Embodying and Improving SODA

Allison Barlow and Bryce Wiedenbeck

Abstract

We implemented variations on the Self-Organizing Distinctive State Abstraction model for robot state-discretization and task-learning in a Khepera robot. We conclude that hill-climbing as implementd in [5] is detrimental in embodied robots. Because the ultimate goal of developmental systems is intelligent, embodied robots, we believe that implementation in physical robots is an important test of robustness for proposed developmental mechanisms.

1 Introduction

In the fields of Adaptive and Developmental Robotics, a surprising number of systems produced through research do not involve physical robots. Instead, researchers focus on developing intelligence systems in virtual worlds, which are often simpler to use, more portable, and cheaper. Pfeifer and Scheier raise concerns that the traditional AI systems in symbolic worlds often lack robustness and generalization[4]. Although simulated worlds have improved to more closely represent the noisy real world, they are still predictable and lacking in complexity. Even in a relatively controlled physical environment, a robot's motors often fail to produce the intended action and a robot's sensors often have to cope with statistically biased noise, neither of which is adequately addressed by existing simulations. If our ultimate goal is to produce robots and robot control mechanisms that can function independently in the real world, we need to test the robustness of our systems in physical robots as well as in simulation.

2 Background

In [5], Provost, et al. propose an interesting model by which a robot can learn to extract distinct states from its continuous sensor inputs and subsequently use those states to learn by reinforcement to accomplish a simple task. They test their system with reasonable success in simulation, so we sought to verify these results by replicating their methods in an actual robot. In our midterm project, we focused on varying the learning of state abstractions, as opposed to task learning, but our results yield some insights about the feasibility of attacking a such higher-level learning tasks. In our final project, we continued in this same line of work and expanded our experiments to task learning.

2.1 SODA

Provost, et al. present a model called Self-Organizing Distinctive State Abstraction (SODA), and demonstrate its feasibility through an experiment with a simulated robot. SODA combines several distinct components into one system that seeks to abstract and use distinct states from continuous input. The first component, Growing Neural Gas (GNG), creates a graph whose nodes approximate the distribution of input gathered by the robot as it wanders randomly in its environment. After this wandering stage is complete, the robot is placed in the same environment, and determines which of the states (graph nodes) from the GNG its sensor values most resemble. It then enters a hill-climbing phase in which it seeks to put itself as exactly in that distinct state as possible by maximizing the similarity between its sensor values and that state. After completing the hill-climbing phase, the robot chooses a primitive motor action (move forward, move backward, turn right or turn left), which it executes until its sensors more closely resemble a different state (this is called trajectory-following). SODA then uses reinforcement learning to develop high-level actions to solve a simple problem. It uses Q-learning to

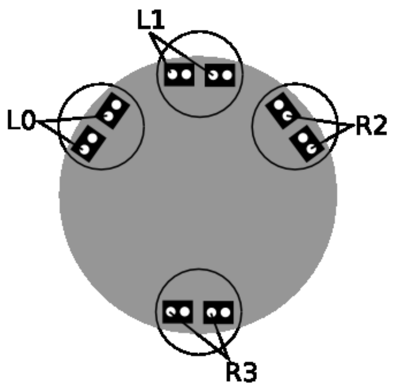

Figure 1: The Khepera's 8 range and 8 light sensors (one of each per block) were grouped into 4 of each (L0-L3 and R0-R3).

determine which primitive action to select for trajectory following from any given state. In Provost, et al. SODA learns to turn right in a T-maze.

2.2 CS81 Midterm Project

Our midterm experiment [1] sought to test the feasibility of the SODA model with a non-simulated robot, and therefore implemented the Growing Neural Gas and hill-climbing/trajectory-following aspects with a Khepera II robot. There was a clear tradeoff between simplicity of representation (number of states) and accuracy of representation (our interpretation of the comprehensiveness of the states), which we could control through an error threshold parameter.

By testing the SODA model in a Khepera robot, we hoped to provide closer scrutiny and confirm or reject the model's applicability as a component of a developmental system. The GNG succeeded in abstracting a reasonable set of discrete states from the robot's sensors, and we hope that these states could aid such a robot with learned or programmed tasks. However, our experiment did locate a number of shortcomings of the SODA system, many of which are addressed by our final project.

3 Experiment

We designed experiments to test several aspects of the SODA model. First, we wanted to test SODA in a physical robot. In [5], Provost, et al. used a simulated robot which we thought may have greatly simplified the task. We therefore implemented the SODA model using a Khepera II robot, interfacing through pyrobot (`www.pyrorobotics.org`). The khepera has a total of 16 sensors: 8 light sensors and 8 range sensors, each of which returns a real value between 0 and 1. To reduce the dimensionality of the input to the GNG, we collapsed these sensors (as shown in figure 1) to 4 of each by summing adjacent sensors, resulting in real input values in the range 0 to 2.

Because we wanted the robot to solve a learning task that required use of both light and range sensors, we replaced the T-maze environment with the plus-maze depicted in figure 2. In each reinforcement learning step the robot started randomly from either the top or bottom of the plus maze and, in order to maximize reward, had to find its way to the light in the right-hand arm of the maze. In simulation, Provost et al. were able to instantly and precisely restart the robot from its initial location at the start of each trial. To achieve close to the same precision would have required us to return the robot to its initial position by hand at the start of each of our thousands of trials. Daunted by this potential problem, we opted instead to write a control function that would automatically return the robot to one of the two

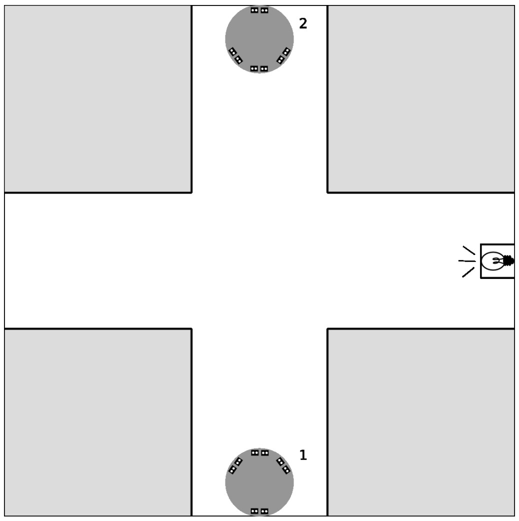

Figure 2: The plus-maze environment. At the start of each reinforcement learning step, the Khepera started at either position 1 or position 2.

starting positions. This resulted in greater variation in the robot's exact starting position and orientation. We also considered conducting some of our initial experiments in simulation before moving them to the physical robot, but found that available simulators didn't provide an accurate enough representation of real-world conditions.

Our second goal was to test whether growing neural gas could appropriately discretize input from more than one type of sensor. The concern was that when the GNG computes the error values that govern the growth of the network, it simply calculates Euclidian distance, which may not be optimal when different input dimensions represent fundamentally different types of values (like light sensors and range sensors). Even though each type of sensor was scaled to the same range (which might not even be feasible in larger systems), we suspected that the distribution of values might be sufficiently different to cause a bias in the GNG toward one sensor type or the other. We therefore created two separate sets of states: one where all 8 sensors were input to a single GNG network, and another where the range sensors were discretized by one GNG while the light sensors were discretized by another. In this second case, the set of states for reinforcement learning was the set of all pairs of one range state and one light state.

Our fear that error distribution would differ significantly between sensor types was borne out by our first attempt at creating GNGs. We used the same error threshold for adding new nodes in both the range and light GNGs, and the first light GNG we created contained only three states that seemed not at all representative of the robot's environment. In our reinforcement learning trials, the error-threshold parameter was tuned separately for all three GNG types.

Our third experiment, motivated by our previous work in [1], was meant to test the hill-climbing portion of SODA's reinforcement learning. Our objections to hill-climbing were manifold: first, it would be difficult to extend to a robot with more complicated motor actuators. Hill-climbing is tractable for Provost et al. and for us only because we have a maximum of four primitive actions; it would be completely unwieldy in a robot with tens or hundreds of actions to test. Second, we observed in our previous experiment that hill-climbing makes a fundamentally flawed assumption: that motor actions are reversible. In a real robot (especially one that doesn't yet have a strong grasp of its environment), this is often not the case, and making that assumption often causes hill-climbing to lead the robot wandering in completely arbitrary directions. Third, hill-climbing cannot be motivated in terms of cognitive science. When people see a situation that is similar to something they recognize, they don't try to force that situation to be exactly like the one they have information about, but rather extrapolate and attempt an action similar to the one that has worked before. Finally, hill-climbing just plain takes too long: in practice, we saw that the Khepera spent a large portion of its time hill-climbing, during which it hardly advanced, whereas the time spent trajectory-following helped to move it toward its goal.

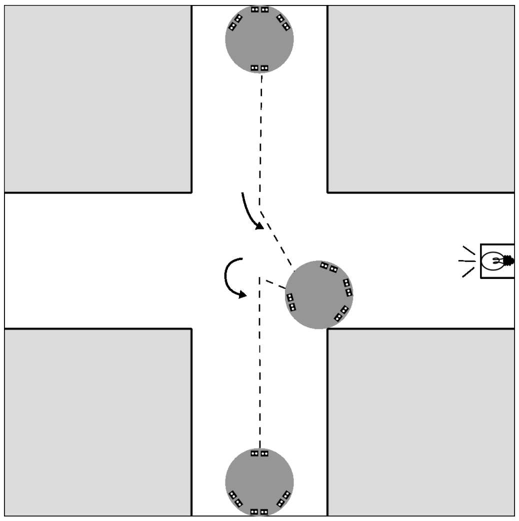

Figure 3: When learning with our original reward function (the sum of all light sensors), the Khepera maximized its reward not by moving to the light, but rather by getting stuck on a corner where it could see the light in three different sensors.

Therefore, we designed our experiment to test whether hill-climbing provided any benefit in reinforcement learning. We conducted three trials: first we compared reinforcement learning with and without hill-climbing using the states from the full GNG, and then compared these to reinforcement learning without hill-climbing on the combined states from the seperate GNGs. Given more time, we would also have tested the two-GNGs with hill-climbing. Each iteration lasted 60 seconds for the trials without hill-climbing and 90 seconds for the one with hill-climbing. After each iteration, the robot returned randomly to one of the two starting positions. The robot's reward for each iteration was equal to the amount of light it saw in its front light sensor at the end of that iteration (this means that the robot gets more intermediate reward than in the experiments of Provost et al.). We saved intermediate results of each trial so that we could evaluate both the final performance achieved and the rate at which that performance was achieved.

We also made a few minor changes from Provost et al. that were consistent across all of our trials. First, when adding nodes to the GNG, rather than checking whether the average error across the network was above some threshold, as in SODA's equilibrium GNG, we simply checked whether the error at the worst GNG-node was above a threshold. This simplified the implementation slightly, but also allowed the network to improve in regions that were locally problematic even if error across the whole network were relatively low. We also replaced SODA's Q-learning with a related solution to the multi-armed bandit problem derived from [3]. This provided two advantages: first, faster propagation of learned values, which helps speed up the time-consuming learning process; and second, better exploration of options from any given state. While Q-learning treats all options other than the best equally, our solution keeps track of additional information on how frequently an action has been selected in a given state, which allows it to explore the second-best option the second-most, the third-best option the third-most, and so on.

4 Results and Discussion

Due to time constraints, we were not able to run all of our trials to completion. We were, however able to run all three trials long enough to draw general conclusions about each system's rate of learning.

Our first attempt at learning revealed a flaw in our fitness function. Our original fitness function sought to maximize the sum of all of the light sensors. The robot learned that if it got too close to the light, only a few of its sensors would see it, decreasing the reward received. Instead, the robot learned to get stuck on a corner, as seen in Figure 3 far enough away from the light to maximize light seen in multiple sensors. Our improved fitness function sought to maximize the maximum valued light sensor

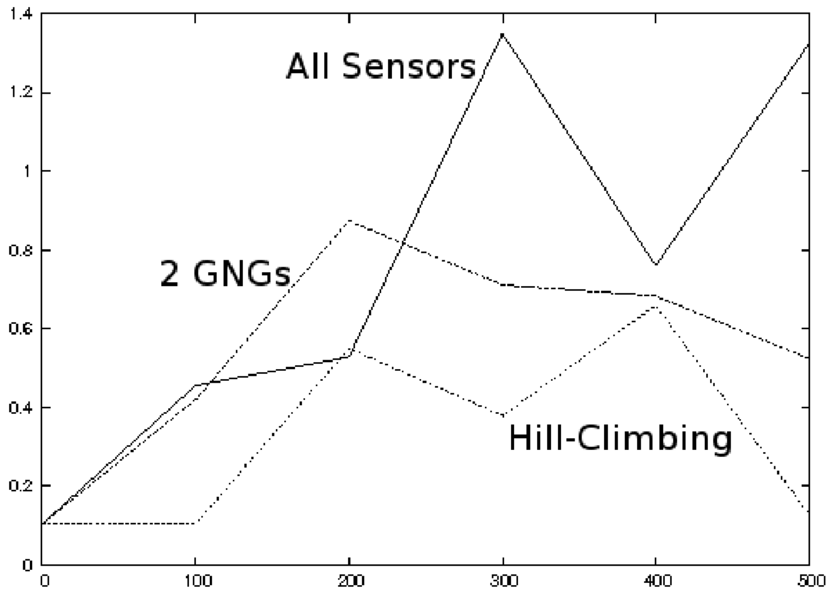

Figure 4: Graph showing the rate at which all three systems learned over the first 500 iterations. Note: maximum possible reward is 2.

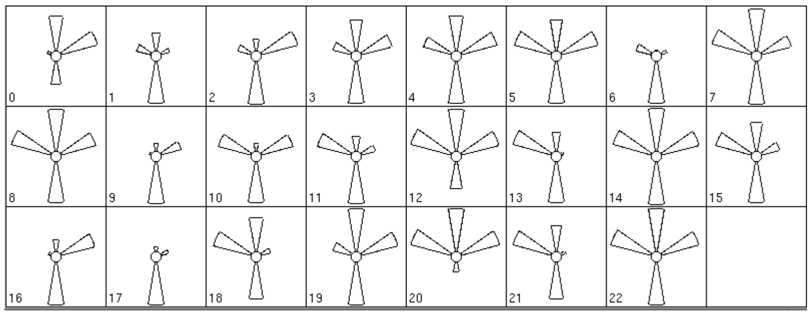

Figure 5: Distinct states from the range-sensor GNG.

pair.

Figure 4 shows results from all three trials. Each point represents the average fitness achieved across five trials at a particular stage of learning. The clear winner is the single GNG without hill-climbing, it learned much faster and much better than the other two systems (observational evidence indicates that the difference may in practice be even more pronounced). This confirms our hypothesis that hill-climbing is unnecessary, even to the point of being counter-productive. Our other hypothesis, seperate GNGs for each sensor type would improve performance turned out to be wrong. In the two-GNG experiment, both GNGs had a decent number of states, and the result of combining them was an unwiedly number of states, requiring a longer time to learn to solve the task effectively. This system, however, still outperformed hill-climbing, which shows what a detriment hill-climbing can be.

Another difficulty with the 2-GNG system is that we never managed to create a satisfying light-sensor GNG. The one shown in figure 6 is the best we achieved, but still has significant obvious gaps, and to produce it, we had to set the GNG error threshold incredibly low. Figures 7 and 5 depict the states produced by the other two GNGs.

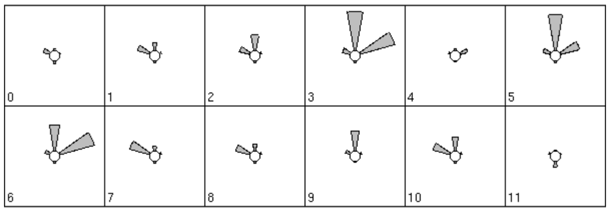

Figure 6: Distinct states from the light-sensor GNG.

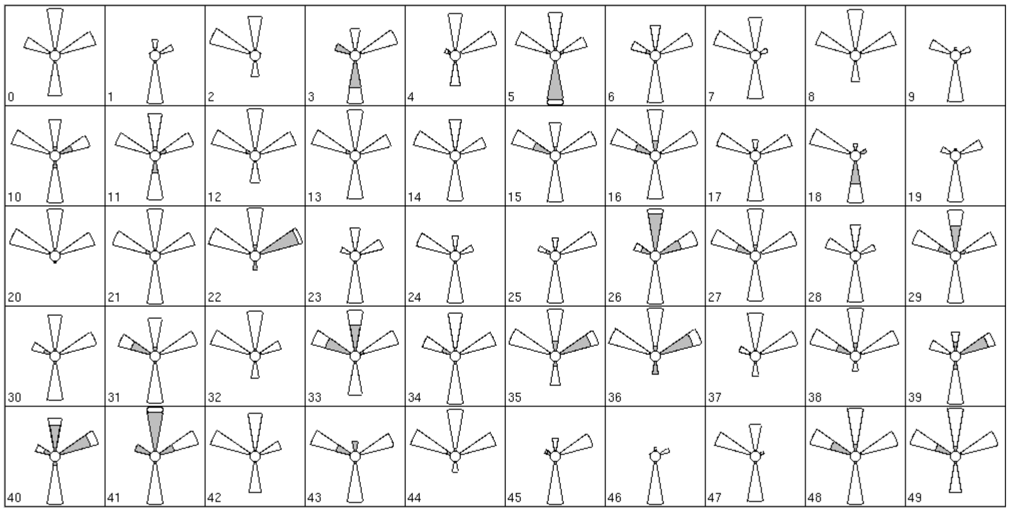

Figure 7: Distinct states from the full-sensor GNG.

5 Future Work

While the first step in continuing this project would clearly involve more extensive and rigorous testing, we also had a number of other ideas that we would like to eventually implement. The first is to improve on SODA's method of wandering random primitive actions to explore its environment. In [1], we used a more intelligent, albeit less adaptive exploration scheme, which produced similar GNG results much more quickly. In [2], Dahl and La Touche suggest the idea of using information about regions of high error in the GNG to indicate areas that the robot needs to investigate more fully.

As an extension to our experiment in which we compared learning with and without hill-climbing, we would like to test other alternatives to hill-climbing. One possibility suggested by [7] involves predicting the result of each primitive action during hill climbing, rather than performing them all. This could be done using any of a number of supervised learning methods. Another possibility would be to consider not just which GNG state is closest, but also how close that state is. It might be possible to interpolate between several neighboring states and base the selection of a primitive action on some weighted average of those states. The learned result of that action could then be propagated to some extent to each of those neighbor states. While this would certainly replace hill-climbing, it might also require significant modification to trajectory-following, because the weighted average between neighbor states could change quite rapidly.

Finally, we would like to compare our modified SODA system to the modified version of SODA presented in [6]. There, Provost, et al. make a number of updates to SODA, including "options" and a significant expansion of the number of states for reinforcement learning, made by considering the three nearest states in the GNG at all times (but not interpolating between them).

6 Conclusion

From this series of experiments, we conclude that hill-climbing is not helpful or perhaps detrimental to learning tasks in embodied robots. Additionally that discretization with a GNG network is robust to cases with multiple types of sensors.

References

[1] A. Barlow and B. Wiedenbeck. Developing growing neural gas categories in a khepera robot. In *CS81 Midterm Projects*, 2008.

[2] G. Dahl and K. La Touche. In *CS81 Midterm Projects*, 2008.

[3] S. Gelly and Y. Wang. Exploration exploitation in go: Uct for monte-carlo go. 2006.

[4] R. Pfeifer and C. Scheier. The fundamental problems of classical ai and cognitive science. In *Understanding Intelligence*, pages 59–78. MIT Press, 1999.

[5] J. Provost, B. J. Kuipers, and R. Miikkulainen. Developing navigation behavior through self-organizing distinctive state abstraction. *Connection Science*, 18(2):159–172, 2006.

[6] J. Provost, B. J. Kuipers, and R. Miikkulainen. Self-organizing distinctive state abstraction using options. *7th International Conference on Epigenetic Robotics*, 2007.

[7] D. Rosen. In *CS81 Midterm Projects*, 2008.