An Analysis of the Factors Influencing the Performance of the SODA Algorithm

Anthony E. Manfredi and Ethan G. Jucovy

May 15, 2006

Abstract

To analyze its robustness, we reimplement the SODA algorithm presented in [2], using a neural network controller and simulated sonar sensors in place of a q-learning table and laser rangefinders. Our experiments do support the claim that the high level perceptual features and action abstraction systems of SODA improve learning in the robot: in our best performing experiment, a robot which selects high-level actions according to the SODA algorithm completes its task 106 times more often and 10 times more quickly than a robot which selects primitive actions only. However, our results are somewhat variable depending on experimental conditions, and they consistently fail to achieve as optimal a solution as reported in [2]: our robot never moves directly to the goal but always wanders considerably. We try four variations of a simple goal-achieving task, modifying experimental parameters in each variation; some experiments result in moderate success, but one experiment fails utterly with the robot never completing its task. We argue that SODA is indeed successful, but its performance is highly dependent on available primitive actions, sensor accuracy, and perceptual features developed.

1 Introduction

Provost, Kuipers, and Miikkulainen[2] developed Self-Organizing Distinctive-state Abstraction (SODA), a generic algorithm by which a robot generates high-level features with which to classify its environment and then builds a set of long-range abstract actions to move between local environmental maxima. A SODA agent can then execute a series of high-level actions to achieve some goal. Provost et al. combined this architecture with a learning algorithm and demonstrated that training a robot to select and execute temporally-extended high-level actions could significantly improve learning time compared to a robot navigating solely with a set of primitive actions. As the present work reimplements and builds on SODA, an overview of the steps of the algorithm are presented in the "Architecture" section, though the reader should refer to [2] for fuller details.

Given the success of the SODA algorithm in the implementation described by Provost et al., we hoped to replicate the original results using a slightly different experimental architecture to evaluate the system's robustness.

Our abstraction system is a modified form of the SODA architecture. Following Provost et al., our robot first learns a set of high-level perceptual features by randomly acting in the environment and constructing a self-organizing map. Provost et al. precede this step with a stage in which the robot learns an abstract motor interface and defines its own set of primitive actions; for simplicity we omit this step and instead provide the robot with a set of primitive actions consisting of various combinations of translational and rotational movement.

Our primary modification is the use of a neural network, instead of the Sarsa(λ) algorithm used in the original experiment,, to train the robot. We attempt to preserve as many of the details of the original experiment as possible in our experiment, although certain changes are unavoidable due to limitations of our simulator implementation: the most significant is our simulator's lack of support for laser rangefinders, so our robot uses sonar sensors rather than a laser rangefinder for distance measurements. A few other minor changes are made in individual experiments, which will be addressed when we discuss our experiments in detail.

2 Architecture

SODA consists of five major steps:

1. Define a set $A0$ of primitive motor actions consisting of $-u^i$ and u^i for each orthogonal motor u^i available to the robot.

2. Learn a set F of high-level perceptual features by exploring the environment for an extended period with a random series of $A0$ actions and training a self-organizing map (SOM), using the sensor signals at each timestep, to converge on a set of recurring high-level features, or *regions*, and associated exemplars.

3. Define a hill-climbing law (HC) for perceptual regions. For each input signal i, determine the perceptual feature $f \in F$ that best approximates i. Then pick the $A0$ action a^0 whose resulting sensory state which best improves the input signal's similarity to the current feature exemplar f, hereafter referred to as the *activation gradient.* This step repeats until all $A0$ activation gradients are negative, at which point the agent has reached a local maximum or *distinctive state* in its present region.

4. Define a trajectory-following law (TF) which carries the agent from one perceptual region to another. Each TF consists of picking a single A0 action and repeating it until the agent is in a new perceptual region; thus there are precisely as many TFs as there are primitive actions.

5. Define a set $A1$ of abstract actions, each of which consists of a single TF and a single HC stage in the resulting new region. Thus, an $A1$ action consists of the agent's moving from an initial perceptual region to a new region, and then exploring the new region until it reaches a distinctive state in that region.

Reinforcement learning methods can then be used to train a SODA agent's $A1$ action selection routine.

3 Implementation

The agent in our experiment is a Pioneer robot, simulated in the Pyrobot Simulator, with a rotational and a translational motor axis.

The robot's $A1$ action selection is controlled by a complementary reinforcement backpropogation neural network (CRBP) with sensory inputs and a large (20-unit) hidden layer. The output of the CRBP is a set of probabilistic bits corresponding to the dimensions of the $A0$ action set; each time a decision is required an action is selected by probabilistically flipping each bit of the output to 1 or 0 and executing the action corresponding to that unique binary sequence, hereafter referred to as a *resolved output*. Care is taken to ensure that the complement of any given binary sequence results in a precisely reversed action so that the complementary reinforcement learning is meaningful.

3.1 Growing Neural Gas

In the present experiment the agent's self organizing map of perceptual features is implemented as a Growing Neural Gas (GNG) network[1] using code developed by Jennifer Barry and Heather Jones. Before each experiment, the robot explores the experimental environment by randomly selecting and executing an action from the A0 action set at each timestep. Sonar readings from this period are used to train the GNG network to adequately represent the environment. New nodes are created when the total network error rises above a maximum error threshold, and training ceases when the total network error drops below a minimum error threshold. Once this training is completed it is not necessary to repeat it again for the same experimental configuration. For a detailed explanation of the GNG algorithm refer to [1].

3.2 Reinforcement Learning

Reinforcement learning is executed on the robot's CRBP. The network is trained upon completion of each $A1$ action, being rewarded if the agent achieves or approaches its goal and punished if it moves away from its goal or stalls.

A single $A1$ action can potentially have a very long temporal extent and therefore training might be highly infrequent; therefore, to facilitate quicker learning, the network's weights are updated multiple times per training sequence. When rewarding the network, the network is repeatedly trained with the resolved output, once per timestep, until its probabilistic output results in an identical resolved output to its original resolved output. Punishment follows a similar algorithm: the network is trained with the complement of the resolved output, once per timestep, until its probabilistic output results in any resolved output nonidentical to the original resolved output.

4 Experiments

To explore various parameters of the experiment and determine what might help or hinder the robot's learning, we modify the conditions of the experiment and run four distinct experiments.

In all of our experiments, the task and experimental environment remain constant. The robot's goal is to reach a fixed area of the world located at the end of a corridor. The world, depicted in Fig. 1, is a t-junction of two hallways, with each hallway being 10 meters long. We place a light at the goal point in the simulated environment, but this is for convenience only; the robot has no light sensors and therefore receives no direct information about the location of the goal.

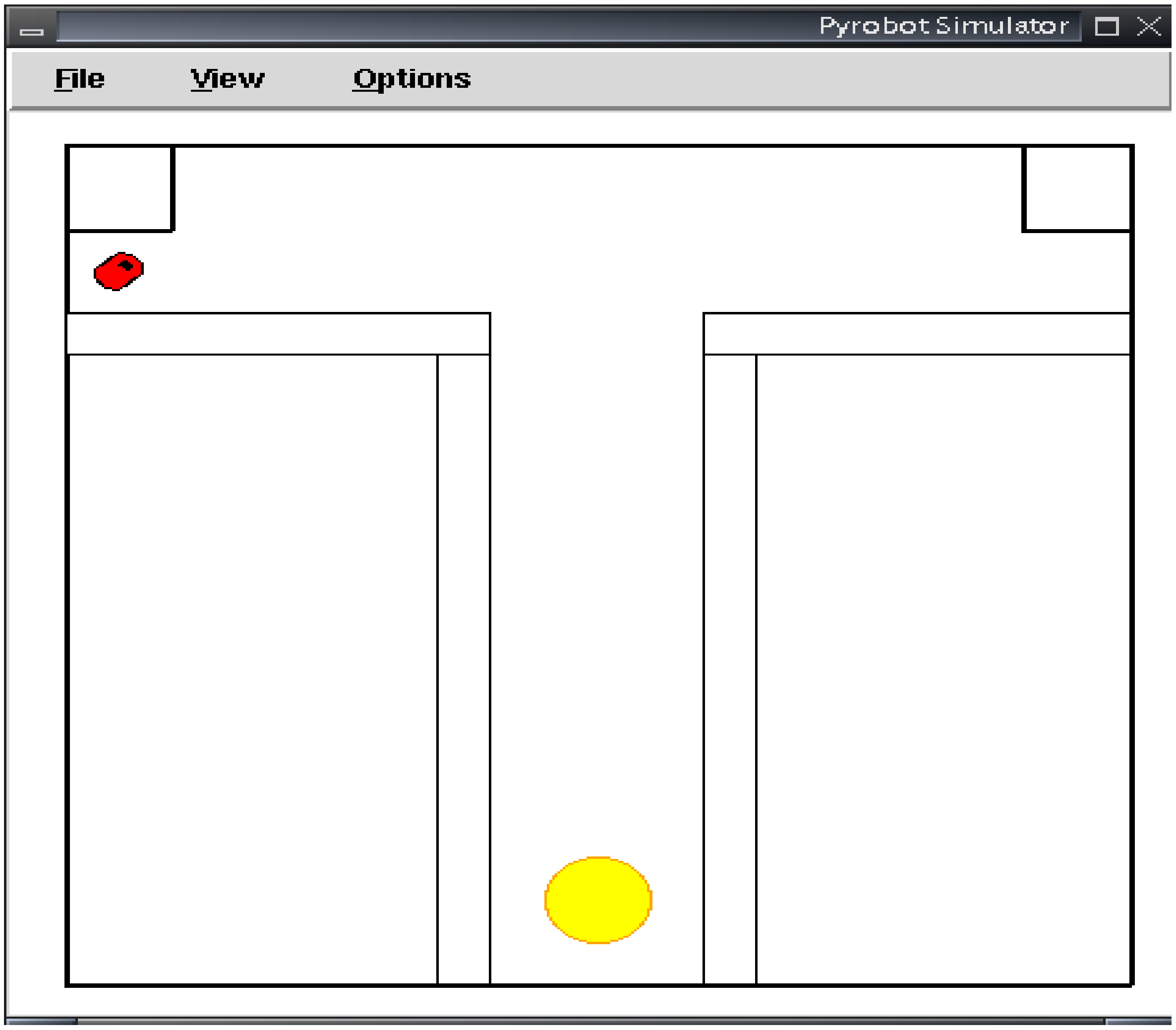

Figure 1: Our experimental setup, with the robot at its starting position and orientation.

We analyze the results of our experiments by several different methods. We calculate a running average of the robot's distance from its goal per trial (where one trial is defined as the period between resets of the robot's position and state). We calculate the number of timesteps taken for the robot to successfully reach the goal, for each successful completion of the task; we do this two ways, both including the time taken in all resets due to stalling (that is, failed attempts) and ignoring it. We also examine the geographical placement of the regions and logged the robot's entire path including the A1 action it was carrying out at any given time; however, these analyses offer no useful insights, so they are omitted from discussion.

4.1 Initial Experiment

In our first experiment, we include several additional modifications of the experiment described in [2]. We give our robot a wider range of primitive actions by including several nonorthogonal actions including both a rotational and a translational component as well as purely translational actions. We omit all purely rotational primitive actions so that the robot is always changing its distance to the goal, for an A0 action set consisting of six primitive actions.

In addition, we scale the robot's sonar sensors logarithmically. The base 10 logarithm of each raw sensor value is used in place of the original. In this way, sensor changes close to the robot are weighted more heavily, so sensor variation far away from the robot are not as salient for defining unique situations as differences near the robot. It is our hope that this will help create more meaningful sensory regions.

We try a wide variety of GNG parameters, resulting in perceptual feature sets ranging from six elements to 207 elements. With too few perceptual regions, the robot has no way to determine its current location with any real precision, and therefore cannot intelligently choose an action to perform based on its current progress. With too many regions, trajectory following periods become short and A1 actions therefore approach A0 actions. Our best-performing network uses 23 perceptual regions; discussion of results refers to the trials which used this region set.

Our results in this experiment are somewhat promising. The robot reaches its goal ten times in 47,327 timesteps; over the same time, the robot stalled and reset 91 times. The running average distance (Fig. 2) of the robot to its goal per trial shows a slight downward trend over time, suggesting that the robot becomes more efficient at moving toward the goal as learning progresses. Looking at timesteps per win (Fig. 3), however, does not indicate any improvement over time; but, if the time elapsed during failed trials is omitted from the calculation (Fig. 4), there is a clear, significant trend toward shorter trials. We believe that this is evidence that the robot is indeed learning to progress toward its goal. We suspect that the differences between the two metrics result from the robot's failure to develop a truly robust solution: if the robot veers too far from its learned course and "gets lost" in another part of the environment it is unable to recover and find a path to the goal. A considerable amount of time is therefore wasted on these failed attempts until the robot happens to stall and return to its starting position.

Another issue, we believe, is insufficient diversity in the perceptual regions formed, which may partially account for the robot's inability to recover from getting lost; it does not have enough distinct regions to differentiate the various parts of the environment. Looking at the regions formed (Fig. 5), there do not appear to be many truly distinct regions; particularly

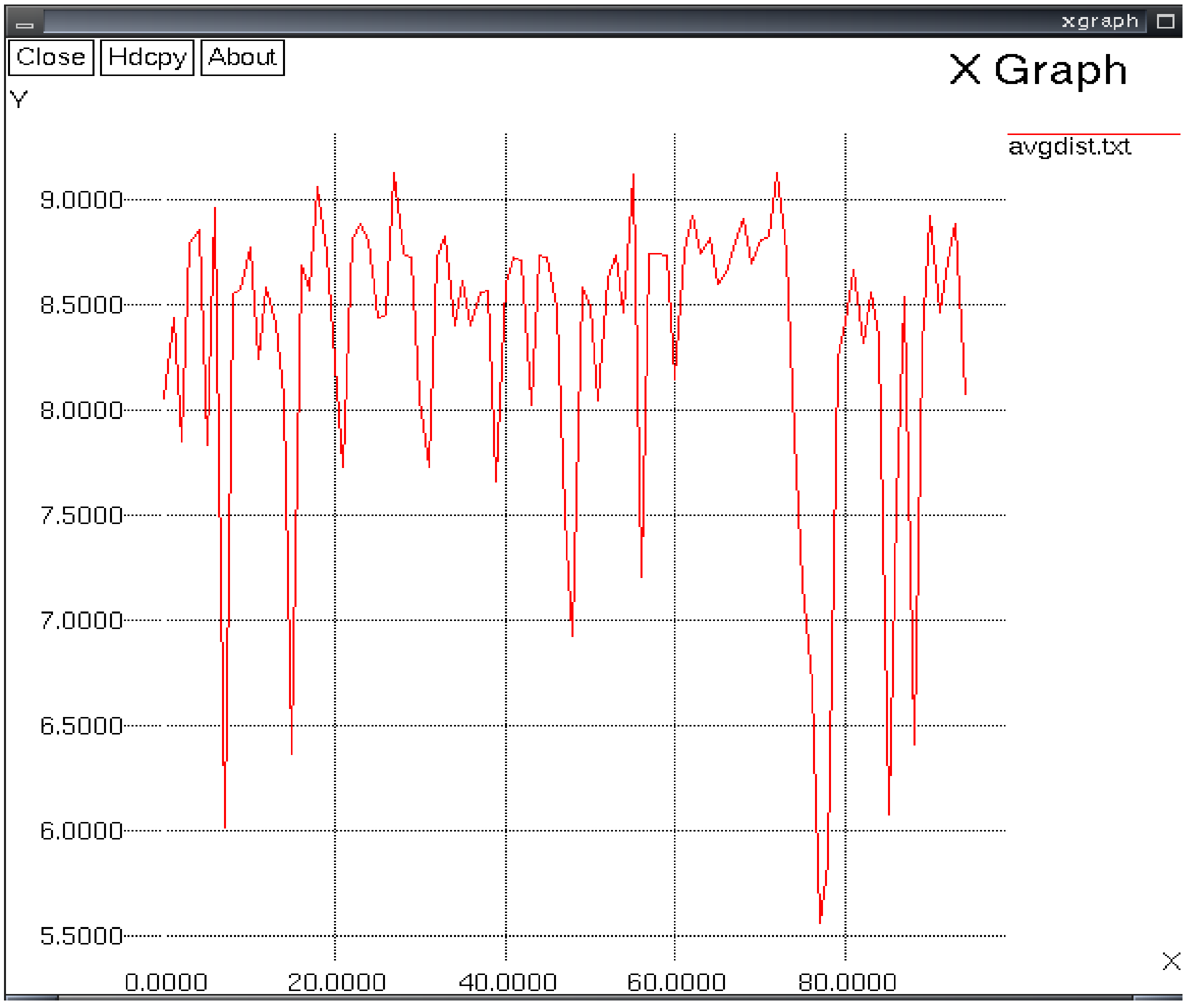

Figure 2: Running average distance to goal per reset, experiment I

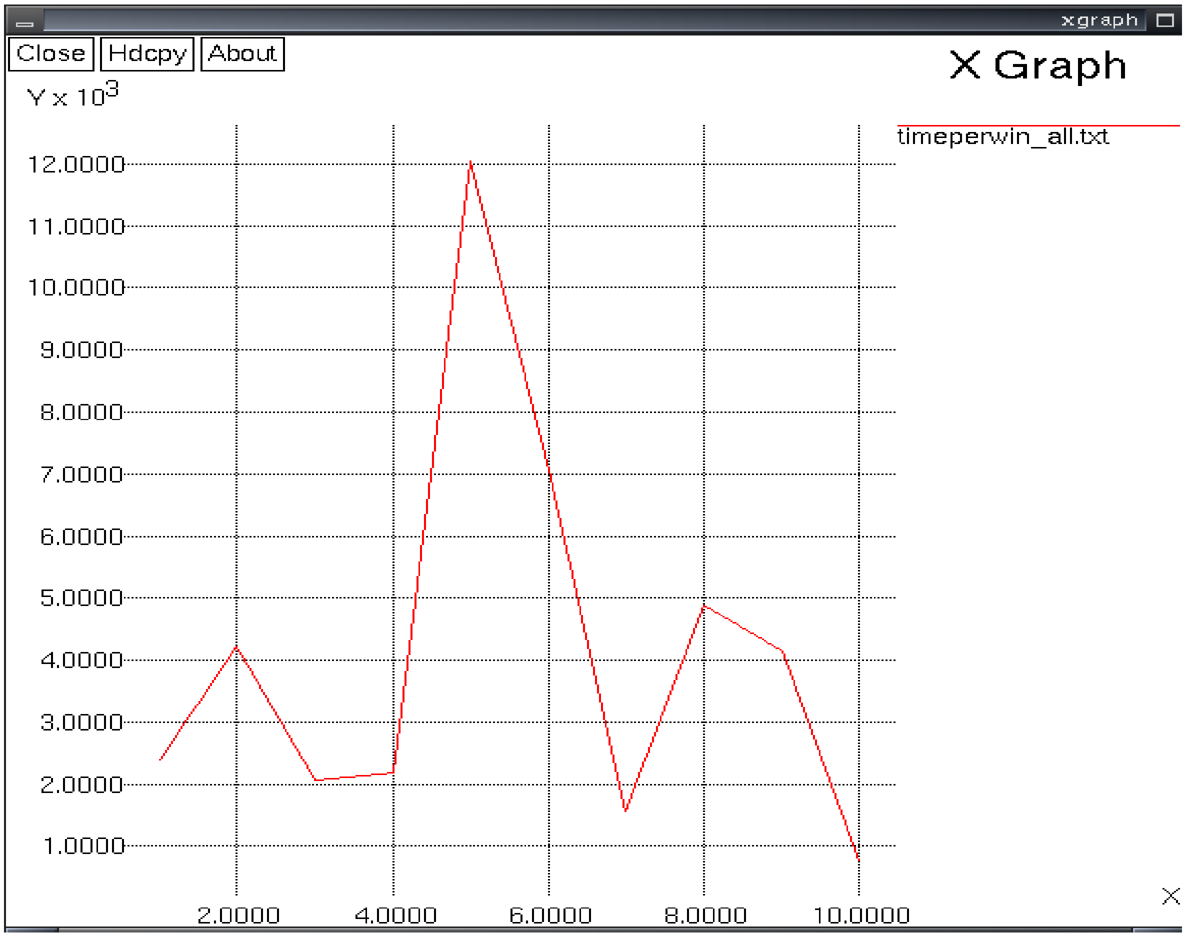

Figure 3: Elapsed time per win, including time from failed attempts, experiment I

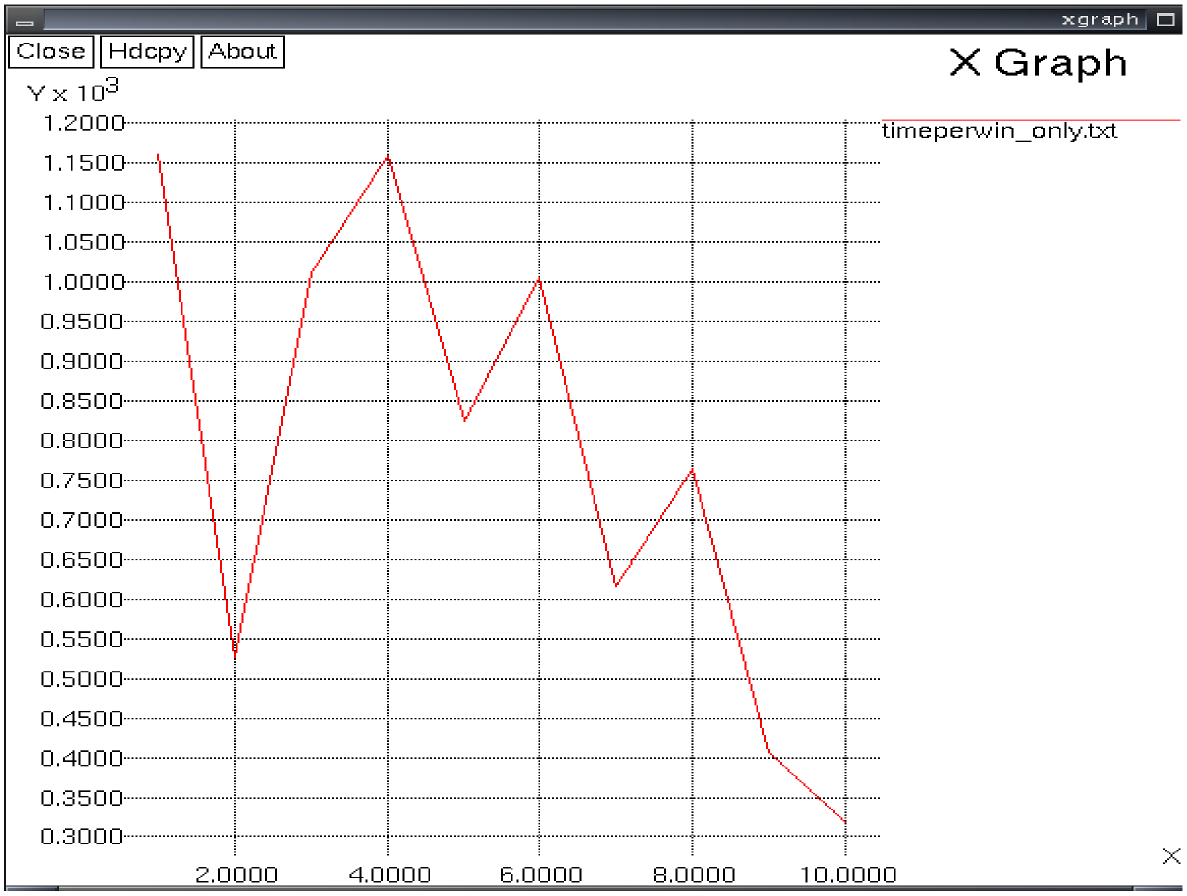

Figure 4: Elapsed time per win, omitting time from failed attempts, experiment I

striking is the lack of regions that might signify open hallways in different orientations. We believe that this might be a result of our logarithmic scaling of the sonar inputs.

4.2 Second Experiment

For the second experiment, we remove the log scaling of sonars to allow wider variation in the robot's sensory space, in an attempt to address the lack of diversity in perceptual features found in the first experiment. We generate a new GNG, using this new input scale, and we attempt to create a GNG with more considerably more nodes than that of the first experiment. The resulting GNG (a sample from which is shown in Fig. 6) contains 164 regions and does appear to have more variety than even the largest feature set developed in the first experiment.

In addition, we run a separate robot using only A0 actions, choosing a new action based on its neural network output at each timestep, and we compare the results from these two robots to determine whether A1 action improves performance.

The A1 robot successfully completes the task eight times in 8,498 timesteps, with 30 resets due to stalling. This is a major improvement over the first experiment; the robot reaches the goal almost five times as often as in the first experiment. The A0 robot, meanwhile, reached its goal only twice in 225,187 timesteps, with 862 stalls, a factor of 106 less frequently than the clearly superior A1 robot. In addition, the number of timesteps in both successful trials for the A0 robot was approximately a factor of ten greater than the average length of a successful A1 trial.

Running-average analysis yields no significant results in this trial; however, elapsed time per win does show a downward trend for the A1 robot, whether time from failed attempts is included (Fig. 7) or omitted (Fig. 8), an improvement over the first experiment which suggests that the removal of logarithmic scaling did indeed improve the robot's ability to navigate the environment.

4.3 Third Experiment

In the third experiment we attempt to more closely follow the conditions of the original experiment in [2]. We remove the robot's back sonar inputs, giving it only 180° visibility, and we redesign the A0 action set to match that in [2], allowing only four actions: forward or backward translational motion and clockwise or counterclockwise rotational motion.

Also following [2], we divide each experimental run into discrete trials with fixed maximum lengths. We reset the robot's position and state after 10,000 timesteps if the robot has not already reset to its starting position as a result either of stalling or of reaching its goal.

Because the input space is again modified, we develop another set of perceptual features. Despite trying a wide range of GNG parameters, we are unable to generate a large set of features; the largest set found (Fig. 9) contains only 15 units.

This third experiment fails consistently: in 10,822 timesteps, the robot never successfully reaches the goal, and in fact the average distance (Fig. 10) shows an upward trend over time. We suspect that both halving the available sonars and forcing the robot to reset after a fixed time hinder the robot's learning.

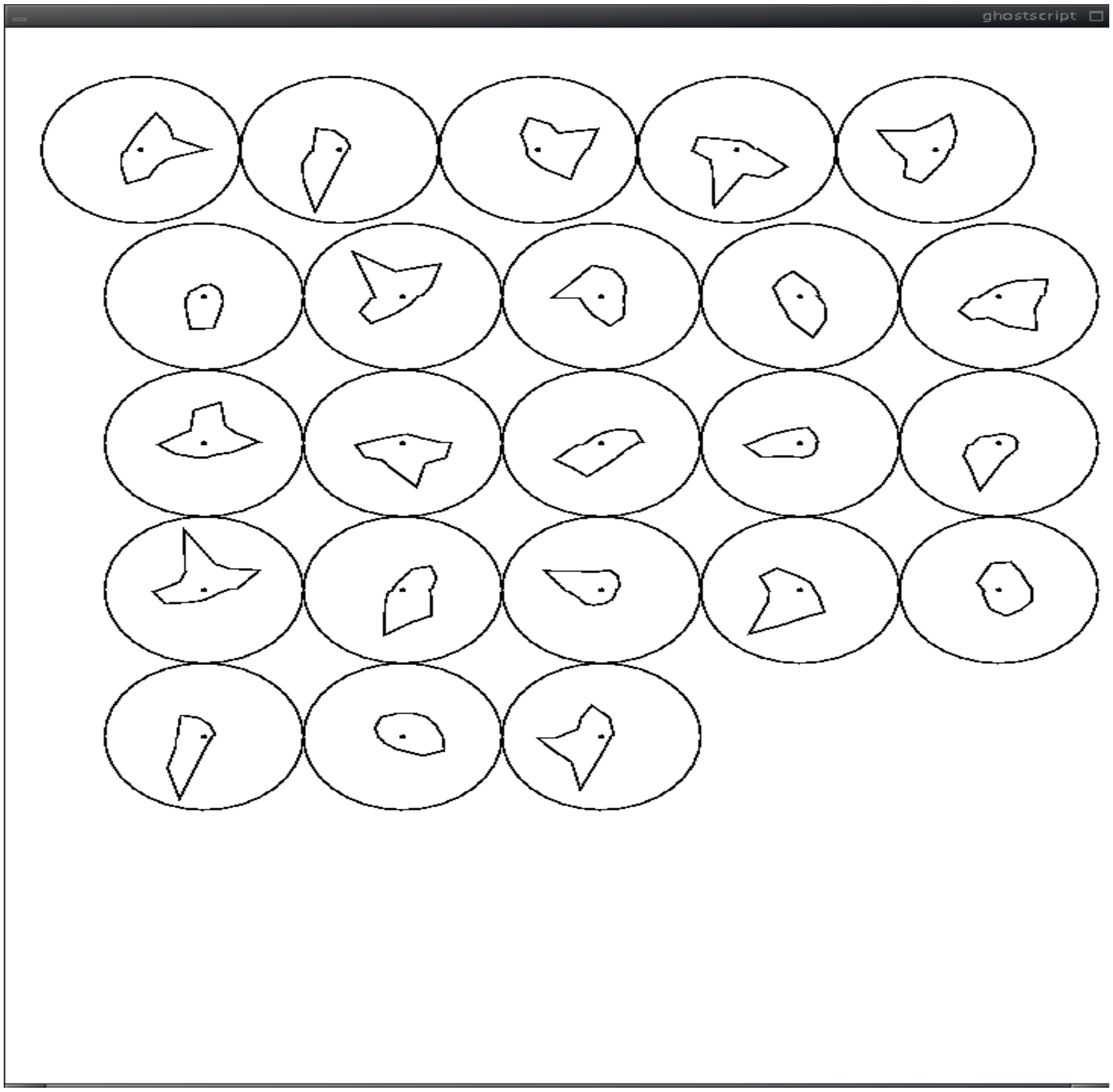

Figure 5: The perceptual regions in our best performing runs of experiment I

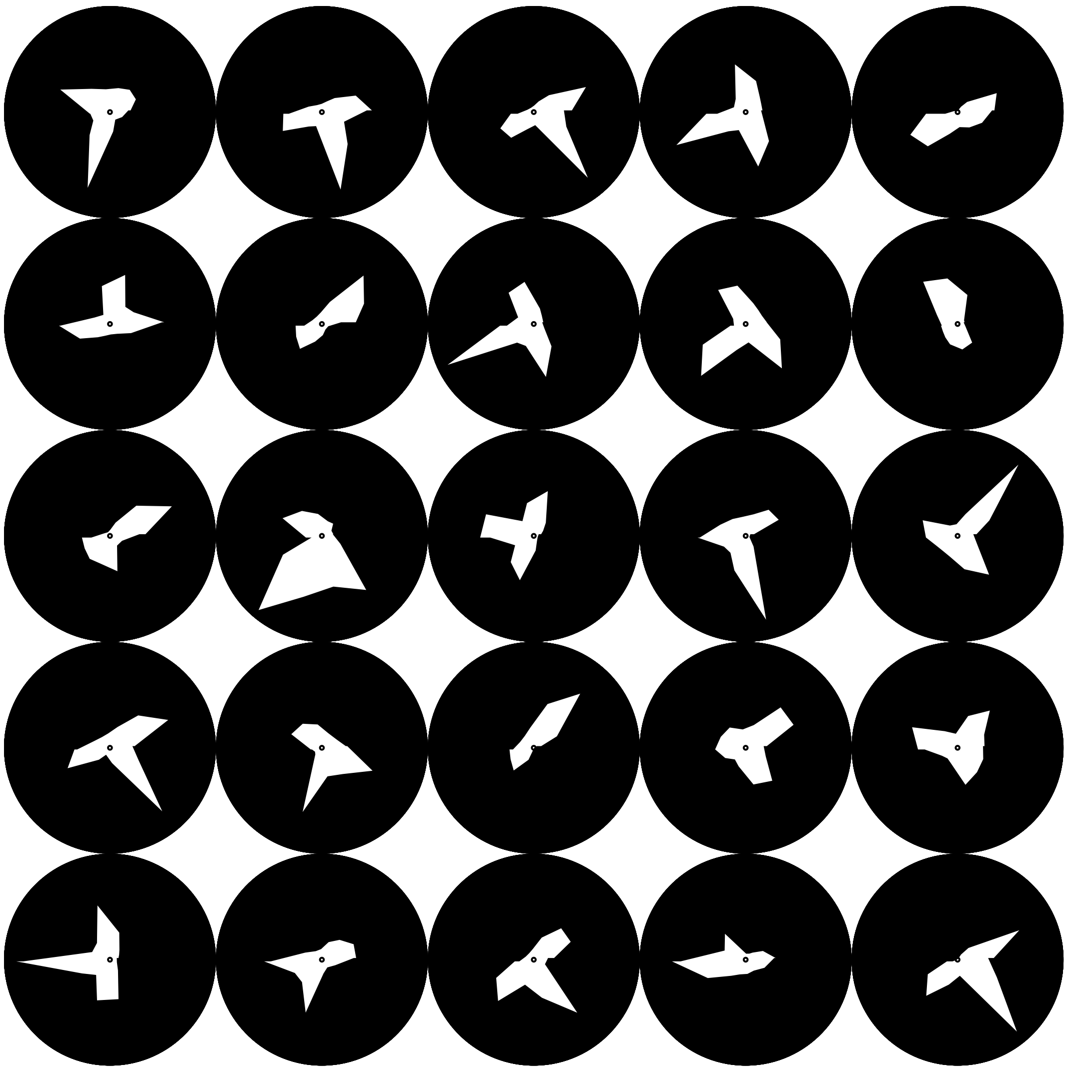

Figure 6: A sample of the perceptual regions developed in experiment II

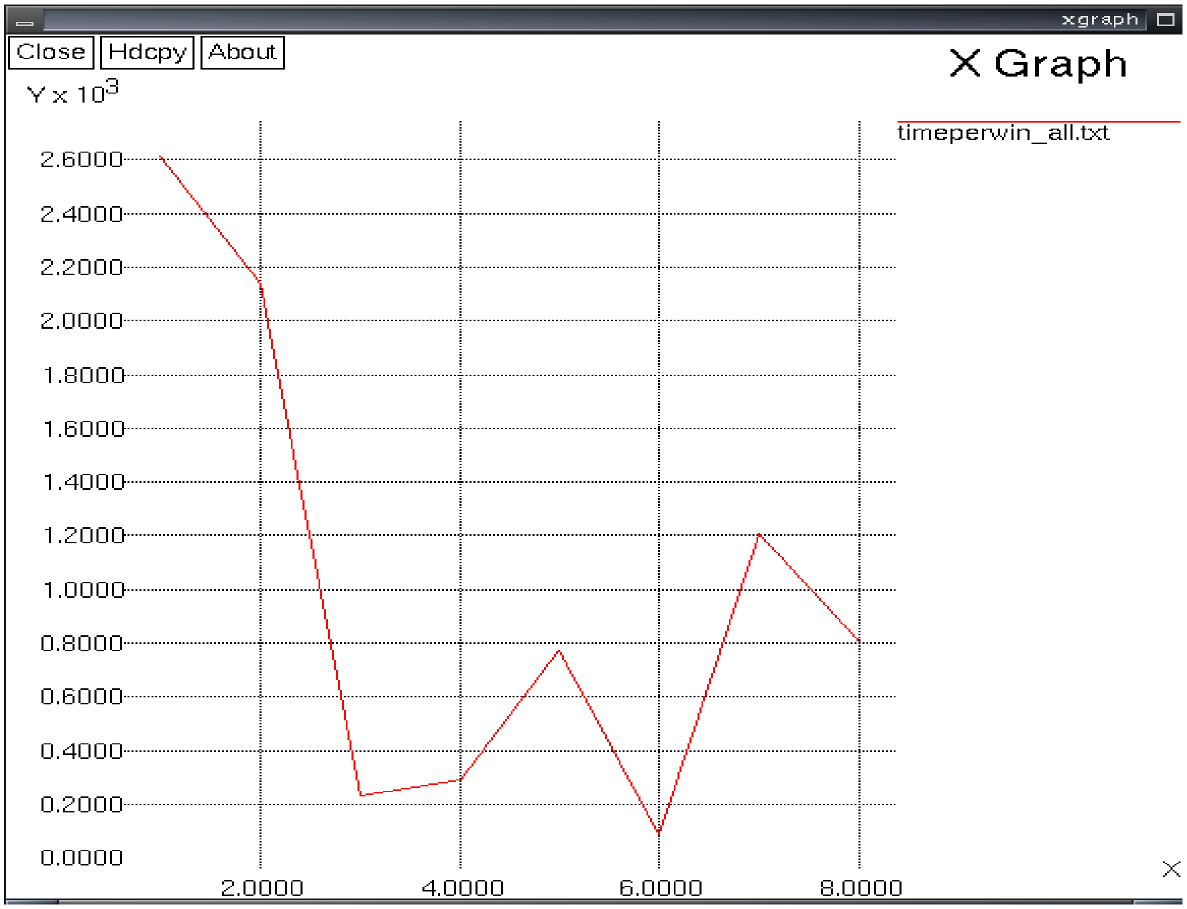

Figure 7: Elapsed time per win, including time from failed attempts, experiment II.A1

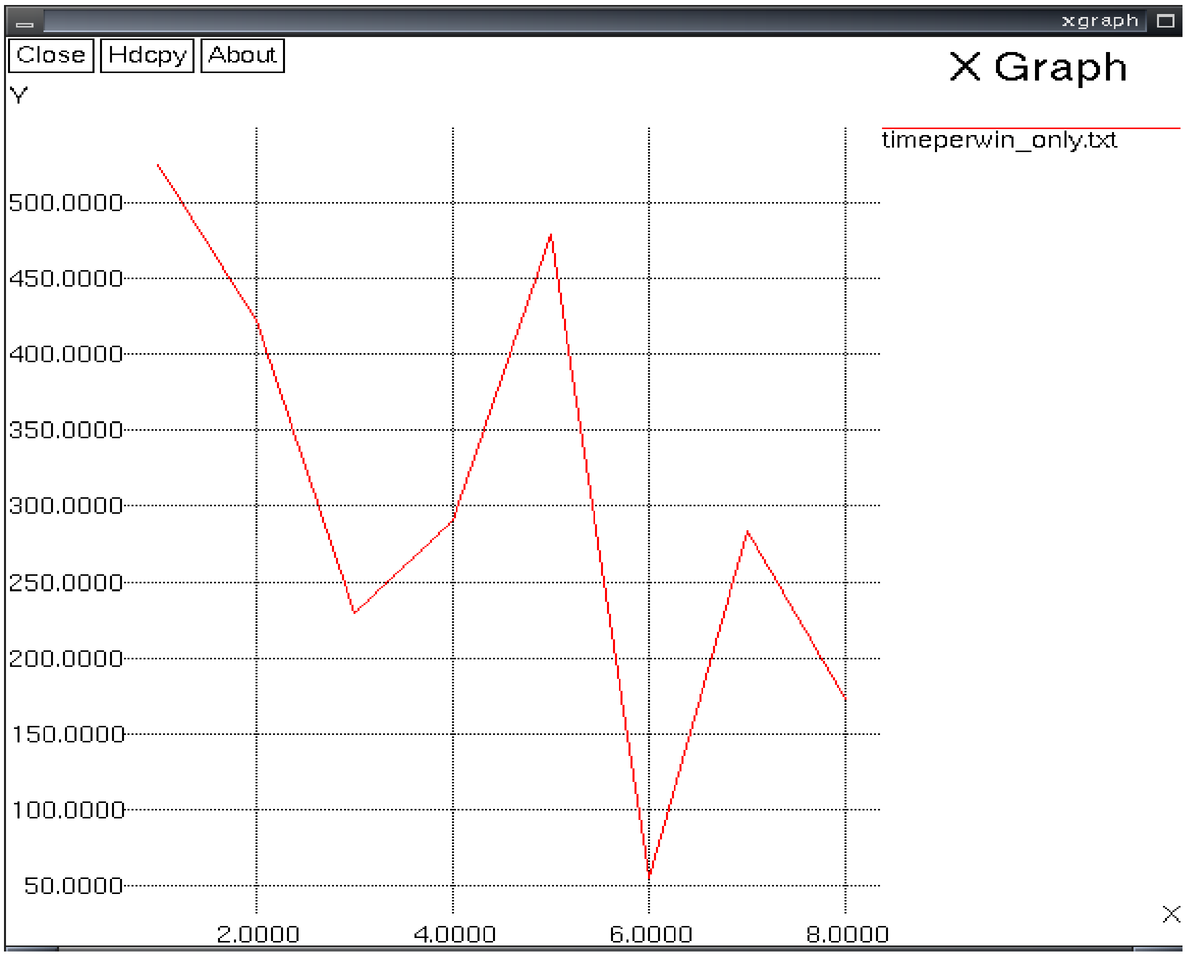

Figure 8: Elapsed time per win, omitting time from failed attempts, experiment II.A1

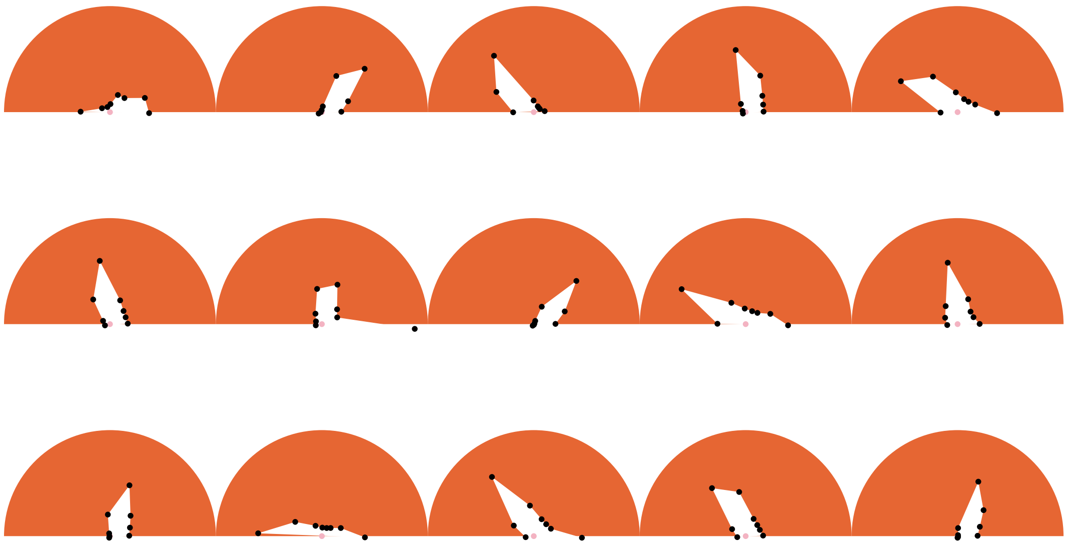
Figure 9: The perceptual regions in our best performing runs of experiment III and IV

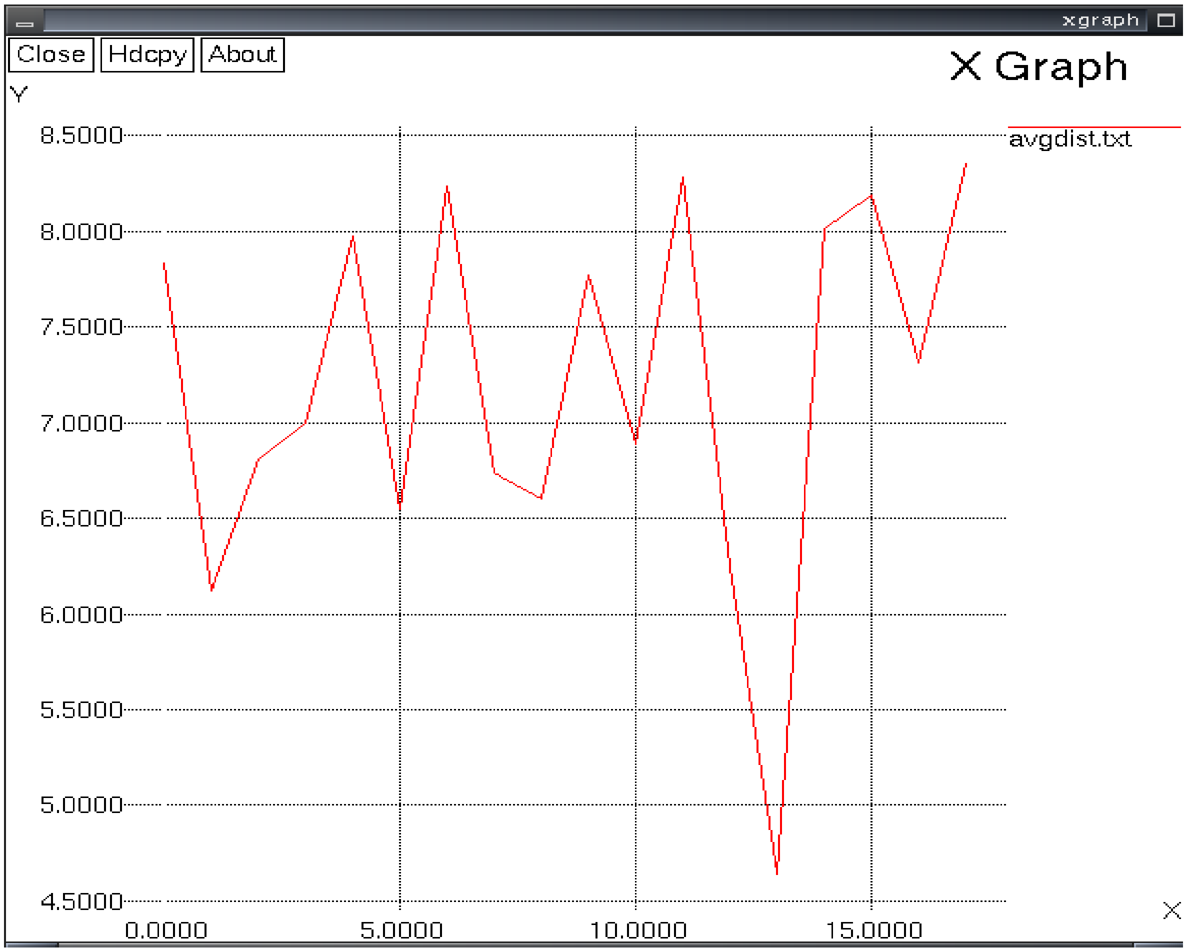

Figure 10: Running average distance to goal per reset, trial III

4.4 Fourth Experiment

The fourth experiment is almost identical to the third, with the one modification of removing the trial length limit added in the third experiment to determine whether this fixed limit does prevent the robot from learning successfully.

This experiment was a significant improvement over the third: the robot completes the task 18 times in 50,255 timesteps, with only five resets due to stalling. Wins are 1.7 times more frequent than in the first experiment and 2.6 times less frequent than in the second experiment. It seems, therefore, that the fixed trial length limit does indeed hinder the robot's progress, but other factors, most likely the reduced sonar field, also play a role.

5 Conclusion

Our results vary considerably depending on the parameters of the experiment and the predefined abilities of the robot. In our best performing experiment, we do find that A1 action selection is a significant improvement over A0 action selection. Modification of the robot's sonar viewfield results in a factor of two decrease in performance over our most successful experiment, while logarithmic scaling of the sonars results in a factor of five decrease in performance.

However, even our best results never approach the near-optimum solutions reported in [2]. This could be a result of either of the major changes between our experiment and the original: the use of sonars rather than a laser rangefinder and the replacement of the Sarsa(λ) learning algorithm with a CRBP neural network. The former possibility in particular is supported by the performance variation of our own experiments with different sonar configurations and scaling; additional experiments using sonars with Sarsa(λ) and a laser rangefinder with CRBP networks would determine which of these factors is most significant.

References

[1] Bernd Fritzke. A growing neural gas network learns topologies. In *NIPS*, pages 625–632, 1994.

[2] Jefferson Provost, Benjamin J. Kuipers, and Risto Miikkulainen. Developing navigation behavior through self-organizing distinctive state abstraction. *Connection Science (accepted for publication)*, 18(2), 2006.