Incremental Evolution of Complex Light Switching Behavior

Megan Schuster
Swarthmore College
500 College Avenue
Swarthmore PA 19081
megan@cs.swarthmore.edu

Chris Harman
Swarthmore College
500 College Avenue
Swarthmore PA 19081
charman1@cs.swarthmore.edu

Abstract

Evolving a robot to perform a specific simple task has been shown to be effective (Stanley and Miikkulainen, 2004) (Harvey et al., 1994) (Marocco and Nolfi, 2006). What eludes most artificial evolutionary robotics systems is the next step: how can we make task-specific intelligence 'bootstrap up' to harder tasks and higher levels of intelligence? In this paper, we use an incremental evolutionary approach to evolve complex behaviors in a fixed-topology neural network control structure. Our robot is given the static task of maximizing the amount of light seen in its environment by locating and flipping on light switches. We use four environments of varying complexity in terms of the number and locations of the light switches. We evolve robots from scratch in the complex environments and compare their performance to robots which were evolved first in simpler environments and then moved to the more difficult environments. We show that pre-evolving robots in simpler environments can sometimes improve performance in more complex environments. However, we find that it is not always obvious what kinds of pre-evolution steps will be most helpful in evolving more complex behaviors.

1 Introduction

Evolutionary robotics is a rapidly-growing field which takes a cue from biology in its efforts to create intelligent artificial agents. Rather than trying to design complete, robust, and adaptive agents by hand, evolutionary robotics instead allows these agents to emerge over time through gradual mutation and the reproduction of successful agents. Typically, the entity being evolved is a neural network controller for a robot body, and mutation operates by making small, random changes to the network's "genes", which are the weights that define the network's behavior. The evolutionary process begins with a population of robots whose neural network brains are initialized with randomly selected weights. Some fitness function must be defined to quantify how well any given agent behaves. Robots that behave well according to this fitness function may then pass their genes on to the following generation, with some probability that those genes will be mutated. Over time, robots that behave poorly will be weeded out of the population, and the behaviors of the more adept robots will be fine-tuned by the mutation process.

The evolutionary approach is appealing because it allows desirable behaviors to emerge without the need for intensive engineering of the robot's control structures. Instead, we simply design a fitness function that will favor the kinds of behaviors we would like to see and wait for evolution to take its course and produce agents that behave appropriately. However, for very complex tasks, the time required to evolve agents with the desired behavior may become prohibitive, and in some cases we might never see evolution lead to a good solution.

Stanley and Miikkulainen (2004) have designed the NEAT architecture to deal with precisely this problem. They point out the difficulty of choosing an appropriate network architecture that will best solve a given task. Their system allows robots to evolve complex behaviors by starting with very simple networks and gradually, over evolutionary time, increasing the complexity of those networks. While a simple network may be able to come up with simple strategies to solve the task, by allowing the network topology to become increasingly more complex, their robots have the ability to evolve increasingly sophisticated solutions to the problems they have been asked to solve.

In this paper, we too seek to evolve robots which are capable of completing increasingly difficult tasks, effectively bootstrapping their abilities from simpler tasks. Unlike Stanley and Miikkulainen, however, we choose to use a fixed-topology network and gradually increase the complexity of the task the robot is asked to solve. Our hope is that robots which are evolved to solve easier tasks before they are evolved to solve more complex tasks will find more effective solutions for the complicated tasks and do so in less evolutionary time than robots which begin evolving immediately on the complicated tasks.

Harvey et al.(1994) have previously applied a similar incremental approach to evolving a vision system. Their robot is given the task of tracking a white target in an otherwise dark environment. They begin by using a large target and evolving robots which can successfully complete the task. They then give these evolved agents a smaller target, and once the robots have evolved to find that , they give them a small, moving target. Harvey et al. succeed in evolving robots that can track the moving target using this incremental technique. However, the question of whether the incremental approach provides any advantage over simply evolving the robot from scratch on a moving target is left unanswered. In this paper, we attempt to address this question of the relative advantages and disadvantages of an incremental evolutionary approach.

2 Methods

We conduct several experiments in which we seek to evolve robots with the ability to intelligently turn on light switches. We have four environments containing one or two light switches which can be placed either toward the center of the environment or in the corner (Figure 1). The difficulty of the environment increases with the number of switches it contains, and we consider switches in the corner to be more difficult to find and turn on than switches in the middle of the room. We evolve robots in each of these environments, beginning "from scratch", using randomly initialized neural network weights. For the more complicated environments, we then also allow robots to pre-evolve in a simpler environment before placing them in the more complex environment. We hope to see that, for more complex environments, allowing the robots to pre-evolve on a simpler task allows them ultimately to evolve more sophisticated solutions in less time for the more complex tasks.

2.1 Experimental Setup

For our experiments, we used a simulated Pioneer robot within the Pyrobot simulator [1]. There is some degree of simulated noise in the robot's sensors, helping to maintain a degree of simulated realism. We borrowed code from several of Pyrobot's preexisting modules for our project as well: the Simple Recurrent Network (SRNBrain.py) and the Genetic Algorithm control code (GA.py).

The environment is very simple for each of our experiments: a walled box along with one (or two) light switches (Figure 1). We defined a light switch to be a square tile on the floor of the environment with a light inside. When the robot rolls onto the tile, the light is toggled depending on the light's current state (on to off or vice-versa). It is important to note that the light does not change state if the robot leaves the light switch; only the robot entering the light switch has an effect on the light. This setup is intended to produce interesting reentry avoidance behavior from the robot. [2]

[1]http://pyrorobotics.org

[2]The structure of our environments was inspired by previous work by Marocco and Nolfi (2006), who also sought to evolve robots to visit tiles on the floor of their environment in an intelligent way. Marocco and Nolfi, however, use auditory signals between multiple robots in the environments to signal which

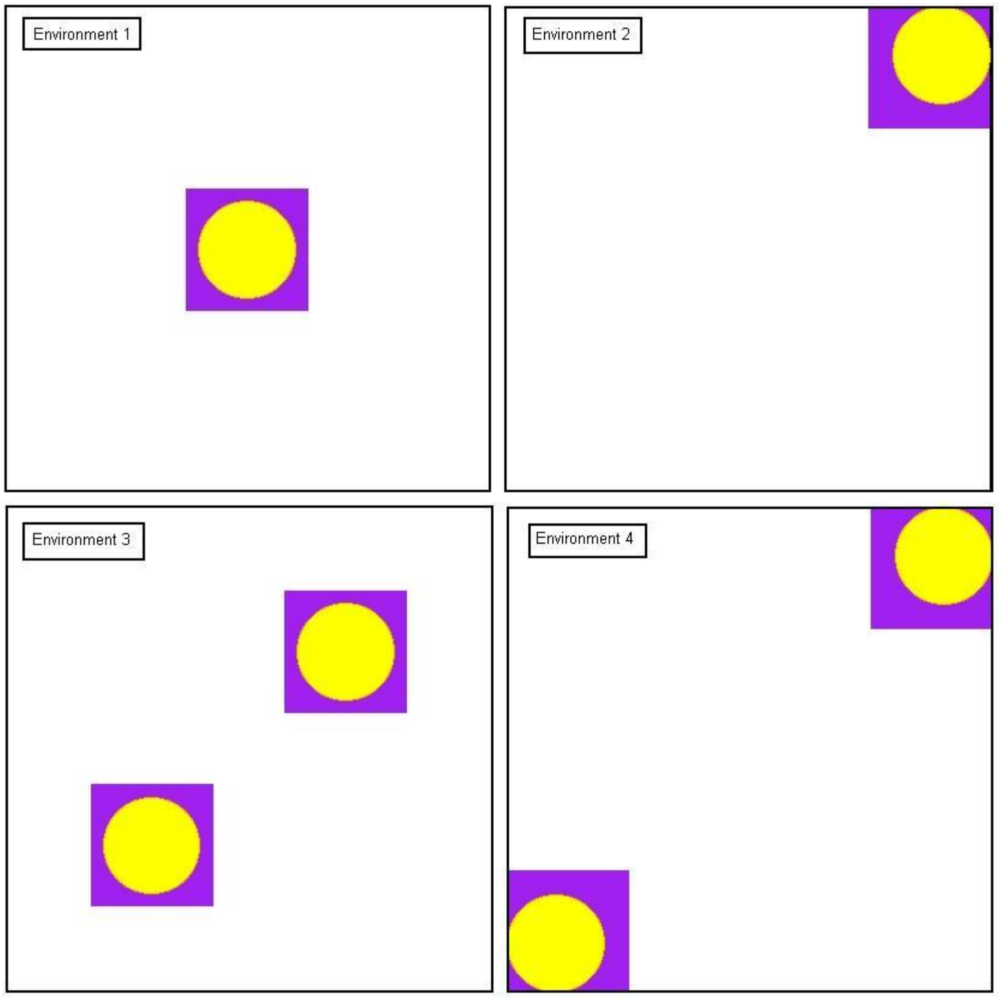

Figure 1: The four simulated environments. Light switches are represented as the square tiles on the floor. In these images, all lights are turned on, as indicated by the yellow circles at each light switch.

The simulated Pioneer robot is equipped with 16 360 degree range sensors oriented radially outward from the robot. These 16 range sensors were conflated into four (front, left, right and back) by grouping them appropriately and taking the minimum of each group at every time step. Also, the simulated robot has two front light sensors that detect the amount of light in the environment. We added a switch sensor to the robot as well that allows the robot to tell if it is standing on top of a light switch or not. The robot's initial placement in the environment is random for each run.

2.2 Recurrent Neural Network Brain

We chose to use a recurrent neural network brain for our robot controller after initial experiments with a standard feed-forward network failed to produce any complex behaviors. A recurrent neural network has connections between the output nodes and hidden nodes, so there is a degree to which the hidden nodes 'remember' their previous action. The input layer for our robots consists of seven nodes and takes as input one value for each of the range sensor groupings (front, left, right and back), one value for each of the two front light sensors, and one value for the switch sensor. The hidden layer consists of three hidden nodes and the output layer has two nodes to control the robot's motion (translation and rotation). The network's weights are randomly initialized for the first generation of the evolutionary process and serve as the genotype over which the genetic algorithm operates.

2.3 Defining the Fitness Function

The choice of fitness function is arguably the most critical component of defining the experiment because it inherently drives the robot's evolutionary path toward a specific behavior. For example, we did not include a stall component in some of our earliest experimental fitness functions and the robot would at times flip the light switch on, only to crash soon after into a nearby wall. This type of behavior would yield high fitness because the amount of light seen in the environment would be maximized, but the robot's exploration of the environment would be severely hindered. Consider the effects of propagating such a negatively biased generation in terms of qualitative behavior to the next generations. The final evolved genotype would likely incorporate the wall-smashing behavior into its behavior, which could never succeed in environments with more light switches where exploration is of paramount importance. As such, we chose our fitness function carefully in hopes of generating effective light-switching behavior. Our fitness function is defined as follows: $(notStalled) \times (|translationVal| + 2 \times (lightSensed))$

The first term in the fitness function, $|translationVal|$, was chosen because we wanted to reward the robot for exploration. We had previously experimented with including the minimum sensor value in this term also, but that had the effect of limiting the robot in its exploration of the environment as it preferred to stay far away from walls. Because our robot's main objective is to maximize the amount of light in its environment, the second term in the fitness function, $2 \times (lightSensed)$ is defined so that the robot will get a large reward for flipping on new light switches. Both of these are multiplied by 0 if the robot's stall sensors are activated, so that a robot receives zero fitness points for every step it spends crashing into a wall, regardless of the number of lights on in the environment. An effective environmental exploration behavior that explores novel portions of the environment–a behavior where it will not reenter a light switch once it has been switched on and will seek out new light switches–is, in essence, the goal of our fitness function.

2.4 Evolutionary Algorithm

We used a genetic algorithm for evolving the robot's ability to effectively flip light switches and maximize the amount of light in their environment. The genetic algorithm uses the weights of the neural network as its genotype and uses a moderately aggressive evolutionary scheme. We used a population size of 20 for each generation. From those twenty, we chose the top 20% (the top four) as elite. Each of the elite members is passed on directly to the next generation. Each elite member also makes four copies of itself, which are passed on to the next generation subject to a 2% mutation rate. This means that

tile should be visited, where we use light as a signal to indicate which tile has already been visited and which remain to be visited.

each gene in the genotype has a 2% chance of being mutated, and the amount of mutation possible is bounded at 1 unit. These parameters were chosen because they were used in previous work by Marocco and Nolfi (2006) which uses an environment similar to ours. We experimentally verified the ineffectiveness of using crossover for mutating neural network weights and chose not to include it as a result.

2.5 Pre-evolution Steps

Harvey et al. (1994) apply the incremental evolutionary approach by evolving a population in a simple environment, then moving that same population to a more complex environment. We have concerns that by allowing our population to converge on a solution that is very particular to the simpler environments, there may be so little diversity in the population that it will hinder their abilities to then evolve more sophisticated behaviors in complex environments. As such, we modify the incremental approach of Harvey et al. slightly.

When we have a population that has been evolved on a simple task that we would like to move to a more complex task, we do not simply transfer the population to a more complicated environment. Instead, we choose the top four individuals in the population at the end of the pre-evolution step and make four copies of each. As before, we allow one copy to pass through unmutated; the other three are subject to the same 2% mutation as before. This leaves four empty spots in the population, which we seed with randomly initialized agents. This ensures some degree of diversity in an otherwise converged population.

All pre-evolved populations used in our experiments were allowed to evolve for 100 generations before advancing to a more complicated task, where they were then allowed 100 generations to evolve in the complex environment. Likewise, control groups which were evolved from scratch in the complex environments were evolved for 100 generations. For all experiments, we run five trials of our evolutionary algorithm and report the average resulting fitness, since the results of the evolutionary process are somewhat sensitive to the initial weights and starting positions of the robots.

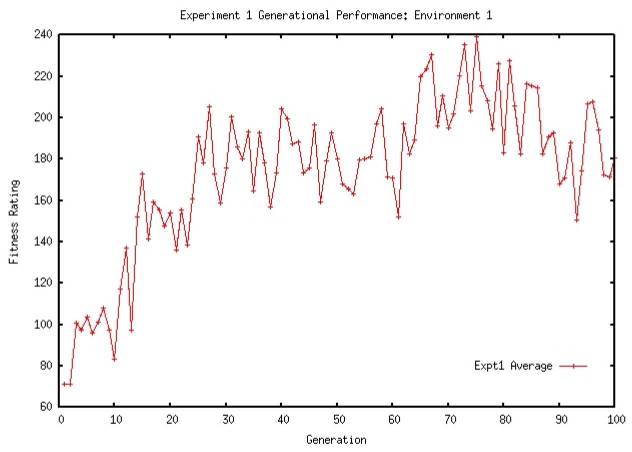

Figure 2: Average fitness versus generation number for robots solving a two-switch task in environment 1 with no pre-evolution, evolving from randomly initialized weights. These fitness curves represent the average fitness at each generation over five trials.

3 Results

3.1 Environment 1

3.1.1 Beginning from Randomly Initialized Weights

By watching robots which have evolved for 100 generations in this environment, we can make some generalizations about the behaviors the robots have evolved over the course of this experiment. For all trials of this experiment, agents have clearly evolved an ability to avoid crashing into walls. This is as we expected, since robots are awarded zero fitness points for every time step at which its stall sensors are activated. This is a skill which will clearly be useful to the robot when it faces more complicated tasks.

The robots also seemed to have evolved to “want” to have the light turned on. Occasionally, we see a robot turn on a switch and later roll back over the switch, turning the light back off. In such cases, the robot will often change its direction of motion in order to reenter the switch a third time and thus turn the light back on. Such behaviors indicate that evolution has indeed produced agents which are interested in seeing high values for the light sensors.

Different trials of this experiment lead to different ultimate strategies for solving the one-switch problem. In some trials, we see the robot find the switch and then stay on top of it, wiggling back and forth.

This ensures that the light always stays on, but at the possible cost of some fitness points for keeping the robot's translation value small at every step. In other trials, we see the robot go to the middle of the environment to flip the switch, then back out of the switch and begin going in circles around it. This strategy awards more points for large translation values at every step as the robot circles the switch, but it also allows the robot to occasionally roll over a corner of the switch and turn the light off, risking the loss of some fitness points for temporarily having the light switched off.

In Figure 2, we see that the robots' fitness score plateaus at about 175 fitness points around generation 30. To put this fitness value in perspective, the maximum possible fitness score is 600 points for a robot who has the light on and translates by the maximum value at every step of the run. Of course, it is unlikely that this would actually be possible for any robot, so a more realistic upper bound would be about 375 fitness points, which is the approximate score of a robot who finds the switch within 3 seconds of the beginning of a run and leaves it on throughout the run, going in large circles around its environment once it has turned the lights on. On the other hand, a robot which goes in large circles with the light *off* throughout the run gets a fitness score on the order of 30. Our average fitness score hovers around 175 for all higher generations, which is quite a bit lower than the score of 375 for the optimum performance just described, but is clearly better than the case where the light is never found. This seems to be due to the fact that it often takes robots considerably longer than 3 seconds to find the switch, depending on their starting position. As such, the fitness for any given run is highly dependant on the initial conditions, leading to an average fitness of any given population which is intermediate between the optimum switching behavior and the no-switching behavior.

3.2 Environment 2

3.2.1 Beginning from Randomly Initialized Weights

As in environment 1, robots which had evolved in this environment for 100 generations had clearly learned to avoid running into walls. Beyond that, though, we did not see any clear evolution of an effective strategy for finding the light switch, as the robot seems to just wander more or less aimlessly in its environment. We noticed that, in general, the robot seems to spend much of its time near walls, often approaching a wall and then wiggling back and forth in front of it for several seconds. This may be because of the fact that the switch is located in a corner, so evolution will favor robots who take a special interest in walls. For example, should such a robot be initialized very near to the switch, it may approach the switch and then move back and forth on top of the switch for most of its run, giving it a very high fitness score and thus favoring the propagation of this wall-dancing behavior into future generations.

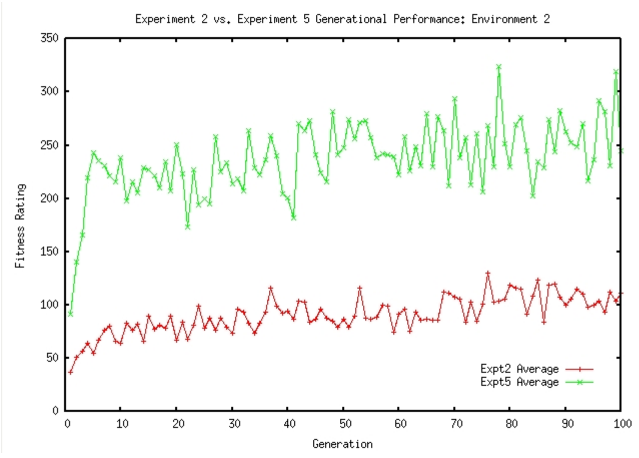

Figure 3: Average fitness versus generation number for robots in environment 2 which were evolved from scratch (red) and were evolved first in environment 1 (green). Fitness curves represent the average fitness at each generation over five trials.

In Figure 3, we see that the fitness function for this experiment is gradually increasing throughout evolutionary time and seems to be hovering around 100 fitness points by the 100th generation. This is likely due to the fact that, while the robot can move around a lot in its environment and gain fitness points that way, whether it finds the light or not seems to be mostly determined by its initial position and the random chance that it will stumble onto the switch. This means that most individuals do not find the light at all, leading to a somewhat lower average fitness score than was achieved in the previous experiment in environment 1. This is a sensible result, as we would expect the robot to be less likely to discover a switch when it is tucked into a corner than when it is right in the middle of the room.

3.2.2 Pre-evolved in Environment 1

When we pre-evolved the robot in environment 1 before evolving it in environment 2, we found that after 100 generations of evolution in environment 2, the agents seemed much more purposeful than in the previous experiment in which agents were evolved from scratch. Where before the robots appeared to be wandering somewhat randomly, here the robots explore the environment in a way that much more consistently leads to discovery of the switch. The most common strategy evolved in this experiment was a wall-following approach; wherever the robot was initialized, it would approach the nearest wall, then follow that wall until it hit the switch. This seems an effective strategy since the robot has no way of determining which corner has the switch until it is actually on top of this switch. Once a robot had found the correct corner containing the switch, it would often avoid that corner with the light in the future as it continued to wall-follow, though sometimes it would still return to the switch and turn the light back off.

From Figure 3, we can see that robots that were pre-evolved in environment 1 were at a clear advantage over those which evolved from scratch. The effect takes place immediately, as even the first generation's fitness level is doubled by allowing pre-evolution in environment 1. Over the first 5 generations, fitness increases dramatically for the pre-evolved agents. After that, fitness seems to somewhat level off, though it still oscillates quite a bit. By the 100th generation, the fitness level for the pre-evolved agents is oscillating around 250 fitness points, which indicates that the light must be on for good portions of the run for most of the robots in the population.

3.3 Environment 3

3.3.1 Beginning from Randomly Initialized Weights

In this two-switch environment, the robots seemed to have evolved fairly effective behaviors even when evolved from randomly initialized weights. For all trials of this experiment, the robots seemed to have adopted the same basic strategy by

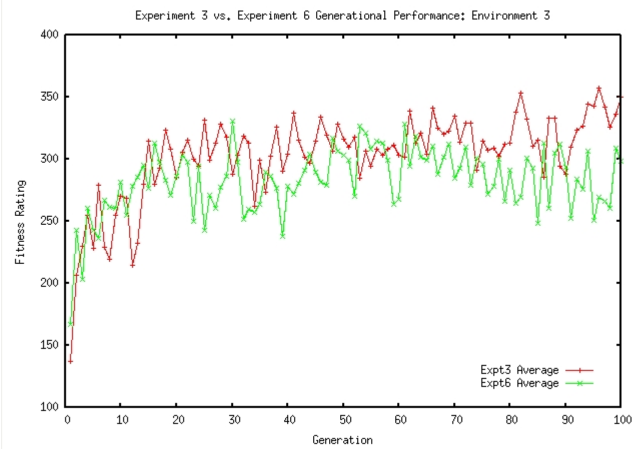

Figure 4: Average fitness versus generation number for robots in environment 3 which were evolved from scratch (red) and were evolved first in environment 1 (green). These fitness curves represent the average fitness at each generation over five trials.

the 100th generation, which was to move in a circle around the center of the environment such that both switches were in its path. In general, this strategy allowed robots to turn on both lights, but sometimes the robot would have enough time to make more than one complete circle in its environment, allowing it to revisit a switch it had already seen. Rather than recognizing that the light was on and avoiding revisiting it, robots would typically continue on their circular paths, switching the lights back off. This is obviously not the type of behavior we were hoping to see in this experiment. This usually happened near the end of the run, such that the light would only be off very briefly before the run was over and thus have only a small effect on the fitness function. Perhaps if the runs were longer, turning off a light which had previously been turned on would have had time to make a greater impact on a robot's fitness, providing an evolutionary incentive for robots to avoid revisiting switches it had already turned on.

In Figure 4, we see that the robot reaches a fitness plateau by about generation 30, and from then on oscillates around a fitness level of about 325. To get a sense of the meaning of this fitness level, recall that the optimal behavior in the one-switch environment lead to a fitness score of about 375. When observing 100th generation robots, we saw that the most effective agents, who quickly turned both lights on and kept them on for the rest of the run, were getting fitness scores of about 600. Our average robot in this environment is almost as good as our best robot in a one-switch environment but clearly is not exhibiting optimal behavior for a two-switch environment. Based on our observations of the robots, this is not because they simply turn on one light and stop there. Instead, 100th generation robots typically find and turn on both switches by the end of the run, but often it takes a good deal of time to find the first switch such that the second switch is only on for a few seconds before the end of the run, leading to average fitness scores that are not much higher than what we can expect from the best agents in a one-switch environment.

3.3.2 Pre-evolved in Environment 1

When allowing the robots to pre-evolve in environment 1 before placing them in environment 3, we saw no detectable difference in the strategy ultimately adopted to solve the two switch problem; as in the case where robots were evolved from random initial weights, the pre-evolved robots seemed to discover the simple circular-path strategy described above. Also as before, these robots did not seem to learn how to avoid reentering switches that they had already turned on. From Figure 4, it seems that pre-evolving the robots in this case made no difference in the fitness level the robots ultimately achieved or the number of generations required to reach that level.

3.4 Environment 4

3.4.1 Beginning from Randomly Initialized Weights

Environment four represents the hardest environment for the robot to learn because of the fact that there are two light switches in the environment and that they are both placed at opposite corners of the environment. Because of the similar structure to the experiment where the robot was evolved in environment two (essentially the environment is copied and flipped along a $y = -x$ diagonal), we would expect the graphical results of this experiment to match closely the results shown for the robot evolved from scratch in environment two (Figure 5), except for the fact that the fitness score should double because the robot's chances of randomly exploring and finding a light switch double early on in the evolutionary process. In fact, this is exactly what happens.

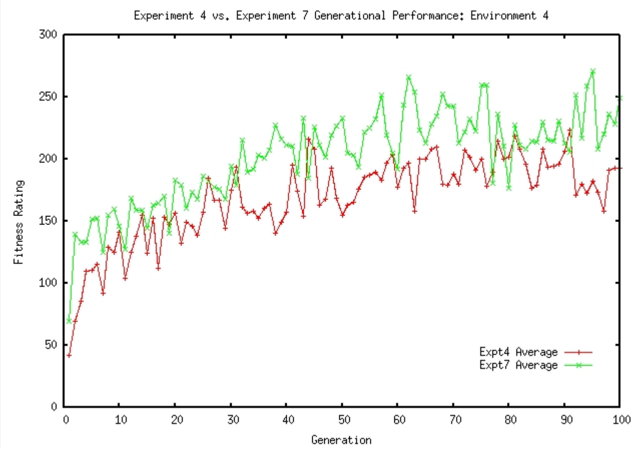

Figure 5: Average fitness versus generation number for robots in environment 4 which were evolved from scratch (red) and were evolved first in environment 1 (green). Fitness curves represent the average fitness at each generation over five trials.

After 100 generations, the evolved robot exhibits light-switching behavior that is initiated by a semi-elliptical exploration of the environment until the first light switch it toggled on. The robot then leaves the light switch and appears to avoid reentering the light switch. At times, this reentry avoidance can cause the robot to slightly reenter the light switch and thwart its previous efforts, but the intention of the move is clear because the robot generally then begins another semi-elliptical exploration of the environment that oftentimes leads it to the other light switch. There is often a need for wall avoidance during the second-switch-finding phase and many times the robot will run out of time steps either right before or right after turning on the second light switch. The downfall of the evolved robot in this environment seems to be when the robot's initial search for a light switch fails to yield and the robot ends up stalled on a wall. Certain orientations along the wall seem problematic for the robot and it remains stuck until the next iteration.

The robot's evolutionary progress converges quickly with this experimental setup (Figure 5). After about 20 generations, the robot's fitness has plateaued at around 150. There is a degree of fine-tuning that takes place between the 20th and the 100th generation, though. The robot appears much more deliberate in its actions; also, it seems that the fitness may be limited by the amount of time re-

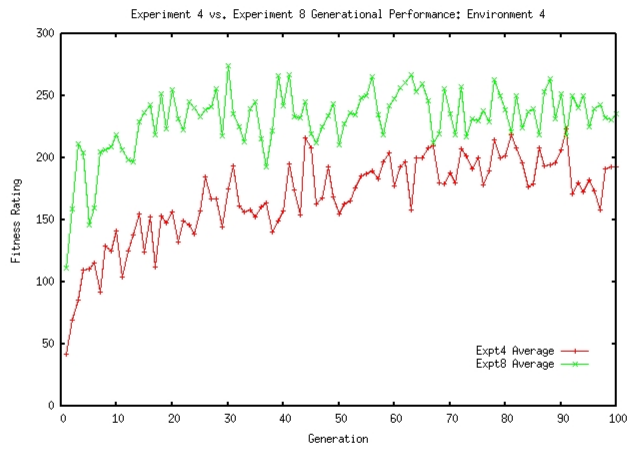

Figure 6: Average fitness versus generation number for robots in environment 4 which were evolved from scratch (red) and were evolved first in environment 2 (green). Fitness curves represent the average fitness at each generation over five trials.

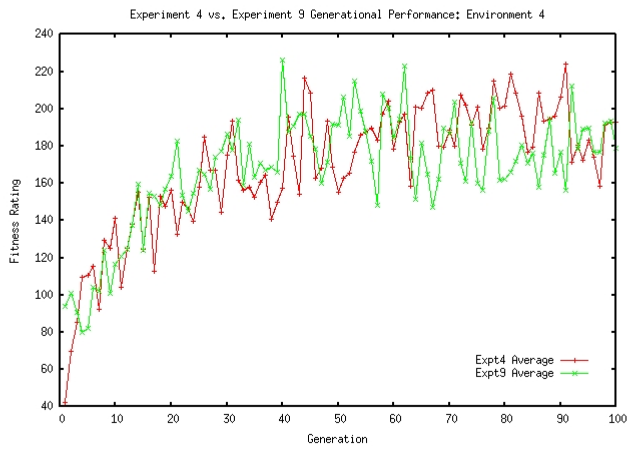

Figure 7: Average fitness versus generation number for robots in environment 4 which were evolved from scratch (red) and were evolved first in environment 3 (green). Fitness curves represent the average fitness at each generation over five trials.

quired to traverse the environment and flip the second light switch.

3.4.2 Pre-evolved in Environment 1

Pre-evolving the robot in environment 1 equips the robot with a seemingly more robust mechanism for avoiding walls relative to the robot that was evolved from scratch. There is also a tendency for the robot to avoid the outer edges of the environment that seems to be a remnant from its pre-evolution. The robot's behavior is also much less exploratory after it has switched on the first light switch, which seems connected to its pre-evolution, where we sometimes saw the robot find the switch and then wiggle back and forth on top of it without further exploring the environment. That said, the robot's rise in evolutionary fitness slightly outpaces that of the robot evolved from scratch (Figure 5), likely because of its proficiency in finding at least one switch every time and rarely stalling.

3.4.3 Pre-evolved in Environment 2

Because environment 4 is very closely related to environment 2, we would expect that pre-training the robot in environment 2 would yield favorable results. The robot's behavior is similar to its behavior after evolving the robot from scratch, except that the robot's stall avoidance appears to be considerably more robust. Presumably, this comes from the necessary wall avoidance mechanism learned to solve the task in environment 2. The robot seems to have adapted well to the new environment, using its previously learned light switch-finding abilities to toggle the first light and and then leaving the switch to seek out the next. On appearance, this would seem to be the most deliberate and task-oriented out of the robots evolved in environment 4.

The results shown in Figure 6 follow closely one's intuition. There is an initial gap in fitness relative to the robot evolving from scratch because the robot is essentially solving the task it was pre-evolved for, allowing it to reliably find at least one switch. This gap is narrowed, though, as the robot evolved from scratch learns to turn on one light with regularity and then the second. What seems to maintain the consistent discrepancy between the two is the clearly more robust wall avoidance behavior. It would seem, given the fitness increase over evolutionary time with this task, that first evolving the robot on a subtask of the harder task at hand yields positive performance results relative to evolving the robot from scratch.

3.4.4 Pre-evolved in Environment 3

The results shown in Figure 7 suggest that there is no relative advantage to evolving the robot in environment 3 before transferring it to environment 4. Because of the relatively simple circular-path strategy evolved in environment 3, the robot seems to have integrated nothing specific into its toolset that could potentially improve its fitness relative to evolving the robot from scratch. After evolving the robot in environment 3, the robot seems to stall quite often and does not explore the environment as expansively as previously seen with other configurations, which is likely a remnant from the light switch being centered in the environment in environment 3.

4 Discussion

Our results indicate that allowing robots to evolve in a simpler environment on a simple task before placing them in a more complex environment can sometimes allow them to evolve more sophisticated solutions to the complex task and can help them to evolve those solutions more quickly. While we were hoping to see a clear progression in terms of the sophistication of the robots' behaviors from environment to environment, where pre-evolution in environment 1 would lead to better results in environment 2, pre-evolution in environment 2 would lead to better results in environment 3, and so on, this was not exactly the case. Instead, we saw that in some cases adding a pre-evolution step made no real difference in the robots' performance, and in other cases, pre-evolving in one environment was more helpful than pre-evolving in another in unexpected ways. For example, we expected that pre-evolving in environment 3 and then placing the robot into environment 4 would result in much better performance in environment 4 since there were two switches in each of these environments, but this was not the case (Figure 5). However, pre-evolution in environment 2 seems to have helped the robots evolve effective strategies more quickly in environment 4 (Figure 6).

The clearest evidence that an incremental approach is useful for evolving complex behaviors can be seen in Figures 3 and 6. Figure 3 depicts the fitness boost obtained when agents are pre-evolved in environment 1 before being placed in environment 2. When agents are evolved from scratch in environment 2, we see that after 100 generations, they have not yet evolved a consistently effective solution to the one-switch problem. Pre-evolution in an environment where the switch is easier to find allowed the robot to quickly (within 10 generations) evolve an effective strategy for turning on the light in the corner. Figure 6 shows that agents which were pre-evolved in environment 2 before being placed converged on a solution to the two-switch problem in environment 4 more quickly than agents which were evolved in environment 4 from scratch.

It is likely that the pre-evolution steps were helpful in these cases because they allowed the robot to evolve some general behaviors that proved beneficial in the more complex environment. For example, in environment 1, agents evolved an ability to avoid getting stuck against walls as well as a drive to see the light turned on, which would be useful starting points for solving the one-switch task in environment 2. When evolving from scratch in environment 2, robots evolved behaviors which led them to spend a lot of time looking at walls. This may have been a helpful behavior in environment 4, since both switches are in corners and robots with a predisposition to look at walls will be more likely to discover these corner switches.

While we are pleased with these positive results, we must also address the question of why our pre-evolution steps were not always helpful. One major reason may be that when designing our environments and evolutionary tasks, we did not correctly predict which environments would be most difficult for the robot to conquer. For example, we were expecting environment 3 to be more challenging than environment 2 because it contained more switches. However, we saw that the robot was able to evolve a reasonably effective strategy for solving the two-switch task in environment 3, even without the benefit of a pre-evolution step. Thus pre-evolution in the simpler environment 1 did not provide any real improvements in terms of the robots fitness level (see Figure 4), but that may be because the task was simply not hard enough to truly benefit from the extra time spent in pre-evolution.

Further, we were not always able to correctly predict which pre-evolution steps would be helpful to the robot. For example, we expected that pre-evolution in environment 3 would be beneficial to later evolution in environment 4, since both of these environments contain two switches but the switches in environment 3 are located more toward the center of the environment and are thus easier to find. Figure 7 shows that this particular pre-evolution step was not useful. Instead, it turned out the pre-evolution in environment 2 was more beneficial 6. Even though environment 2 has only one switch, it is placed in a corner, just like the switches in environment 4. It seems that pre-evolution which helped the robot explore the correct regions of the environment (i.e. in the corner of the room) was more helpful than pre-evolution which helped the robot learn to expect the correct number of switches, which was not a result we anticipated.

5 Conclusion

Our results seem to indicate that it is possible for an incremental approach to be beneficial when trying to evolve complex behaviors, with some caveats. First, it may not always be clear to the human engineer when a task is complicated enough to warrant the use of more time-consuming pre-evolution steps. Second, it may not always be clear to the human engineer exactly what intermediate evolutionary steps will help the robot to ultimately evolve some sophisticated behavior. What may seem to the human like a reasonable decomposition of a complex task into simpler and simpler behaviors may not work well for the robot and its unique perspective on the world.

We have seen that when we do find useful intermediate-difficulty tasks on which to evolve our robots, we can in fact improve the rate and quality of their evolution on more difficult tasks (Figures 3 and 6), suggesting that an incremental approach might prove useful when attempting to evolve very complex behaviors.

Still, this approach may not be the most efficient or the most generalizable technique. For every complex task we wish to solve, we must hand-designed a new set intermediate tasks. This can be time-consuming and may require many trials before we hit on a set of intermediate tasks which actually prove useful to the robots when they ultimate try to evolve complex behaviors. If we would like a more generalizable technique that can be used without much modification to evolve a wide variety of complex behaviors, a system more akin to NEAT (Stanley and Miikkulainen, 2004) might be a better choice.

6 Acknowledgements

Many thanks to Professor Lisa Meeden for advising this project. We also thank the folks of PyRo for providing a starting-point for our genetic algorithm.

References

I. Harvey, P. Husbands, and D. Cliff, 1994. ”Seeing the Light: Artificial Evolution, Real Vision” *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats* pp 392-401.

D. Marocco and S. Nolfi, 2006. ”Emergence of communication in teams of embodied and situated agents.” *Proceedings of the 6th International Conference on the Evolution of Language* pp 198-205.

K.O. Stanley and R. Miikkulainen, 2004. ”Competitive Evolution through Evolutionary Complexification.” *Journal of Artificial Intelligence Research, 21* pp 63-100.