Nature versus Nurture in Evolutionary Computation: Balancing the Roles of the Training Environment and the Fitness Function in Producing Behavior

Jordan Wales, Jesse Wells and **Lisa Meeden**
jwales1@swarthmore.edu, jwells1@swarthmore.edu, meeden@cs.swarthmore.edu
Swarthmore College, Swarthmore, PA 19081

1 Introduction

There are at least three levels at which one can, through a fitness function, influence the development of evolved robot behavior. We have termed these levels: sensory state, behavioral expression, and task achievement. A *sensory state* fitness function attempts to maintain sensor values in set ranges that we believe to be correlated with particular desired patterns of behavior. In contrast, a *behavioral expression* fitness function focuses on rewarding or punishing higher-level behaviors. Finally, rather than monitoring *how* the goal is met, a *task achievement* fitness function is simply based on a measure of the final outcome.

If one considers the sensory level to be the lowest level, behavioral expression the middle, and task achievement the highest, then the necessary complexity of the fitness function increases as one moves to lower levels of control. And, as the complexity of the fitness function increases, the training environment must be simplified.

At the lowest, sensory state, level it is often the case that training cannot initially occur in the target environment. Instead, the robot must be "bootstrapped" through increasingly complex environments until the desired behavior is achieved in the target environment. The resulting behavior is also typically not robust; when transferred to other environments, performance is severely degraded. In short, the environment must be tailored so that the fitness function can exert its influence more directly on the way that the evolving controller translates sensor readings into actions.

At the highest, task achievement, level the fitness function is so abstract that the evolutionary process relies much more heavily on the environment for its cues. Therefore, a single training environment can be used, but it must be very complex to provide a wide range of potential situations. However, the training time can be quite lengthy.

At the intermediate, behavioral expression, level the guidance provided by the fitness function is somewhat abstract, but not overly so. In this case, both the environment and the fitness function can more equally share the load in guiding the evolving behavior. We contend that fitness functions at this level, when combined with reasonably complex training environments, are the most successful at producing robust behavior in fewer generations of evolutionary computation.

We explored these ideas by evolving an artificial neural network (ANN) controller for a simulated Khepera robot navigating through various labyrinths. The Khepera [4, 3] is circular in shape and miniature (diameter 55 mm, height 30 mm, and weight 70g). It has two motors which power two wheels. Its standard sensory apparatus consists of eight infra-red proximity sensors.

2 Training environments

The environment was conceived of as a simple labyrinth—a twisting and turning passage that constitutes a single path. We selected a labyrinth rather than a maze because the robot is constrained to a single path and can therefore be guided through a series of encounters with vari-

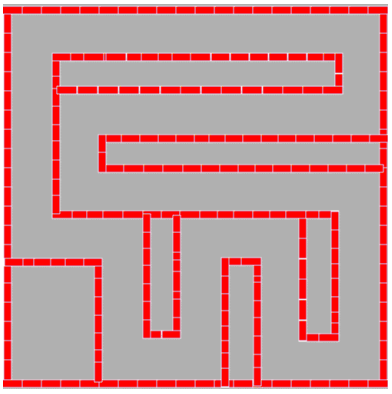

Figure 1: The rectilinear environment.

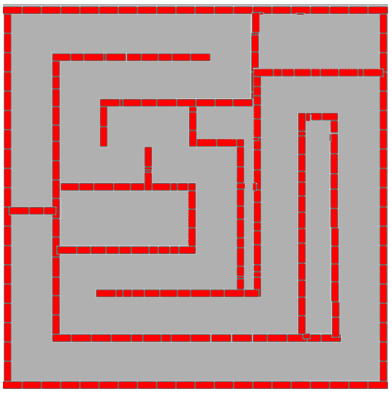

Figure 2: The dead end environment.

ous types of obstructions and path configurations. This constrained environment provides a good forum for the development of a controller able to successfully maneuver in tight spaces.

Three different environments were used. Each was developed to abide by the labyrinth paradigm while presenting a different set of principle features. The rectilinear environment, shown in Figure 1, contains 90-degree turns and two dimensional corners only. The dead end environment, shown in Figure 2, contains 90 degree turns, one and two dimensional corners and a dead end. The angle environment, shown in Figure 3, contains turns of multiple angles, and one and two dimensional corners.

The differences in the environments can be illustrated by rating their difficulty. The rectilinear environment is clearly simpler than the other two. All passages are of equal width, all turns are 90

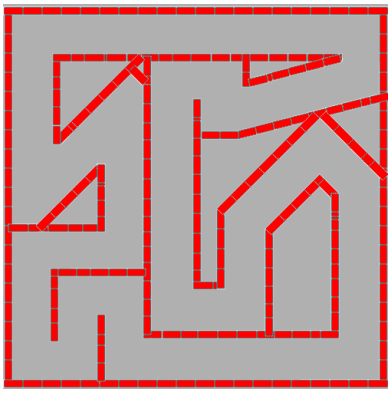

Figure 3: The angle environment.

degrees, and each turn is bounded by a straight passage. The dead end environment is more difficult than this because a 180 degree turn is required at the dead end, and the additional one dimensional turns are more difficult to sense. The angle environment is the most difficult because, in addition to one dimensional turns, it also has passages of uneven width and turns of multiple angles.

3 Fitness Functions

In designing possible fitness functions for this task, we began with Floreano and Mondada's idea of creating fitness criteria based on variables that can be directly measured on the robot at each time step [1]. For example, they designed a fitness function for navigation and obstacle avoidance that maximized the product of three components: speed, straightness of motion, and distance from obstacles.

3.1 Sensory State

The design of this fitness function was predicated upon the assumption that we wanted the robot to develop specific behaviors, and that these behaviors would lead to successful navigation of the labyrinth. The desired behavior was straight and fast forward motion between the parallel walls.

We took an explicit approach to ensuring the last of these conditions, movement between parallel walls, would be met through what we termed the

Voronoi criterion. Based on the assumption that the ANN would use sensory data to determine its next move, we hypothesized that the sensory state of the robot would be a good indicator of how well it was maintaining its path between the walls on either side of the corridor. Therefore, obstacle-avoiding behavior would be characterized by infra-red readings that were equal on both sides of the robot.

In experimenting with various sensory-based fitness functions, we discovered that the evolutionary process found numerous loopholes through which the robot was able to obtain a high score by selecting an undesired, simpler behavior. These loopholes required further modifications of the fitness function, eventually leading to a rather ad hoc and unweildy fitness measure. This fitness function maximized the product of maintaining centeredness, speed, and straightness. In addition, the result was decremented when a collision occurred.

3.2 Behavioral Expression

Rather than attempt to reward the robot for sensor states that we believed were indicative of good behavior, we elected to score the robot directly on behavior. This philosophy led us to remove the Voronoi criterion and to reward the robot, as before, for straight and fast motion, with a decrement for collisions. This removed our bias that being centered in the passageways was an important aspect to solving the task.

3.3 Task Achievement

This final fitness function was by far the simplest. We evaluated performance based solely on the percentage of the environment covered by the robot. To calculate this, the environment was divided into a grid of robot-sized cells and each cell was initially marked as unvisited. Then as the robot moved, its current position was used to update the appropriate grid cell as visited. In this case, there is no explicit reward for moving fast, straight or avoiding obstacles. Yet, each of these features of movement are implicitly required in order to cover the most ground.

This led to some very interesting behaviors that were side effects of the discretized area measure we employed. Those results are explored in our discussion of experimental results.

4 Experiments

Our aim was to investigate the effects and interactions of two variables—namely, training environment and fitness functions. We used the three fitness functions and the three environments described above. Each possible combination was evolved four times.

The ANN controller had a fixed, Elman-style architecture consisting of 8 inputs for the infra-red sensors, 3 hidden nodes, and 4 outputs to control wheel speeds. The goal of the evolutionary process was to find a successful set of weights. Each individual in the population was represented as a real-valued string of weights. Initially, the weights were set randomly, and each individual was evaluated for 300 simulated seconds. This is enough time to complete two loops through each labyrinth. Selection was done by tournament, with the loser being replaced by a mutated copy of the winner. No crossover was used. For details on the merits of this style of evolutionary computation see [2]. Each evolution consisted of 300 generations, with a population of size 30. The ANN controller that produced the best fitness score during the evolutionary process was returned as the result.

Each best ANN from each of four evolutions was tested in each environment 4 times. This final evaluation was whether the robot traversed the entire path of the labyrinth or not. This was considered the ultimate measure of success for a labyrinth navigating robot.

5 Results

Path completions were infrequent for ANNs trained under the state-based fitness function, see Figure 4. As one would expect, path completion occurred most often in the Rectilinear environment—the environment most closely fulfilling the assumptions (straight, parallel pas-

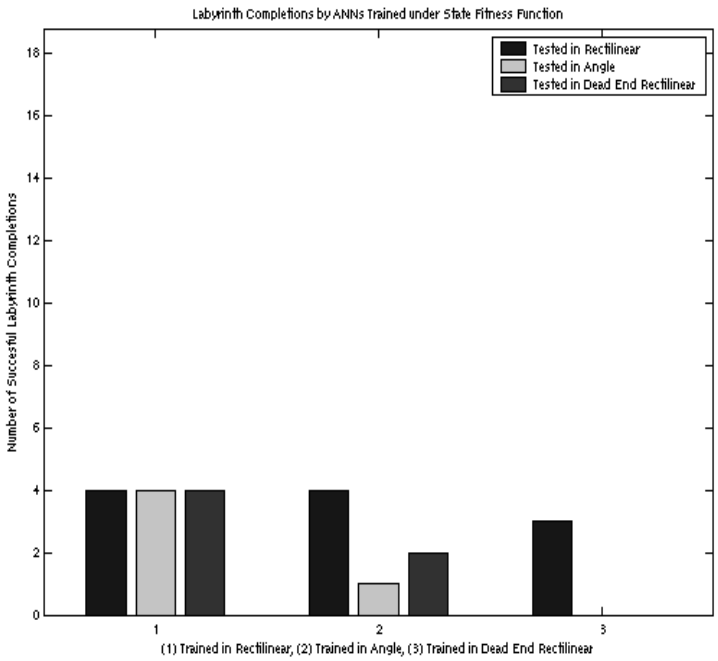

Figure 4: Labyrinth Completions by ANNs Trained under State Fitness Function

sages) upon which the Voronoi criterion was based.

It is interesting that ANNs trained in the Rectilinear environment under this fitness function performed equally well when tested in all three environments. This result accords with the assertion that a simple training environment allows the fitness function greater influence over ANN development. These results could be taken as indication that the theory of implicit progression (state implies behavior, which implies success) was indeed correct. However, these results should be treated with caution; the successful test runs by ANNs trained in the Rectilinear environment were all produced by one of the four ANNs. The other three trained in that environment with the state-based fitness function did not complete any runs successfully.

ANNs trained under the state-based function in the Angle and Dead End environments had only a few successes. The ANN trained in Angle had the most successes when tested in Rectilinear. Because of its configuration, Rectilinear is most conducive to simple movement fulfilling the Voronoi criterion. With so few successes, it is difficult to draw reliable conclusions, however one should note that the ANN was more successful in Dead End even when trained in the Angle environment. This could possibly indicate, again, the dominance of the fitness function rather than the environment in formation of behavior.

Figure 5 shows results for ANNs evolved under the Behavioral Expression fitness function. This function may at first seem quite limited in its usefulness, obtaining a majority of successes only when trained in the Angle environment. It was predicted earlier that the behavior-based fitness function would produce ANNs with generalized behavior capable of dealing with novel features if those ANNs were given a rich enough training environment. Indeed, the ANNs trained in the Angle environment gave a strong performance on test in all environments. Behavior was generalized sufficiently to accommodate to novel features such as the dead end. Evidently, the Angle environment provided the rich environment that was needed.

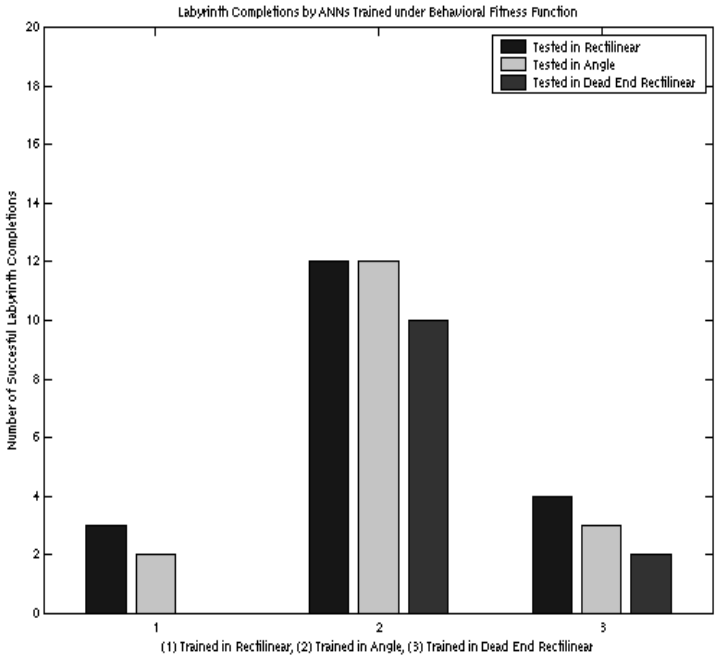

Figure 5: Labyrinth Completions by ANNs Trained under Behavioral Fitness Function

The ANNs trained in the Dead End environment did not have the benefit of the varying angles in their training experience. In test, it was observed that ANNs trained under the behavior-based and state-based fitness functions never became stuck on corners or projections. Their most common failure was to execute a 180-degree turn at a bend in the hall, leading to a doubling-back that prevented a complete path traversal in the allotted test time. Among ANNs trained in the rectilinear environments, the 180-degree turn usually came either at the first or second bend in the corridor. The robot would make another 180-degree turn at the next encountered forward obstacle, leading to a back-and-forth behavior confined to a single straight portion of the labyrinth.

ANNs trained in the Angle environment would fail by doubling back after several twists and turns through the labyrinth, retracing the path and losing a lot of time. This type of doubling back and path retracing was less frequently observed among ANNs trained in Rectilinear environments.

We hypothesize that the reason for this lies in the geometry of the turns required by the purely rectilinear environments: they require only 180-degree and and 90-degree turns, permitting the robot to turn blindly 90 or 180 degrees toward the closest obstacle-free heading as soon as it is confronted with a forward obstacle. The Angle environment, presenting bends with a variety of severities, forces the robot to monitor its sensors while executing a turn. Thus, the Angle-trained ANN can deal with the dead end, even though such a turn was previously not required of it, because the requirements of the varying angles produce behavior that generalizes well to a dead-end turn-around.

Following this logic, it is to be expected that the ANNs trained in purely rectilinear environments will be ill-equipped for the angle environment since there is nothing in those environments to encourage a more complex turning ability.

The Task Achievement fitness function ensures that the dominant ANNs are those that can cover the most ground in the environment. Peculiarities of our method for determining area covered led to

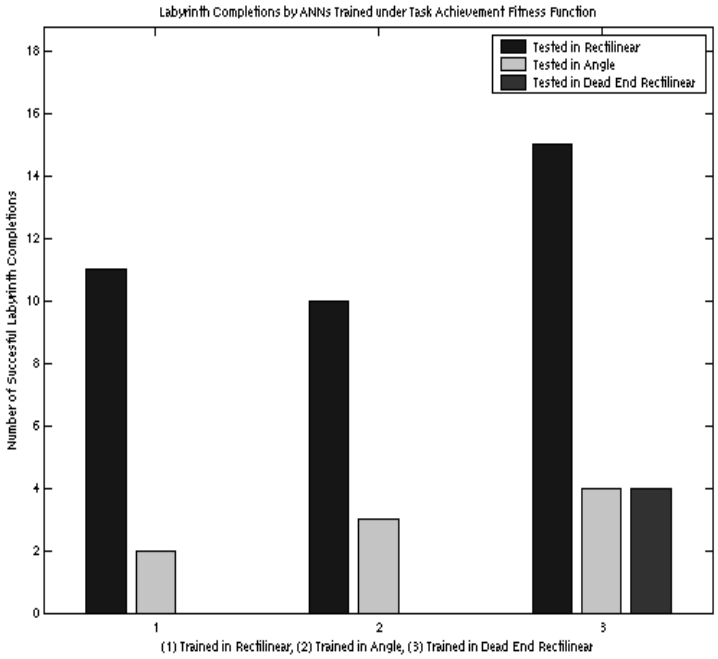

Figure 6: Labyrinth Completions by ANNs Trained under Task Achievement Fitness Function

some strange behaviors. The environment was divided into grid squares measuring 30 units on a side. The robot itself is circular with a diameter of 60 units. It was observed that the robot would often move sinusoidally through corridors, passing through as many grid squares as possible. Some robots clung to the walls of the corridor, resulting in slow movement that prevented successful path traversal.

Figure 6 shows that these ANNs performed consistently well in the basic Rectilinear environment but in other environments performance was not appreciably better than that of the state-based ANN. Getting stuck on corners and in angles were the most common failure of the Task Achievement trained ANNs. This makes sense if one considers that conservative turning helps prevent the robot from doubling back, permitting it to maximize ground covered.

By the same token, this reluctance to turn results in a luck-of-the-draw turning behavior. At an angle, there may be no clearly free heading in the forward sensor readings; this is surely the case with the dead end. In situations where the necessary turn is not crisply defined in the forward sensor readings, the ANN, having learned to avoid doubling back, often does not complete the required turn in time. Also, having evolved to move along walls rather than at a distance from them, the ANN may not turn the robot out of corners into which it has moved. All these factors lead to the results observed—consistent success occurs only in the simple, well-defined rectilinear environment.

Without a general behavioral goal to guide it, the ANN develops to perform as well as possible when confronted with features from the training environment. Further investigations hint that, under the Task Achievement fitness function, a much longer training period with a larger population and a more complex environment produces ANNs which are more robust. However, the ANN trained under this function does not produce generalized behavior patterns. Therefore, novel features are never dealt with particularly well. When considering the number of 90-degree turns necessary in all three environments, it is easy to

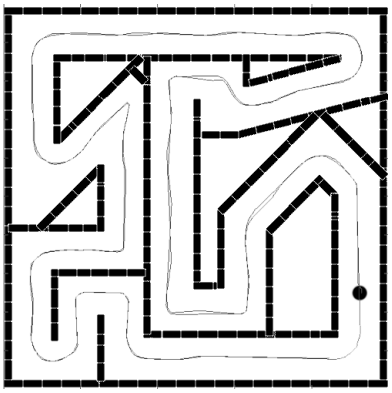

Figure 7: Path taken by an ANN trained in the Angle environment under the behavioral expression fitness function.

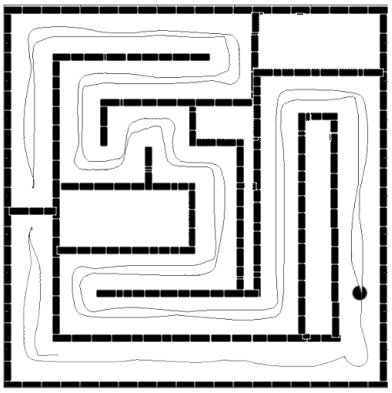

Figure 8: Path taken by an ANN trained in the Dead End environment under the task achievement fitness function.

see that the comparative infrequency of angles or dead ends in the Angle and Dead End environments made these much less salient elements during training.

See Figure 7 for the a sample traversal of a ANN trained under the behavioral expression regime. See Figure 8 for the a sample traversal of a ANN trained under the task achievement regime. Notice that it has a curvier path than seen in Figure 7 because straightness was not explicitly measured and covering wider ground was rewarded.

6 Conclusions

In summary, we found that the most successful fitness function and training environment combination was one in which a good balance was struck between those two factors with respect to their influence on behavioral development. The behavior-based fitness function encouraged desired behavior while leaving the evolutionary process relatively free to develop its own strategies for attaining this behavior based on the features encountered in the training environment. This resulted in behaviors that were easily generalizable.

At opposite extremes, the state-based function paid little heed to the environment itself and the task achievement function allowed the evolutionary process to adapt to the specific requirements of the training environment. A primarily rectilinear environment results in rectilinear behavior. Repeated training in very complex environments is necessary under task-achievement fitness measures, and repeated training in environments moving from simple to complex is necessary under the state-based fitness measure.

References

[1] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions of Systems, Man, and Cybernetics-Part B: Cybernetics*, 26(3):396–407, 1996.

[2] L. Meeden. An incremental approach to developing intelligent neural network controllers for robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 26(3):474–485, 1996.

[3] O. Michel. Khepera simulator package version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia–Antipolis. World Wide Web, URL: http: // wwwi3s.unice.fr / ~om / khep-sim.html, 1997.

[4] R. Mondada, E. Franzi, and P. Ienne. Mobile robot miniturization: A tool for investigation in control algorithms. In *Proceedings of the Thrid International Symposium on Experimental Robots*, Kytoto, Japan, 1993.