2006

Autonomous Shaping: Knowledge Transfer in Reinforcement Learning

George Konidaris
University of Massachusetts Amherst

Autonomous Shaping: Knowledge Transfer in Reinforcement Learning

George Konidaris GDK@CS.UMASS.EDU
Andrew Barto BARTO@CS.UMASS.EDU
Autonomous Learning Laboratory, Computer Science Dept., University of Massachusetts at Amherst, 01003 USA

Abstract

We introduce the use of learned shaping rewards in reinforcement learning tasks, where an agent uses prior experience on a sequence of tasks to learn a portable predictor that estimates intermediate rewards, resulting in accelerated learning in later tasks that are related but distinct. Such agents can be trained on a sequence of relatively easy tasks in order to develop a more informative measure of reward that can be transferred to improve performance on more difficult tasks without requiring a hand coded shaping function. We use a rod positioning task to show that this significantly improves performance even after a very brief training period.

1. Introduction

Although reinforcement learning is well suited to many sequential decision problems, tasks characterized by delayed reward—where a long sequence of unrewarded actions are required to reach a reward state—remain difficult to solve quickly, both in terms of finding initial solutions, and in terms of convergence toward an optimal solution. One effective way to speed up learning in such cases is to create a more informative reward signal using "shaping rewards" (Dorigo & Colombetti, 1998; Ng et al., 1999; Perkins & Hayes, 1996) or "progress indicators" (Matarić, 1997). Unfortunately, this requires significant design effort, results in less autonomous agents, and may alter the optimal solution, leading to unexpected behavior.

We propose that agents that must repeatedly solve the same type of task should be able to learn their own shaping rewards, and thus learn to solve difficult tasks quickly after a set of relatively easy training tasks. This is accomplished by learning over two separate representations, each in a different space: a reinforcement learning representation in *problem-space* that is Markov for the particular task at hand, and one in *agent-space* that may not be Markov but that is retained across successive task instances (each of which may require a new problem-space, possibly of a different size). The agent learns to initially estimate reward for novel states from "sensations" in agent-space in order to speed up reinforcement learning in problem-space.

Although this method also applies to other types of sequential decision problems, in this paper we focus on goal-directed exploration tasks because they most clearly illustrate our point, and we present the results of a rod positioning experiment in which our method significantly improves performance after only a brief period of training.

2. Background

2.1. Shaping

Shaping is a popular method for speeding up reinforcement learning in general, and goal-directed exploration in particular (Dorigo & Colombetti, 1998). Although this term has been applied to a variety of different methods within the reinforcement learning community, only two are relevant here. The first is the gradual increase in complexity of a single task toward some given final level (e.g., Randløv & Alstrøm, 1998; Selfridge et al., 1985), so that the agent can safely learn easier versions of the same task and use the resulting policy to speed learning as the task becomes more complex.[1] Unfortunately, this type of shaping does not generally transfer between tasks—it can only be used to gently introduce an agent to a single task, and is therefore not suited to a sequence of distinct tasks.

Alternatively, the agent's reward function could be augmented through the use of intermediate shaping rewards or "progress indicators" (Matarić, 1997) that provide a more informative reinforcement signal to the agent. Ng et al.

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006.

[1] We note that this definition of shaping is closest to its original meaning in the psychology literature, where it refers to a process by which an experimenter rewards an animal for behavior that progresses toward the completion of a complex task, and thereby guides the animal's learning process. As such it refers to a training technique, not a learning mechanism (see Skinner, 1938).

(1999) proved that an arbitrary externally specified shaping reward function could be included in a reinforcement learning system without modifying its optimal policy, and Wiewiora (2003) showed that this is equivalent to using the same shaping function as a non-uniform initial state value function (Sutton & Barto, 1998). The major drawback is that this requires significant engineering effort. In this paper we show that an agent may be able to learn its own shaping function from experience across several tasks without having to have it specified in advance.

2.2. Sequences of Goal-directed Tasks

In this paper we are concerned with a sequence of goal directed exploration problems (Koenig & Simmons, 1996). In each, the agent is in an environment (characterized by a set of states and actions and transition probability and reward functions) and must get to some goal state s', where it will receive a positive goal reward, while receiving a movement penalty for each action. We are interested in the problem of the initial discovery of s', which is an embodied search problem (Koenig & Simmons, 1996; Koenig, 1999) where the agent is performing a search in an unknown environment by moving through it. This is distinct from the problem of efficiently achieving policy convergence over the entire state space once the goal has been found, for which other methods exist (e.g., Thrun 1992). This allows us to focus on the speedup we obtain in first reaching the goal state, although we must also ensure that our method does not damage later convergence to an optimal (or near-optimal) policy.

3. Learning Shaping Rewards

We propose that instead of having a very informative but difficult to engineer reinforcement signal, agents should be able to learn to augment their reward structures by learning which sensory patterns predict reward across tasks. This information can be used as a shaping function that provides a first estimate for the value of newly discovered states when learning a value function for a new task. Such an agent would start with some pre-specified (possibly random or uniform) shaping function, and then refine it in several related but distinct task instances over its lifetime.

We require that the sequence of goal-directed problems are related in the sense that the agent is required to solve a sequence of variations on the same type of task, and that there is some commonality between the tasks so that the agent can retain learned knowledge usefully across them. We thus define the notion of a *sequence of reward-linked related tasks* as follows.

The agent experiences a sequence of environments generated by the same underlying generative world model (e.g., they have the same physics, the same types of objects may be present in the environment, etc.). From the sensations it receives in each environment, the agent creates two representations. The first is a state descriptor that is sufficient to distinguish Markov states in the current environment. This induces a Markov Decision Process (MDP) with a set of actions that are fixed across the sequence (because the agent does not change) but a set of states, transition probabilities and reward function that depend only on the task the agent is currently facing. The agent thus works in a different state-space with its own transition probabilities and reward function for each task in the sequence. We call each of these a *problem-space*.

The agent also uses a second representation from the sensations that are consistently present and retain the same semantics across the sequence of tasks. This space is shared across the sequence of tasks, and we call it *agent-space*. These two representations stem from two different representational requirements: problem-space models the Markov description of a particular environment, and agent-space models the (potentially non-Markov) commonalities across a set of environments.

We thus term the tasks in the sequence *related* if the sequence consists of environments that share an agent-space. This ensures that they are generated by the same underlying world model and are experienced by the same agent. We term the sequence *reward-linked* if the reward function in each environment consistently allocates rewards to the same types of interactions across environments (e.g., reward is always x for consuming a food particle and y for reaching a light source, no matter which environment the agent is in). This ensures that there is some useful relationship and potential correlation between sensations in agent-space and reward across the sequence, which the agent can learn to exploit.

One very simple example of such a sequence is a sequence of k-armed bandit problems where arms that always give a low payoff are colored red, broken arms (that always give a zero payoff) are colored orange, and all other arms are colored green. Although the color of the arms is not sufficient to solve the problem even though there is no state space (since the agent needs to decide between the green arms), the colors are always present so the agent can learn that red or brown arms have a low expected value, and thus learn to prefer green ones.

Another example of such a sequence is a sequence of buildings where a robot that is equipped with pressure, light and temperature gauges and a map is required to find a heat source while avoiding obstacles. Each state in the problem-space is uniquely determined by the robot's map position and pose, but the sensations received at each state are meaningful across the sequence, and thus form the agent-

space. The robot can eventually learn to use its temperature gauge as a heuristic measure of proximity to the source, and thereby be able to find heat sources in more complex buildings in less time, even though this heuristic is not in general sufficient to solve the problem by itself (because the heat sensor reading is not Markov in problem-space).

3.1. A Framework for Learning Shaping Functions

The agent is solving n problems, each with its own state space, denoted $S_1, ..., S_n$. We then view the ith state in task S_j as consisting of the following attributes:

$$s_i^j = (d_i^j, c_i^j, r_i^j, v_i^j),$$

where d_i^j is the problem-space state descriptor (sufficient to distinguish this state from the others in S_j), c_i^j is an agent-space sensation, r_i^j is the reward obtained at the state and v_i^j is the state's value (expected total reward for action starting from the state). We are not concerned here with the form of d_i^j, except to note that it may contain or be disjoint from c_i^j, and we assume that estimates of the v_i^j values of previously observed states have been obtained by a reinforcement learning algorithm, learning the value function V_j:

$$V_j : d_i^j \mapsto v_i^j.$$

This function maps from state descriptor to expected return, but it is not portable between tasks because the form and meaning of d (as a problem-space descriptor) may change from one task to another. However, the form and meaning of c (as an agent-space descriptor) does not change, so we introduce a function L that preserves value information between tasks and acts as the agent's internal shaping reward. L estimates expected return for novel states, given the agent-space descriptor received there:

$$L : c_i^j \mapsto v_i^j.$$

Once an agent has completed task S_j and has learned a good approximation of the value of each state using V_j, it can use its (c_i^j, v_i^j) pairs as training examples for a supervised learning algorithm to learn L. Alternatively, training could occur online during each task, although this may result in noisy or unstable shaping functions. After a reasonable amount of training, L can be used to estimate a value for newly observed states in problem-space, and thus provide a good initial estimate for V that can be refined using a standard reinforcement learning algorithm. Alternatively (and equivalently), L could be used as a separate external shaping reward function.

4. A Rod Positioning Experiment

In this section we empirically evaluate the potential benefits of a learned shaping function in a rod positioning task (Moore & Atkeson, 1993). Each task consists of a square workspace that contains a rod, some obstacles, and a target. The agent is required to maneuver the rod (by moving its base coordinate or its angle of orientation) so that its tip touches the target while avoiding obstacles. An example 20x20 unit task and solution path are shown in Figure 1.

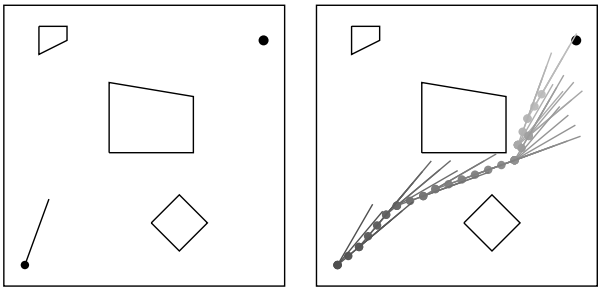

Figure 1. A 20x20 rod positioning task and one solution path.

Following Moore and Atkeson (1993), we discretize the state space into unit x and y coordinates and 10^o angle increments. Thus, each state in the problem-space can be described by two coordinates and one angle, and the actions available to the agent are movement of one unit in either direction along the rod's axis, or a 10^o rotation in either direction. If a movement causes the rod to collide with an obstacle it results in no change in state, so the portions of the state space where any part of the rod would be interior to an obstacle are not reachable. The agent receives a penalty of 1 for each action, and a reward of 1000 when reaching the goal (whereupon the episode ends).

We augment the task environment with five beacons, each of which emits a separate signal that drops off with the square of the Euclidean distance from a strength of 1 at the beacon to 0 at a distance of 60 units. The tip of the rod has a gradient sensor array that can detect the values of each of these signals at the adjacent state in each action direction. Since these beacons are present in every task, the sensor readings are an agent-space and we include an element in the agent that learns L and uses it to predict reward for each adjacent state given the five signal levels present there.

The usefulness of L as a reward predictor will depend on the relationship between beacon placement and reward across a sequence of individual rod positioning tasks. Thus we can consider the beacons as simple abstract signals present in the environment, and experimentally evaluate the usefulness of L while manipulating their relationship to reward across the sequence.

4.1. Experimental Structure

In each experiment, the agent is exposed to a sequence of training experiences, during which it is allowed to update L. After each training experience, it is evaluated in a large test case, during which it is *not* allowed to update L.

Each individual training experience places the agent in a small task, randomly selected from a randomly generated set of 100 such tasks, where it is given sufficient time to learn a good solution. Once this time is up, the agent updates L using the value of each visited state and the sensory signal present at it, before it is tested on the much larger test task. All state value tables are cleared between training episodes.

Each agent performed reinforcement learning using Sarsa(λ) ($\lambda = 0.9, \alpha = 0.1, \gamma = 0.99, \epsilon = 0.01$) in problem-space and used training tasks that were either 10x10 (where it was given 100 episodes to converge in each training task), or 15x15 (when it was given 150 episodes to converge), and tested in a 40x40 task.[2] L was a linear estimator of reward, using either the five beacon signal levels and a constant as features (requiring 6 parameters, and referred to as the linear model) or using those with five additional features for the square of each beacon value (requiring 11 parameters, referred to as the quadratic model). All parameters were initialized to 0, and learning for L was accomplished using gradient descent with $\alpha = 0.001$. We used two experiments with different beacon placement schemes.

4.2. Following a Homing Beacon

In the first experiment, we always placed the first beacon at the target location, and randomly distributed the remainder throughout the workspace. Thus a high signal level from the first beacon predicts high reward, and the others should be ignored. This is a very informative indication of reward that should be fairly easy to learn, and can be well approximated even with a linear L. Figure 2 shows the 40x40 test task used to evaluate the performance of each agent, and four sample 10x10 training tasks.

Figure 3 shows the number of steps (averaged over 50 runs) required to first reach the goal against the number of training experiences completed by the agent for the four types of learned shaping elements (linear and quadratic L, and either 10x10 or 15x15 training worlds). It also shows the average number of steps required by an agent with a uniform initial value of 0 (agents with a uniform initial value of 500 performed similarly while first finding the goal). Note that there is just a single data point for the uniform initial value agents (in the upper left corner) because their performance does not vary with the number of training experiences.

[2]We note that in general the tasks used to train the agent need not be smaller than the task used to test it. We used a small training task in this experiment to highlight the fact that the size of problem-space may differ between related tasks.

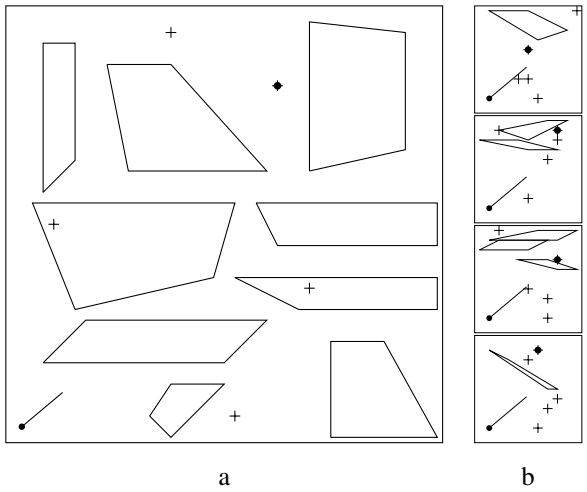

Figure 2. The homing experiment 40x40 test task (a) and four sample 10x10 training tasks (b). Beacon locations are shown as crosses, and the goal is shown as a large dot. Note that one of the beacons is on the target in each world.

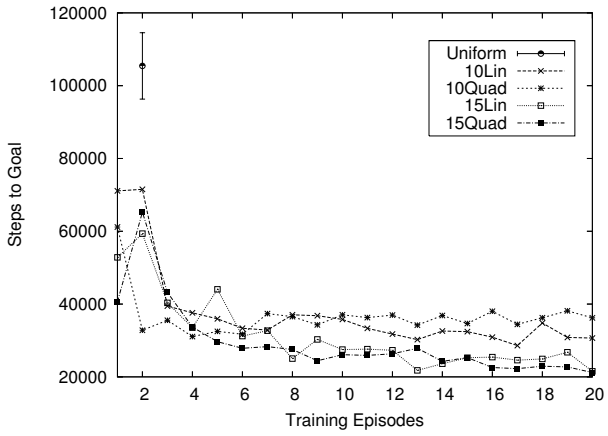

Figure 3. The average number of steps required to first reach the goal in the homing task.

Figure 3 shows that training significantly lowers the number of steps required to initially find the goal in all cases, reducing it after one training experience from over $100,000$ steps to at most just over $70,000$, and by six episodes to between $20,000$ and $40,000$ steps. This difference is statistically significant (by a t-test, $p < 0.01$) for all combinations of L and training task sizes, even after just a single training experience. Figure 3 also shows that the complexity of L does not appear to make a significant difference to the long-term benefit of training (probably because of the simplicity of the reward indicator), but the size of the training task

does. The difference between the number of steps required to first find the goal for 10x10 and 15x15 training task sizes is statistically significant ($p < 0.01$) after 20 training experiences for both linear and quadratic forms of L, although this difference is clearer for the quadratic form, where it is significant after 6 training experiences.

Figure 4 shows the number of steps (averaged over 50 runs) required to reach the goal as the agents repeat episodes in the test task, after having been allowed 20 training experiences (L is still never updated in the test world), as well as the number required by agents with value tables uniformly initialized to 0 and 500. This illustrates the overall learning performance of the agents over time on a single new task once they have had many training experiences against that of agents using uniform initial values. Figure 4 shows that the learned shaping heuristic significantly speeds up the first few episodes and does not damage convergence, taking slightly longer than an agent using 0 as a uniform initial value but about the same as that of an agent using 500. This suggests that once a solution is found the agent must then "unlearn" some of its overly optimistic heuristic estimates to achieve convergence. Note that a uniform initial value of 0 works well here because it discourages extended exploration, which is not necessary in deterministic domains such as this.

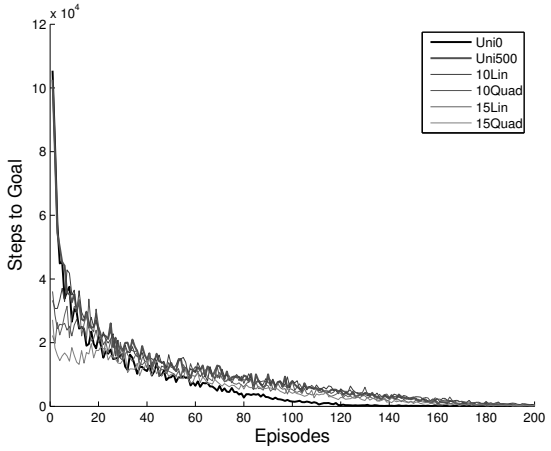

Figure 4. Steps to reward against episodes in the homing test world after 20 training episodes.

4.3. Finding the Center of a Beacon Triangle

In the second experiment, we always arranged the first three beacons in a triangle at the edges of the task workspace, so that the first beacon lay to the left of the target, the second directly above it, and the third to its right. The remaining two were randomly distributed throughout the workspace. This provides richer reward information, but should present a harder function to learn. Figure 5 shows the 10x10 sample training tasks given in Figure 2 after modification for the triangle experiment. The test task was similarly modified.

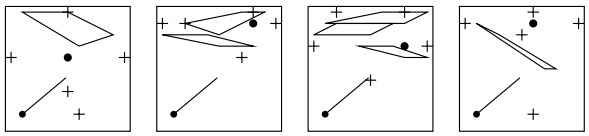

Figure 5. Sample 10x10 training tasks for the triangle experiment.

Figure 6 shows the number of steps initially required to reach the goal for the triangle experiment, again showing that even a single training experience results in a statistically significant ($p < 0.01$ in all cases) reduction from the number required by an agent using uniform initial values, from just over $100,000$ steps to at most just over $25,000$ steps after a single training episode. Figure 6 also shows that there is no significant difference between forms of L and size of training task. This suggests that the richness of the useful signal more than makes up for it being difficult to learn correctly—in all cases the performance of agents learning using the triangle beacon arrangement is better than that of those learning using the homing beacon arrangement. Figure 7 shows again that the initial few episodes of repeated learning in the test task are much faster, and that training does not affect convergence, with the total number of episodes required to converge again lying somewhere between the number required by an agent initializing its value table to 0 and one initializing it to 500.

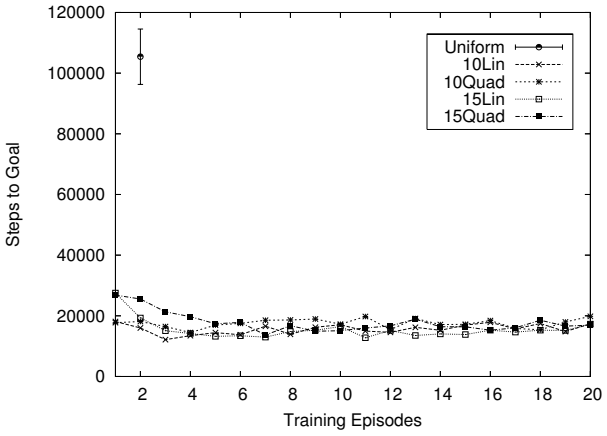

Figure 6. The average number of steps required to first reach the goal in the triangle task.

4.4. Summary

The above two experiments show that an agent that can learn its own shaping rewards through training can find an initial solution to a novel task much faster than an agent

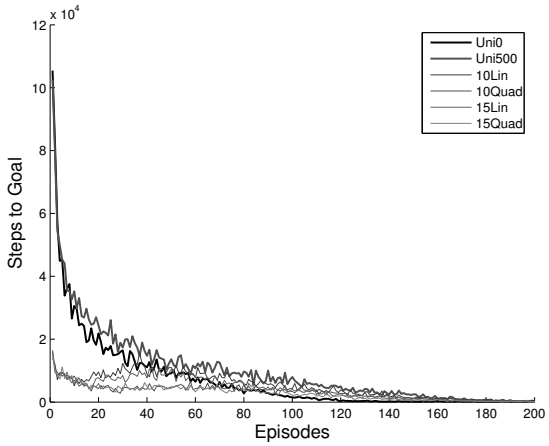

Figure 7. Steps to reward against episodes in the triangle test task after 20 training episodes.

that uses uniform initial values, even after only a few training experiences. They also show that such training does not damage the agent's convergence characteristics, even after many training episodes.

The results also seem to suggest that a better training environment is helpful but that its usefulness decreases as the signal predicting reward becomes richer, and that increasing the complexity of L does not appear to significantly improve the agent's performance. Although this is a very simple domain, it suggests that given a rich signal from which to predict reward even an inaccurate estimation of reward is sufficient to improve performance.

5. Related Work

Prior reinforcement learning research exists on finding useful macro-actions across sequences of tasks (Bernstein, 1999; Pickett & Barto, 2002; Thrun & Schwartz, 1995) and building structured representations of a state space to speed up later learning (Mahadevan, 2005; Van Roy, 1998). However, these approaches require tasks that are in the same state space and differ only in their reward functions. Taylor et al. (2005) use a hand-coded transfer function to seed one task's value function with learned values from another similar task with a potentially different state space, but they require the explicit manual construction of a transfer function between each pair of value functions. An appropriate sequence of tasks in the research presented here requires only that the agent-space semantics (including semantics with respect to reward) remain consistent, so each task may have its own completely distinct state space.

Konidaris and Hayes (2004) show using a similar method that using training mazes to learn associations between reward and strong signals at reward states results in a significant improvement in the total reward obtained by a realistically simulated robot learning to find a puck in a novel maze. The research presented here employs a more general mechanism where the agent learns a heuristic from all of the states that it has visited.

6. Discussion

The results presented above suggest that agents that employ reinforcement learning methods can be augmented to use their experience to learn their own shaping rewards. This could result in agents that are more flexible than those with pre-engineered shaping functions. It also creates the possibility of training such agents on easy tasks as a way of equipping them with knowledge that will make harder tasks tractable, and is thus an instance of an autonomous developmental learning system (Weng et al., 2000). In addition, this system provides another example of the use of layered learning systems (Stone & Veloso, 2000), and of the interesting and potentially complex behavior that results from the interaction of two learning systems.

However, the ideas presented here have some drawbacks. Determining the form of c (the agent-space descriptor) by identifying a relevant agent-space and selecting an appropriate learning method for L create a potentially difficult design problem. We expect that most of the difficulty will lie in choosing an appropriate agent-space in order to allow for the use of a relatively simple learning algorithm, and to facilitate rapid learning and short training times.

In some situations, the learning algorithm chosen for L, or the sensory patterns given to it, might result in an agent that is completely unable to learn anything useful. We do not expect such an agent to do much worse than one without any shaping rewards at all.

Another potential concern is the possibility that a maliciously chosen or unfortunate set of training tasks could result in an agent that performs worse than one with no training. Fortunately, such agents will still eventually be able to learn the correct value function (Ng et al., 1999).

6.1. Learned Shaping Rewards and Generalization through Value Function Approximation

Learned shaping rewards are used to assign initial values to novel states in problem-space in order to accelerate the learning of accurate values for those states. This is a form of generalization, where the shaping function retains knowledge from experience with the environment and uses it to better predict state values in later tasks. As such it is strongly related to the use of value function approximation for generalization across states.

In a value function approximation system, some compact value function representation (such as a neural network) is used instead of a value table, and it is trained to represent values experienced from visited states. Novel states evaluated with this value function may therefore be given values based on previous experience in similar states, and thus already include previously learned knowledge.

The use of learned shaping rewards coupled with a value function table is distinct for two reasons. First, it only generalizes forwards, and not backwards: novel states are given initial values based on generalization, but the values of previously encountered states are never disturbed. Therefore although learned shaping values do not generalize as broadly, they cannot cause an algorithm that would otherwise converge to fail to do so, which can occur with function approximation (Sutton & Barto, 1998). They may therefore be considered a safer form of generalization.

Second, value function approximation usually only generalizes within a single task. An approximated value function that is used to generalize over one MDP may not be applicable to another related but distinct MDP. The semantics of each state descriptor may change (as a trivial example, consider two MDPs that are identical to each other but with different goal states), or the size of the input to the approximator may change. A learned shaping function, by virtue of its split representation, can be used to generalize across a sequence of distinct tasks provided the agent-space semantics are consistent.

There is also an important point that should be made here: there is a formal difference between an MDP state label in problem-space and the sensory input received at that state. A problem-space state descriptor should ideally be the smallest piece of information that is sufficient to discriminate between states, so that the agent is solving the smallest possible faithful Markov model of the underlying problem. Using sensor input as a state descriptor might facilitate generalization, but it also often results in a state space that is both very large (thus necessitating generalization) and too small (because it is not Markov). It may be better to factor the sensory input so that (some function of) a small subset of it is used as a problem-space Markov state descriptor, and the remainder used by a learned shaping function or some other separate element for generalization.

This is most obviously true for navigation problems. Returning to the example of the robot learning to find a heat source in a map, the map itself is sufficient to distinguish the robot's states, and including its sensor readings into its state descriptor would vastly increase the size of the state space without changing the size of the underlying problem. It is much easier, and conceptually much cleaner, to use the compact descriptor given by some discretization of the map to generate a very small problem-space, and then use a separate learning element to generalize by learning to estimate novel state rewards from its remaining sensors.

6.2. Shaping Rewards as a Search Heuristic

It is unlikely that in any useful scenario an agent will be able to learn a completely accurate measure of value using L. If it could, we could do away with reinforcement learning altogether and simply ascend L. Instead, we expect to be able to learn a rough approximation of value that functions as a heuristic.

In standard classical search algorithms, such as A*, a heuristic gives an inexact measure for the distance between a particular node in the search space and the goal. In an embodied search, the agent must physically search some unknown environment, and thus can only keep a single node "open" at any one time. In algorithms like Learning-Real-Time A* (LRTA*) (Korf, 1990), the agent uses a heuristic measure combined with the cost of moving to the nodes it can immediately reach to determine where to go next. Since Real-Time Dynamic Programming (RTDP) is the stochastic generalization of LRTA* (Barto et al., 1995), and shaping rewards act to order the selection of unvisited nodes, shaping rewards provide a heuristic initialization of the value function in embodied search problems Therefore, agents solving embodied search problems that are able to learn their own shaping functions are learning their own heuristics.

7. Conclusion

In this paper we introduced the use of learned shaping rewards in a sequence of goal-directed reinforcement learning tasks. This is accomplished by having two separate representations: a Markov problem-space representation for reinforcement learning that differs for each task, and an agent-space representation that does not. The second representation is used to learn a shaping function that can provide value predictions for novel states across tasks in order to speed up learning in problem-space. Our experimental results show that the use of learned shaping rewards can significantly improve performance in a rod positioning experiment with even a single training experience.

Acknowledgments

We would like to thank Gillian Hayes, Colin Barringer, Sarah Osentoski, Özgür Şimşek, Michael Littman, Ashvin Shah and Pippin Wolfe for their useful comments. Andrew Barto was supported by the National Science Foundation under Grant No. CCF-0432143 and by a subcontract from Rutgers University, Computer Science Department, under award number HR0011-04-1-0050 from DARPA. Any opinions, findings and conclusions or recommenda-

tions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Barto, A., Bradtke, S., & Singh, S. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*, 81–138.

Bernstein, D. (1999). *Reusing old policies to accelerate learning on new MDPs* (Technical Report UM-CS-1999-026). Department of Computer Science, University of Massachusetts at Amherst.

Dorigo, M., & Colombetti, M. (1998). *Robot shaping: An experiment in behavior engineering*. MIT Press/Bradford Books.

Koenig, S. (1999). Exploring unknown environments with real-time search or reinforcement learning. *Advances in Neural Information Processing Systems (NIPS) 12* (pp. 1003–1009).

Koenig, S., & Simmons, R. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, *22*, 227 – 250.

Konidaris, G., & Hayes, G. (2004). Estimating future reward in reinforcement learning animats using associative learning. *From Animals to Animats 8: Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior* (pp. 297–304).

Korf, R. (1990). Real-time heuristic search. *Artificial Intelligence*, *42*, 189–211.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. *Proceedings of the Twenty Second International Conference on Machine Learning (ICML 05)*.

Matarić, M. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots*, *4*, 73–83.

Moore, A., & Atkeson, C. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *13*, 103–130.

Ng, A., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: theory and application to reward shaping. *Proceedings of the 16th International Conference on Machine Learning* (pp. 278–287).

Perkins, S., & Hayes, G. (1996). Robot shaping – principles, methods and architectures. *Artificial Intelligence and Simulation of Behaviour 1996 – Workshop on Learning in Robots and Animals*.

Pickett, M., & Barto, A. (2002). Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. *Proceedings of the Nineteenth International Conference of Machine Learning (ICML 02)* (pp. 506–513).

Randløv, J., & Alstrøm, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. *Proceedings of the 15th International Conference on Machine Learning* (pp. 463–471).

Selfridge, O., Sutton, R. S., & Barto, A. G. (1985). Training and tracking in robotics. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 670–672).

Skinner, B. F. (1938). *The behavior of organisms: An experimental analysis*. New York: Appleton-Century-Crofts.

Stone, P., & Veloso, M. (2000). Layered learning. *Proceedings of the 11th European Conference on Machine Learning* (pp. 369–381). Barcelona, Spain: Springer, Berlin.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Taylor, M., Stone, P., & Liu, Y. (2005). Value functions for RL-based behavior transfer: a comparative study. *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*.

Thrun, S. (1992). *Efficient exploration in reinforcement learning* (Technical Report CS-92-102). Carnegie Mellon University.

Thrun, S., & Schwartz, A. (1995). Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems* (pp. 385–392). The MIT Press.

Van Roy, B. (1998). *Learning and value function approximation in complex decision processes*. Doctoral dissertation, Massachusetts Institute of Technology.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., & Thelen, E. (2000). Autonomous mental development by robots and animals. *Science*, *291*, 599–600.

Wiewiora, E. (2003). Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, *19*, 205–208.