on the robots' actual performance. The result is a learning system that can produce immediately productive colony individuals that will continue to learn and adapt to changes.

To test this concept, a contrived environment of eight robots learning reasonable search patterns to find food was developed. Tests were done to compare standard EE, EC running in eight separate populations on each robot, and EE with PAL. Tests in simulation confirm the validity of this method.

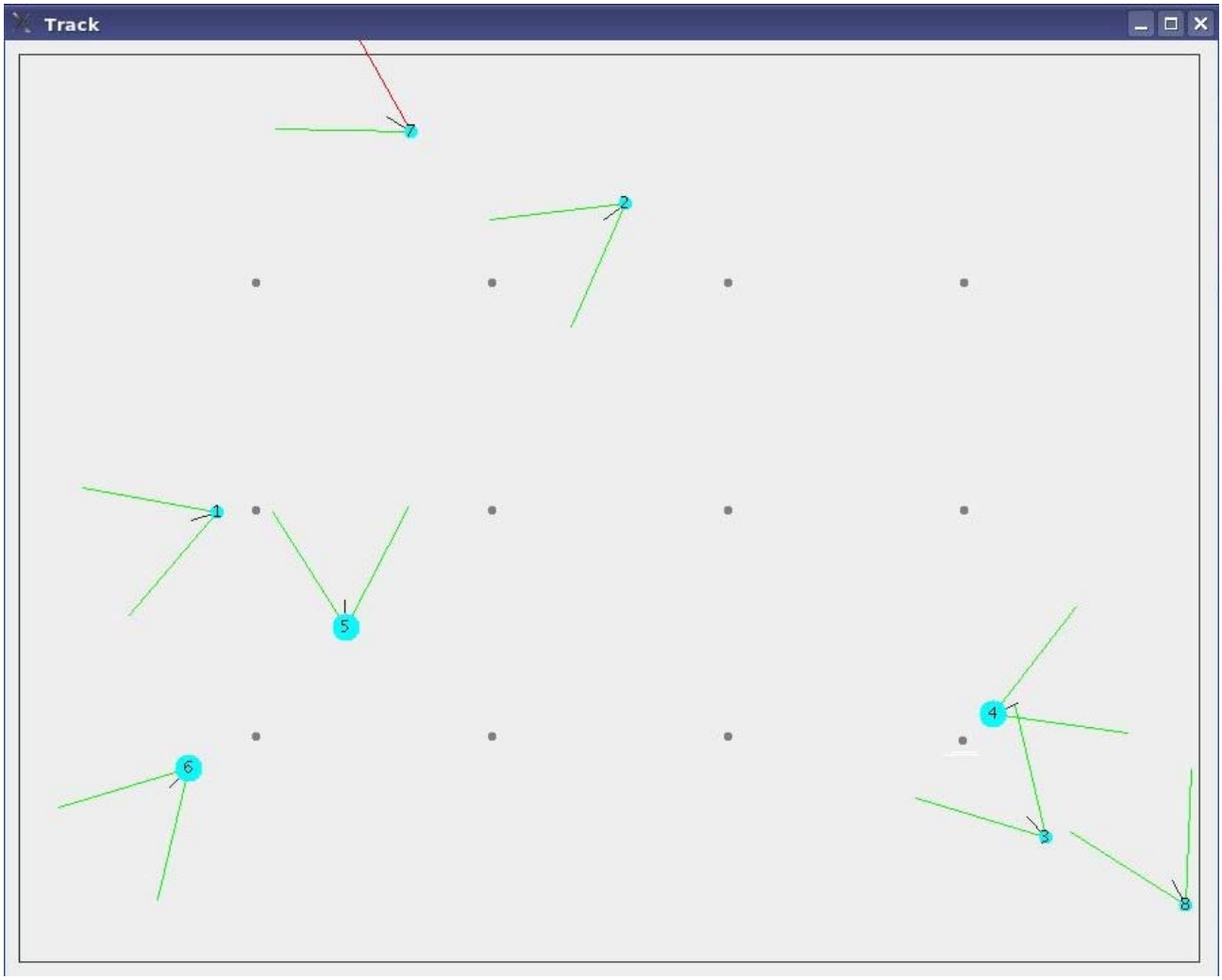

Fig. 1. The task environment. The 12 food locations are shown as dots. The robot symbols are numbered, show direction of heading and sensor location, and grow in size as their fitness (energy stored) grows.

II. Search for Stationary Prey

The problem or task that the learning system was to solve needed to be at a complexity level that would show a distinction between the methods. The task selected was area coverage. Stationary prey (deposits of food) were placed in an evenly distributed pattern throughout the environment, which was a 870 by 670 pixel rectangle. The food was placed in only 12 locations and once eaten, took time to slowly regenerate. This forced the robot to learn an effective area coverage pattern to find sufficient food for survival. Each robot had its own distinct food supply, but all the food was in the same twelve locations. This ensured that all of the robots were learning a similar task that was complicated enough to show the distinction in the three methods. The task environment is shown in Figure 1. Each robot, as it eats, stores energy, which it burns as it moves around in the environment. The robots are shown with their sensor spans.

As a visual aid, robots with more energy are displayed as being larger. Each robot has two lines drawn in front of it, 30 degrees to either side of the direction it is facing, which represent vision sensors. Robots can only see a short distance (100 pixels). The robots are able to distinguish whether the left, right, or both sensors are seeing either prey or a wall. The left and right sensors both have a range of 45 degrees and vision overlaps by 15 degrees in the middle (Figure 2).

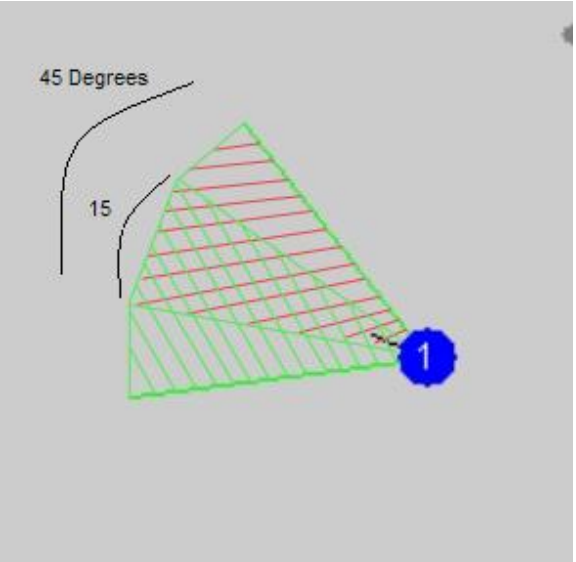

Fig. 2. Vision Sensors

The robots are faced with a "search for prey" scenario, where robots are forced to evolve efficient search patterns to prosper in the task environment. Once a prey is found, it remains inactive for a certain period of time, forcing robots to develop a search pattern that allows them to cover a large area in a small amount of time. The prey is spaced equally throughout the task environment to encourage the robot to evolve to search the entire task environment for prey. The movement of each robot is modeled after that of a hexapod robot (ServoBot) using actual measurements. Turn rates, taken from the ServoBot as it performs various turns, are used to define the movement capabilities of the simulated robot.

III. Cyclic Genetic Algorithms

A *cyclic genetic algorithm* (CGA) is a variation on the standard genetic algorithm [9] model, which is one of the primary methods of EC. The CGA uses a chromosome to define a series of actions rather than to define characteristics of a solution [10]. Standard GAs typically use a chromosome to define characteristics, such as speed or turn rate of a robot or the weights of a neural network that the robot uses to perform its task. A CGA differs in that its chromosomes are made up of actions to be completed (such as control instructions). The genes are usually made up of two parts, the first defines an action to perform, and the second defines how many times to repeat that action. For example a simple cyclic chromosome could have four genes, each defining an action and a number of repetitions for each action. When the robot runs, it will loop through its chromosome and continually perform the four actions until the robot is stopped.

CGAs can be setup so that different parts of the chromosome are looped through in a cycle depending on conditional statements (CGA with conditional branching [11]). For example, if a robot sees a wall to the left, control moves to the part of the gene that defines the actions and repetitions for the robot when it sees a wall to the left. The robot will then act based on that part of the gene until the condition is no longer true, then it will go on to act based on the section of the chromosome that defines actions for when no walls are seen.

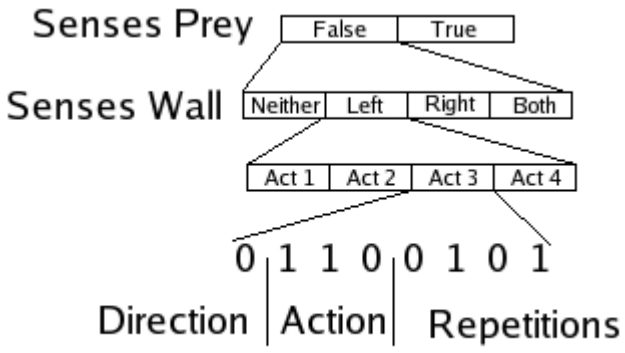

Fig. 3. The CGA chromosome used.

A CGA with conditional branching was used to evolve the robots in this research. The chromosome is split up into eight loop segments, each representing a different condition (Figure 3). At the top layer, the condition pertains to whether the robot senses prey or not. If it does not sense prey, it determines what sensors are sensing a wall. Depending on this, it will go into one of four cycles (loops) where it will continually repeat four genes (action/repetition pairs) until the sensor state changes.

Each of the four genes contains a direction bit, which dictates whether the robot is moving forwards or backwards; three action bits, which correspond to different degrees of turn; and four repetition bits, which define how many times the movement is repeated before moving to the next gene. The robot will continue to loop through the same part of the chromosome until the condition changes (sensor input changes). The final chromosome was 256 bits long.

IV. The Application of Embodied Evolution with Punctuated Anytime Learning

As in PAL, each robot contained a population of chromosomes (in this case 32), which were constantly being evolved in simulation as the robots performed its function using the most fit chromosome in its population. Because this research is simulating how the EE would work with real robots, the internal GA that tests fitness based on a simulated model does not perfectly reflect the movements of robots in the real world. To simulate this slight inaccuracy, when evolving with the GA, values for movement and direction change were rounded so as to not match the actual values used by the robots for movement. The GA working on the simulation would send its results to the robot, which would then perform EE in the task environment and improve the simulated results to solve the problem of simulations being intrinsically inconsistent with the real world.

The fitness tester for the GA worked by placing the robot in the environment with 12 food items, then letting the robot run for 1000 steps, or iterations. Each robot started with an energy level of 350, with each movement decreasing the energy by 1. If a robot found a food item, its energy was increased by 250. At the end of the 1000 steps, the fitness was defined to be the robot's average energy level averaged doubled over 10 of these trials. The GA then evolved using standard crossover with a mutation rate of 0.03.

To test EE with PAL, modifications were made to the simulation. For a pair of robots to be able to transmit genetic information, in other words, in order for the Embodied GA to work, several conditions had to be met by both robots. Both robots had to be within a short range of each other. Both robots also must have not recently exchanged information with any other robots. This "breed timeout" was added to ensure that robots only exchanged information once instead of continuously, as robots may be within range for several iterations. Since breeding decremented both robots' energy by 100, both needed to be able to support such a loss (i.e. both must have an energy value of more than 100).

The chance of two robots breeding upon being in breeding range was dependent on the energy levels of the two robots so that a robot with a high energy would be more likely to breed with a robot with low energy and less likely to breed with a robot with high energy. The chance of breeding was determined probabilistically by comparing the energy level of the higher energy robot with the lower energy robot. One minus the fraction of the lower energy value over the higher energy value is the probability any two robots will breed when they come within breeding proximity of each other, thus allowing better solutions to resist having their population changed while encouraging weaker solutions to exchange genetic information with other robots. If the two robots did not succeed in breeding, both would remain unable to breed for a period of time so as to not repeatedly try and mate with the same robot as long as they stayed within breeding range of each other.

Once all the conditions were met, the two robots would combine their populations of chromosomes to make a single population of 64 chromosomes. The 64 chromosomes would be evolved on the simulated model using standard GA techniques, including crossover and mutation, ensuring the two populations are thoroughly mixed together. Out of the two robots, the one with the higher energy would keep its population while the one with lower energy would be given the 32 most fit chromosomes from the population of 64, the fitness would be determined as is described above, by running the robot for 1000 steps over 10 trials and doubling the average energy. The new robot would then have its energy level reset to 350 to simulate the birth of a new robot.

V. Results

Three tests, each with five runs, were performed to determine the validity of EE with PAL. In the first test, each robot ran a CGA that evolved its own behavior. In the second test, EE, using a single chromosome on each robot, was used. In the third test, EE with PAL was used.

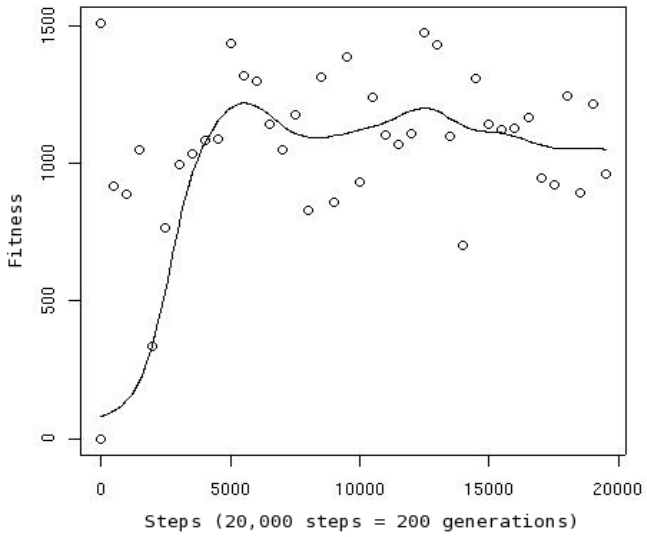

Figure 4: Results of CGA with no EE between robots.

Each test was run for 20,000 steps. For the normal GA, this means that the 32-chromosome populations on each robot were independently evolved for a total of 200 generations, as the learning is being performed in the background while the

robots perform tasks based on their current solution in the task environment. To simulate this background learning, the GA evolved one generation every 100 steps. For the EE, it means that the robots are allowed to roam the environment for 20,000 steps, with EE occurring asynchronously whenever two robots came within breeding distance of each other. For EE with PAL, this means that the robots roam their task environment performing EE asynchronously whenever two robots come within breeding range of each other for 20,000 steps, during which, the populations on the robots are updated every 100 steps by the PAL. While the robots are evolving, the learning system saves the fittest robot controller for each of the eight robots every 500 iterations (steps).

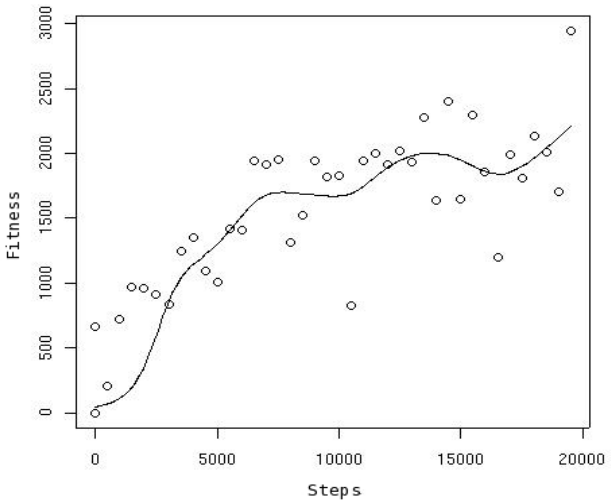

Figure 5: Results of EE with one chromosome per robot.

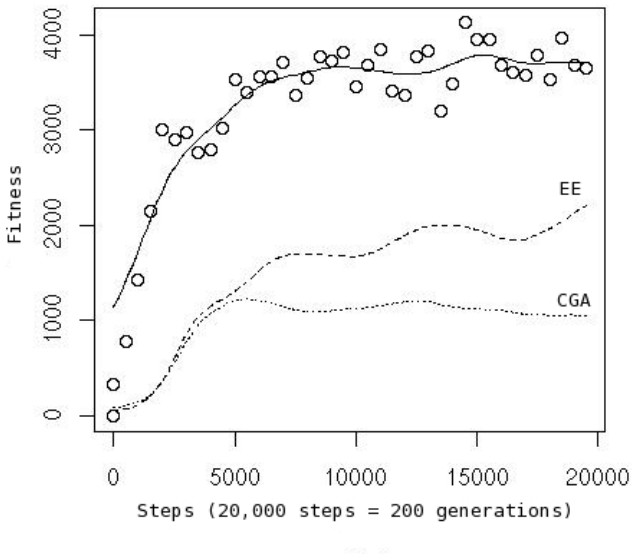

Fig. 6. Result of combined EE with PAL compared to with the other two methods.

Graphs showing the average improvement in fitness over time of the five runs for each test were produced. They include the Nadaraya-Watson kernel regression estimated line to show the general trend of fitness over time. Both the lone CGA (Figure 4) and the lone EE (Figure 5) show some degree of improvement. However, the EE with PAL graph (Figure 6) shows that, not only did using both together provide a solution with the highest overall fitness, but that the results were more consistent than the other methods (this can be seen by comparing the distances of the actual points from the regression line).

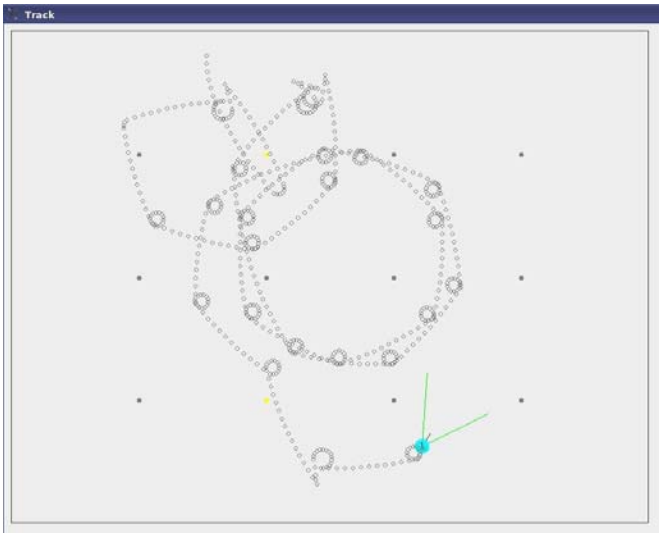

Fig. 7. Path of robot using best chromosome found after 20000 steps using the CGA alone.

Fig. 8. Path of robot using best chromosome found by EE with one chromosome per robot after 20000 steps.

Observations of the paths of the robots using the evolved controllers confirm that the EE with PAL produced superior control programs. The robots evolved using the CGA or EE alone tend to search a smaller total area while the EE with

PAL robot was able to cover much more ground and find more prey. Typical paths are shown in Figures 7, 8 and 9. In the CGA-only model, the robot moves in a circular pattern that includes many small loops, which do not accomplish much in terms of searching, but waste energy (Figure 7). The path found using EE with one chromosome per robot (Figure 8) was able to find a solution similar to the CGA running alone in the sense that it moves in a primarily circular search path. In addition it managed to smooth out the tiny, useless loops. However, the total area covered by the robot is more or less as limited as the solution when using the CGA by itself.

The results found using a combination of EE with PAL (Figure 9) showed a more robust search path, covering more ground over less time than either of the other two solutions. Instead of looping around the same area over and over, it is able to search the entire area. The previous two solutions also leave many prey items completely untouched, which the combination of the EE with PAL only miss a few of the total prey items. Had the prey been placed randomly rather than at specific, evenly-spaced locations, the solution using the EE with PAL would likely perform even better than the solutions using either EE with one chromosome per robot or using the CGA without any EE.

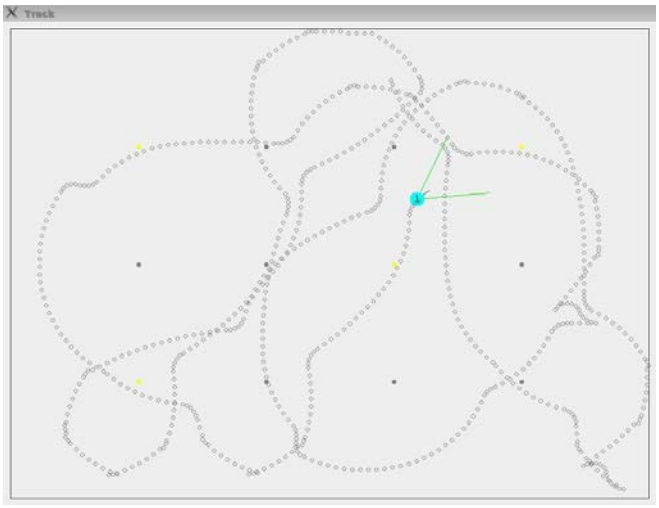

Fig. 9. Path of robot evolved using EE with PAL.

VI. Conclusion

Using a combination of embodied evolution with punctuated anytime learning on a CGA produces better results than using either EE or the CGA by itself and allows for the asynchronous and autonomous properties that are characteristic of embodied evolution. This new learning method for autonomous robots combines the strengths of EE and PAL to produce a means of evolving controllers for a colony of robots as they perform complex tasks.

The work presented in this paper was intended to show the benefit of this system without the complications of robot interaction. In future work, the tests will be altered to simulate an actual colony environment where a group of robots are working to perform a similar task. Examples would be jobs such as moving supplies from one location to another while avoiding obstacles or searching for and gathering objects in a specific location. Further research will involve tests on a colony of actual robots. The robots in the simulation were based on actual robots so as to minimize the effort required to accomplish this expansion. Although tested using a CGA for the form of EC, we believe that this method is equally viable for any form of EC which is being used to evolve controllers for robots working in a colony.

References

[1] A. Wu, A.Schultz, Alan C., and A. Agah, (1999). Evolving Control for Distributed Micro Air Vehicles, *Proceedings of the IEEE 1999 International Symposium on Computation Intelligence in Robotics and Automation*, Monterey, CA, 1999.

[2] W.-P. Lee, J. Hallam, and H. Lund, (1997). Applying Genetic Programming to Evolve Behavior Primitives and Arbitrators for Mobile Robots, *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Indianapolis, IN, 1997.

[3] F. Mondada and D. Floreano, (1995). Evolution of Neural Control Structures: Some Experiments on Mobile Robots, *Robotics and Autonomous Systems* 16, 1995, 183-195.

[4] O. Miglino, H. Lund, and S. Nolfi, (1995). Evolving Mobile Robots in Simulated and Real Environments, Technical Report, Institute of Psychology, CNR, Rome, 1995.

[5] G. Parker, (2000). Co-Evolving Model Parameters for Anytime Learning in Evolutionary Robotics, *Robotics and Autonomous Systems* 33, 2000, 13–30.

[6] G. Parker, (2002). Punctuated Anytime Learning for Hexapod Gait Generation, *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*. October 2002

[7] S. Ficici, R. Watson, and J. Pollack, (1999). Embodied Evolution: A Response to Challenges in Evolutionary Robotics, *Proceedings of the Eighth European Workshop on Learning Robots*.

[8] R. Watson, S. Ficici, and J. Pollack, (2002). Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots. *Robotics and Autonomous Systems* 39/1, 2002, 1-18.

[9] J. Holland, (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[10] G. Parker and G. Rawlins, (1996). Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots, *Proceedings of the World Automation Congress (WAC '96), Volume 3, Robotic and Manufacturing Systems*. May 1996.

[11] G. Parker, I. Parashkevov, H. Blumenthal, and T. Guildman, (2004). Cyclic Genetic Algorithms for Evolving Multi-Loop Control Programs, *Proceedings of the World Automation Congress (WAC 2004)*. June 2004.