# Towards Automatic Shaping in Robot Navigation

Todd S. Peterson, Nancy E. Owens & James L. Carroll
Computer Science Department
Brigham Young University
todd@cs.byu.edu, owens@cs.byu.edu, james@jlcarroll.net

May 14, 2001

## Abstract

Shaping is a potentially powerful tool in reinforcement learning applications. Shaping often fails to function effectively because of a lack of understanding about its effects when applied in reinforcement learning settings and the use of inadequate algorithms in its implementation. Because of these difficulties current shaping techniques require some form of manual intervention. We examine some of the principles involved in shaping and present a new algorithm for automatic transferral of knowledge which uses Q-values established in a previous task to guide exploration in the learning of a new task. This algorithm is applied to two different but related robot navigation tasks.

## 1 Introduction

Reinforcement learning is an attractive method for developing robot behaviors because it does not require a model of the environment, nor does it require the designer to specify the exact behavior of the robot in every situation that may occur. There are a few key concerns when applying reinforcement learning in robot settings: first, reinforcement learning can be very slow in large domains; and second, policies learned in reinforcement learning apply only to single tasks. One method for overcoming these concerns is to transfer knowledge gained from one task to the performance of a new task through a process called shaping. Unfortunately, current shaping methods require manual intervention.

### 1.1 Reinforcement Learning

Reinforcement learning [11] is a class of algorithms that uses temporal differences to learn optimal control policies (or near optimal policies in the case of learning with function approximation) for agents that are situated in a real or simulated environment.

A popular on-line algorithm for learning the optimal policy function is *Q-learning* [14]. In Q-learning the policy is formed by determining a Q-value for each state-action pair. A Q-value is the discounted expected on-line return for performing an action at the current state.

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \, max_a Q(s_{t+1}, a),$$

where $Q(s,a)$ is the Q-value for state $s$ and action $a$, $R(s,a)$ is the reinforcement received from taking action a from state $s$ and $\gamma$, where $0 < \gamma < 1$, is the discount factor. The update equation is

$$\Delta Q(s_t, a_t) = \alpha(R(s_t, a_t) + \gamma max_a Q(s_{t+1})),$$

where $\alpha$ is the learning rate.

In order to avoid learning sub-optimal policies the agent often explores his environment semi-randomly. One example of a semi-random exploration policy is to choose actions through a Boltzmann distribution:

$$prob(a_t) = \frac{e^{Q(s_t,a_t)/\tau}}{\sum_a e^{Q(s_t,a_t)/\tau}},$$

where $\tau$ effects the amount of randomness in the agents actions, and decays over time.

Reinforcement learning is particularly slow when applied directly to robot settings because of the extra time needed for physical interactions. One way of decreasing the learning time in robot settings is to use a simulator to speed up the interaction of the robot with its environment. Unfortunately, inherent noise in the sensors and the actuators of the robot prevents a simulator from exactly representing the dynamics of the environment or the robot's actuators. Function approximation is often used to increase the simulation's accuracy, but is not sufficient to completely solve this problem. Because of these and other modeling difficulties, tasks learned on simulators often transfer poorly to the real robot.

The second concern with applied reinforcement learning is that policies learned to achieve one task are not easily modified for new situations. Combining function approximation with reinforcement learning allows an agent to apply a previously learned policy to new environments provided that the new environments share some underlying environmental characteristics, but it does not facilitate the modification of a learned policy to match new goal conditions. If the underlying environmental characteristics do not match, or if goal conditions are sufficiently different, further training is required in order to achieve adequate performance.

### 1.2 Robot Shaping

Because of these and other difficulties, several researchers have used various shaping techniques to transfer knowledge from one problem domain to another. Shaping is a term used in animal psychology [8, 9] to describe a process in which an animal is trained to perform a complex behavior in stages. The animal is first trained to perform a very simple task, and is then retrained to perform similar, although slightly more difficult, tasks in gradual degrees until the desired behavior is attained. Shaping covers a variety of different approaches (ex. [2, 5, 7, 13]), each of which is somewhat successful. However, each of the techniques requires some form of manual intervention.

The first approach to shaping focuses on manually changing the reward structure of the problem so that the problem is easier to learn [13, 3, 5, 4]. For example a gradient reward was added to the terminating reward in our previous work on a hierarchical reinforcement learning strategy applied to a mine field navigation task [10]. This approach to shaping has shown to be effective in speeding up the learning process, but it requires intimate knowledge of the environment in order to appropriately place intermediate rewards.

Another approach to shaping is to change the physics of the problem in order to make the problem easier to learn. This approach was applied to a mountain car task [7]. The height of the mountain was gradually increased for each run, enabling the agent to learn to climb the mountain faster than if the task is learned on just the final height. This approach has also shown to be effective, but it is impractical to change the physics of a real-world environment, and also requires knowledge of how to appropriately change the physics. It can be practical to apply this approach to shaping in a simulator, then transfer the knowledge learned in a simulator into a robot or other physical system.

In the last approach to shaping, the policy learned for one task is modified through reinforcement learning in order to perform a new task [2]. The idea here is to only change the portion of the previously learned policy which is different than the original. In order for the task to be learned in less time than just learning the task from scratch, Bowling et. al. needed to fix the overlapping portion the task so that it could not be modified. This required manual intervention in order to determine which portions should be learned and which should remain fixed.

Although each of the shaping techniques described above allowed the agent to learn complex tasks faster than traditional reinforcement learning techniques, they all required some form of *design intervention.*[1] Part of the reason intervention is required is in the subtleties involved in appropriately transferring previously learned policies.

In this work we demonstrate the failure of a naive approach to shaping on a simple wall-following task. We then present an analysis of shaping in situations where the problem dynamics stay the same, but where the reward structure of the problem is different, and present an algorithm for applying shaping in this situation that doesn't require manual intervention. We then demonstrate the success of this technique on simplified simulations, and on a more realistic simulation of obstacle-avoidance and wall-following.

## 2 The wall-following task

Using a reinforcement learning agent in a simulated environment, we explored the application of shaping in learning a wall-following behavior. Specifically, the agent has as 8 sonars inputs, evenly spaced throughout 360 degrees. The agent is given a reward of 1.0 whenever the agent detects a wall with its left or right sonar and is within 35% of its maximum sonar range. The agent is given a reward of $-1.0$ for colliding with a wall.

The agent has a choice of five different actions, each of which represents the desired turn angle

[1] manual intervention by the designer of the system

for the robot. The actions include a left or right turn of 0, 20, or 60 degrees. The robot must continue moving forward as it turns, and cannot move backwards. The agent uses a CMAC [1] to approximate the Q-values for each action. The CMAC has 10 layers, each layer containing 3 bins per input.

A natural precursor to the wall-following task is the task of wall-avoidance. In this task the agent was trained to avoid collisions with walls. Specifically, the agent was given a reward of $-1.0$ whenever the agent collided with a wall, and a reward of 0.0 otherwise. Final Q-values from the learned wall-avoidance task were then used as the initial Q-values in a wall-following task.

We trained five agents directly on the wall-following task, and five additional agents first on wall-avoidance and then on wall-following. Figure 1 shows the average learning curve of the agents trained from scratch on the wall-following task. Figure 2 shows the average learning curve of the agents trained on the wall-following task through shaping. As can be seen in Figure 2 this approach to shaping was dismally inadequate.

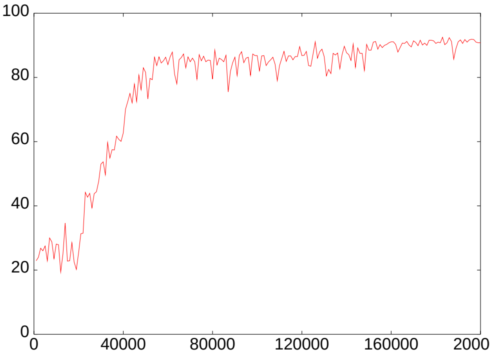

Figure 1: Average of 5 runs of learning wall-following from scratch. Shows the percentage of time the agent is performing wall-following.

Not only did the shaping attempt not learn the desired task more quickly than a “control” agent learning the task from scratch, but the agent failed to learn the task *at all* within $200,000$ time steps (a more than generous allotment of time). Other researchers have also noticed this phenomenon Bowling98[2].

This unexpected result lead us to a more careful examination of the principles involved in shaping, particularly in shaping situations which involve modified reward structures in a static environment. The hope was that a better understanding of the mechanics of shaping would lead to an effective algorithm which could be successfully applied to various shaping situations.

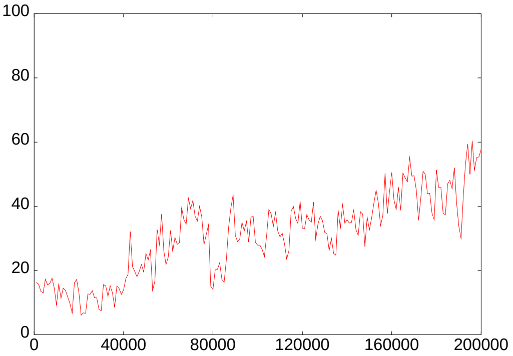

Figure 2: Average of 5 runs of learning wall-following through shaping. Shows the percentage of time the agent is performing wall-following.

## 3 Analysis

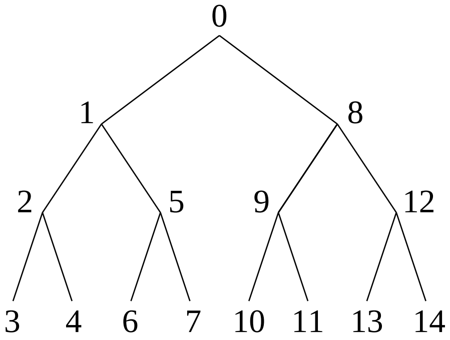

Figure 3: Simple analysis task. Agent starts task at node 0 and terminates at one of the nodes at the bottom of the tree.

We begin our discussion of shaping by considering a very simple decision task shown in Figure 3. In the task the agent starts at the top node of the tree. Each of the leaf nodes of the tree are termination nodes for the decision problem. One of the paths (unknown to the agent) terminates in a reward. The agent must learn to consistently choose the path which leads to the reward. This task is very easy and is learned in about 10 iterations using a best-first exploration strategy.

In the second phase of training, the task is changed by altering the location of the reward. If the reward remains close to its previous location (is moved perhaps one node to the right) then a large portion of the optimal policies of the first and second tasks are identical. Intuition suggests that information acquired during the first task should be easily applicable to the second, making this an ideal problem for shaping.

Several shaping approaches were tried, but each failed to significantly decrease learning time.

### 3.1 Direct Transfer of Q-values

If the agent trains on the initial task until all of the Q-values reach their optimal values, all of the Q-values in the whole tree will be zero, except for those on the branch leading towards the goal. When the agent is retrained on the task, using Q-values learned from the previous task as the initial Q-values for the new task (a technique which we call direct transfer of Q-values), the agent takes slightly *longer* to converge to an optimal policy.

The reason for this extra time is explained by considering that the agent is losing information about its environment faster than it is gaining information. If we assume a discount factor of .95 and a learning rate of 1.0 (which is sufficient for a deterministic world), and a best-first (deterministic) exploration strategy, the results of training are as follows. During the first episode of training the agent (incorrectly) assumes that the goal will be exactly where it was. The agent receives a reward of 0.0, and updates its Q-values to be zero. At this point, all of the Q-values actions leading towards leaf nodes are zero. This Q-value of 0.0 is propagated backwards each successive episode until the value of the root node is also updated to zero. The agent is then required to do a blind search for the reward.

We might assume that the learning rate or the discount factor is the problem here. However, similar problems arise with other learning rates. When a reward is moved from its original position the original policy must be unlearned. The difficulty is that if the learning rate is too low, the agent spends even *more* time unlearning the task than it would take to learn the initial task with a high learning rate.

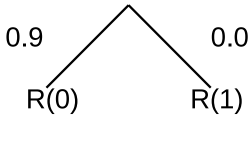

Figure 4: Simple decision problem. If Q-values are initialized as shown, the problem is difficult to learn.

For example consider the above simple decision problem. When the reward is moved just one position to the right the agent has a very difficult re-learning task. It must drop the left Q-value of .9 all the way down to say .49, while moving the 0 up to .5.

Thus the learning rate must balance the agent's need to unlearn incorrect old information, while preserving old information which was correct. Best performance was achieved with a learning rate near 0.5. Changing the discount factor had little effect.

### 3.2 Modified Prioritized Sweeping

To help unlearn old incorrect information we tried an algorithm which propagates the change in Q-values more quickly, (similar to prioritized sweeping [6].) The modified prioritized sweeping doesn't use a model of the environment, it simply keeps a history of the past $n$ updated nodes, which will be the state's predecessors, and propagates updates back to these nodes.

Modified prioritized sweeping decreased the overall search time for both shaped and traditional reinforcement learning, but did not decrease the shaping time in comparison to the original learning time.

### 3.3 Alternate Exploration Strategies

We also tried several alternate exploration strategies [12] including recency-based, counter-based, and error-based exploration. None of these methods work in conjunction with direct transfer of Q-values for the same two reasons: First, if the learning rate is too high, correct information is overwritten as new Q-values are updated. Second, if the learning rate is low enough to prevent the overwriting of good information, it takes too long to unlearn the incorrect portion of the previously learned policy.

## 4 Memory-Guided Exploration

One solution to these problems is to keep the old Q-values in a separate place, and initialize the new Q-values to the same starting position as when learning from scratch. This is advantageous because the agent doesn't need to spend additional time unlearning incorrect information.

As learning commences, all of the Q-values are set to some initial value such as 0.5. Even when a small learning rate is used, the policy becomes correct with one update, even though Q-values may be far from their optimal values.

For example, in the decision problem shown in Figure 5 only one iteration is required to find the optimal policy, (even if the agent makes the wrong choice), as compared to many iterations in the decision problem shown in Figure 4.

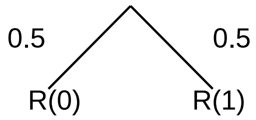

Figure 5: Simple decision problem. If Q-values are initialized to 0.5, the problem is learned in one iteration.

If the old Q-values are not used to initialize the new Q-values, how then are they used to facilitate shaping? If the old Q-values can be used to guide the agent in its initial exploration, the agent should be able to find the optimal policy much faster. The update of the new Q-values is done from scratch, and is unaffected by the old Q-values. In this way the agent is prevented from having to unlearn an incorrect policy.

One exploration policy that worked well was a version of choose best exploration with an evaluation function of:

$$Eval(s,a) = W_1 * Q(s,a) + W_2 * oldQ(s,a),$$

where $W_1$ and $W_2$ are weights that represent the amount of consideration that should be given to each value. $Q(s,a)$ represents the current Q-value of state $s$ and action $a$, and $oldQ(s,a)$ represents the Q-values learned in the primary task. $W_2$ should eventually decay to 0.0 so that the final policy is based only on the new Q-values. This ensures that the agent will eventually learn the new task.

The effect of this exploration policy is to initially bias exploration near the previous policy. $W_2$ is set to be small enough that if the previous policy is incorrect, as Q-values are updated the exploration shifts toward the new Q-values. If the optimal policy is similar to the old policy then this will significantly reduce the time required to learn the new task. In cases where the primary and secondary tasks are dissimilar, shaping would normally be inappropriate. However, this shaping algorithm requires only a few more time steps to learn a dissimilar task than it would to learn it from scratch. Therefore, a robot encountering a completely unknown problem can safely choose to apply shaping if a primary policy has been learned previously.

## 5 Results

We applied memory-guided exploration to the simulated wall-following task. The initial exploration weights of $W_1 = 1.0$, $W_2 = 0.1$. $W_1$ remained constant and $W_2$ decayed to 0 by $50,000$ steps.

Figure 6 shows the agent learning the wall-following task from scratch, and Figure 7[2] shows the agent using the memory-guided exploration algorithm.

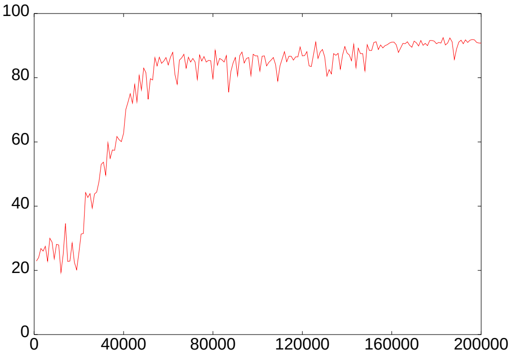

Figure 6: Average of 5 runs of learning wall-following from scratch. Shows the percentage of time the agent is performing wall-following.

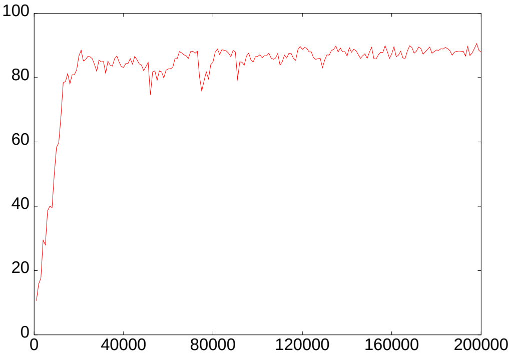

Figure 7: Average of 5 runs of agent shaping wall-following using memory-guided exploration. Shows the percentage of time the agent is performing wall-following.

Using the new exploration policy, the agent learned the same shaping task in $20,000$ steps as compared to $50,000$ steps to learn the task from scratch, and more than $200,000$ steps with direct transfer of Q-values.

[2] The CMAC offsets were incorrectly restored during information transfer, further complicating the task. Direct transfer failed to compensate, while memory-based exploration succeeded. Preliminary data suggests that while direct transfer can perform better than shown here, memory-based exploration is still more effective.

# 6 Conclusion

Effective shaping in a reinforcement learning setting can be difficult. This happens for two main reasons, first the agent must unlearn a biased incorrect policy, and second as new Q-values are updated useful information about the old policy is lost. These difficulties can be overcome through the use of a memory-guided exploration policy. This policy allows the agent to initialize and update Q-values normally, while retaining knowledge of the previous policy which can be used to guide the agent's exploration.

This algorithm was shown to be effective in shaping an obstacle-avoidance behavior into a wall-following behavior. Memory-guided exploration attained the desired behavior much more quickly than either learning the task from scratch or the use of a naive shaping algorithm based on direct transfer of Q-values. This algorithm also differs from previous shaping approaches in that it is more easily automated.

# 7 Future Research

The successful development and application of this algorithm suggest several potential avenues of future research. Examination of our preliminary graphs indicates that the proficiency of an agent in learning a shaped task is dependent upon how fully the original task was learned. Further research must be done to determine whether this is the case for all shaping situations or only for certain shaping problems.

This exploration strategy could conceivably use memory of more than one past skill to explore in an unfamiliar situation. Thus, an agent with a repertoire of several previously learned skills could apply knowledge acquired in all of these tasks to a new problem. A filtering mechanism could be used to determine which of the previously-learned tasks are most similar to the current one.

One immediate application of this algorithm is the transfer of simulator-learned policies to real-world robots. Because of the difficulties discussed earlier, simulators rarely model a problem accurately, and policies which execute perfectly on simulators often fail to produce the desired results in the real world. However, simulated and real-world tasks have an extremely high degree of similarity. Agents trained on a simulator could use memory-guided exploration to apply simulated information to the real world. This approach would combine the quick learning rates achieved with simulators with the optimal behavior obtained through real world training.

# References

[1] J. S. Albus. A new approach to manipulator control: The cerebellar model articular controller (cmac). *Trans. ASME, J. Dynamic Sys., Meas., Contr.*, 97:220–227, 1975.

[2] M. Bowling and M. Veloso. Reusing learned policies between similar problems. In *Proceedings of the AIIA-98 Workshop on new trends in robotics.* Padua, Italy, 1998.

[3] M. Dorigo and M. Colombetti. The role of the trainer in reinforcement learning. In *Proceedings of MLC-COLT '94 Workshop on Robot Learning, S.Mahadevan et al. (Eds)*, pages 37–45. New Brunswick, NJ, 1994.

[4] M. Dorigo and M. Colombetti. Precis of robot shaping: An experiment in behavior engineering. *Adaptive Behavior*, 5:3–4, 1997.

[5] M.J. Mataric. Reward functions for accelerated learning. In *Machine Learning: Proceedings of the Eleventh International Conference.* Morgan Kaufmann, CA, 1994.

[6] A. W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[7] J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Machine Learning: Proceedings of the Eleventh International Conference.* Morgan Kaufmann, CA, 1999.

[8] B. F. Skinner. *The Behavior of Organisms: An Experimental Analysis.* Prentice Hall, Englewood Cliffs, New Jersey, 1938.

[9] B. F. Skinner. *Science and Human Behavior.* Colliler-Macmillian, New York, 1953.

[10] R. Sun and T. Peterson. A hybrid model for learning sequential navigtion. In *Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 234–239, San Diego, CA., 1997. IEEE.

[11] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.

[12] Sebastian Thrun. Exploration and model building in mobile robot domains. In *Proceedings of the 1993 International Conference on Neural Networks*, 1993.

[13] D.S. Touretzky and L.M. Saksida. Operant conditioning in skinnerbots. *Adaptive Behavior*, 5(3/4):219–247, 1997.

[14] C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, University of Cambridge, 1989.