Autonomous Shaping: Learning to Predict Reward for Novel States

George Konidaris Andrew Barto

Technical Report 2005-58

Autonomous Learning Laboratory
Computer Science Department
University of Massachusetts at Amherst

15th September 2005

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 predictor that estimates intermediate rewards, accelerating learning in later tasks that are related but distinct. Such agents can be trained on a series of relatively easy tasks in order to develop a more informative measure of reward that allows them to perform well on more difficult tasks, without requiring hand coded shaping functions. We use a rod positioning task to demonstrate that this approach significantly improves performance even after a very brief training period.

1 Introduction

Although reinforcement learning is well suited to many sequential decision problems, tasks characterised by delayed reward – where a long sequence of unrewarded actions are required to reach a reward state – remain difficult to solve efficiently, both in terms of finding initial solutions, and in terms of convergence towards 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, produces 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 problem should be able to learn their own shaping rewards, and thus learn to solve difficult problems quickly after a set of relatively easy training problems. 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 problem 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 simple rod positioning experiment in which our method significantly improves performance after even a brief period of training.

2 Background

2.1 Sequences of Goal-directed Tasks

In this paper we are concerned with a series of goal directed exploration problems (Koenig & Simmons, 1996). In each, the agent is in an environment (a set of states S with associated action set A) 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 convergence over the entire state space once the goal has been found, for which other methods exist (e.g., Thrun (1992)).

We require that the series of goal-directed problems solved by the agent are related in the sense that the agent is required to solve a sequence of variations on the same type of task. Therefore the agent experiences a series of Markov Decision Processes (MDPs) generated by the same underlying environment (or type of environment), so that each has the same set of actions (because the agent itself does not change) but a different set of states, transition probabilities and reward function. At each state the agent receives a sensation, some portion of which is used to build a descriptor that distinguishes Markov states in a particular task (generating problem-space), and some (possibly overlapping) portion of which is consistently present across the series of tasks and retains the same semantics (generating agent-space). This approach is distinct from that taken in prior reinforcement learning research on finding useful macro-actions across sequences of tasks (Bernstein, 1999; Pickett & Barto, 2002; Thrun & Schwartz, 1995) or building structured representations of a state space to speed up later learning in it (Mahadevan, 2005; Van Roy, 1998), where the tasks must be in the same state space but may have different reward functions. Taylor and Stone (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. However, this requires the explicit construction a transfer function between each pair of value functions, which implies a systematic and identifiable relationship between problem-spaces. An appropriate sequence of tasks in our research requires only that the agent-space semantics remain consistent, so each task may have its own completely distinct state space.

One simple example of such a series would be 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 series, and thus form the agent-space. The robot could eventually learn to use its temperature gauge as a heuristic measure of proximity to the source, and thereby be able to solve more difficult maps in less time, even though this is not in general sufficient to solve the problem by itself (because it is not Markov in problem-space).

Note that we require that the individual tasks are distinct, or at least not obviously identical, since otherwise we can simply transplant state or state-action values from one to the other.

2.2 Shaping

One popular method for speeding up reinforcement learning in general, and goal-directed exploration in particular, is *shaping* (Dorigo & Colombetti, 1998). Although this term has been applied to a variety of different methods, 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 and 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. 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 separate goal-directed problems.

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. (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 the use of shaped optimistic initial values (Sutton & Barto, 1998). The major drawback here is that this requires significant engineering effort and results in a loss of agent autonomy. However, an agent may be able to learn its own reward augmentation function from experience across several tasks, without having to have it specified in advance.

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 then 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 off with some pre-specified (possibly random or uniform) shaping function, and then refine it in several related but distinct problem instances over its lifetime. This is an instance of layered learning (Stone & Veloso, 2000), where learned shaping is used to make reinforcement learning more efficient.

Previous work on a realistically simulated robot learning to find a puck in a maze has shown that using training mazes to learn associations between reward and strong signals at reward states results in a significant improvement in total reward in a novel maze (Konidaris & Hayes, 2004). In this paper we present a more general mechanism, where the agent learns a heuristic from all of the states that it has visited.

3.1 A Framework for Learned Shaping Functions

We consider an agent solving a set of n problems, each with its own state space, denoted $S_1, ..., S_n$, and assume that learning is taking place over states (the state-action case is similar). We then view the ith state in task S_j (s_i^j) to consist of the following attributes:

$$< d_i^j, c_i^j, r_i^j, v_i^j >$$

where d_i^j is the problem-space state descriptor (sufficient to distinguish this state from the others in S_j, perhaps just i), 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). 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 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, so we define a function L that preserves value information between tasks, and acts as the agent's internal shaping reward. L estimates 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 Rodworld Experiment

In this section we empirically evaluate the potential benefits of a learned shaping function in a rodworld (Moore & Atkeson, 1993). A rodworld consists of a square workspace which 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 rotation) so that its tip touches the target while avoiding obstacles. An example 20x20 rodworld and solution path are shown in Figure 1.

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 either along the rod's axis or perpendicular to it, or a 10^o rotation in either direction. If a movement causes an agent 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 rodworld with five beacons, each of which emit a separate signal that drops off with the square of the distance from a strength of 1 at the beacon to 0 at a distance of 60 units. The tip of the rod has a sensor array which can detect the values of each of these signals at an adjacent state, forming the agent-space, and an element that learns L and uses it to predict reward given an array of five values representing these signals.

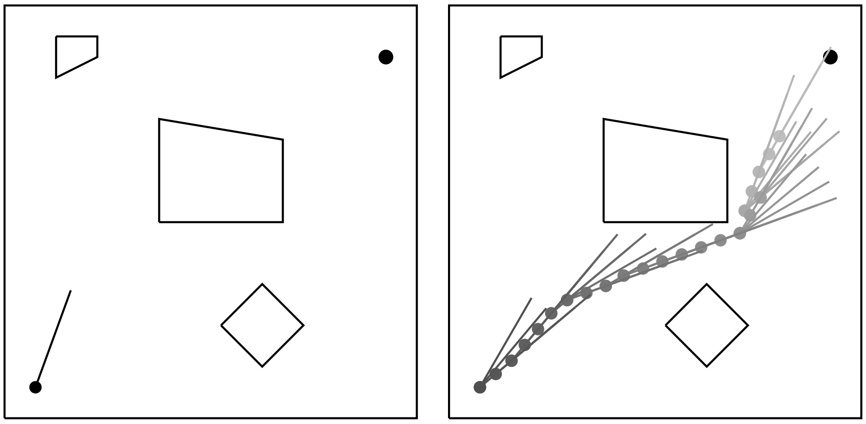

Figure 1: An 20x20 rodworld and one of its solution paths.

4.1 Experimental Structure

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

Each individual training episode places the agent in a small rodworld, randomly selected from a randomly generated set of 100 such worlds, 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 rodworld. 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 rodworlds that were either 10x10 (where it was given 100 episodes to converge in each training rodworld), or 15x15 (when it was given 150), and tested in a 40x40 rodworld. L was either a linear or quadratic estimator of reward given the five beacon signal levels as input, resulting in 6 and 11 parameters to be learned, respectively. All of these parameters were initially set to 0, and learning for L was accomplished using gradient descent (Mitchell, 1997) with $\alpha = 0.001$. We used two experiments with different beacon placement schemes.

4.2 Following a Homing Beacon

In the first experiment, we 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, can be well approximated even with a linear L. Figure 2 shows the 40x40 test rodworld used to evaluate the performance of each agent, and four sample 10x10 training rodworlds.

Figure 3 shows the number of steps (averaged over 50 runs) required to first reach the goal against the number of training episodes experienced by the agent for the four types of learned shaping elements (linear and quadratic L, and either 10x10 or 15x15 training worlds), as well as that required by an agent with a uniform initial value of 0 (agents with a uniform initial value of 500 performed similarly when first finding the goal). Note that there is just a single data point for the uniform initial value agents

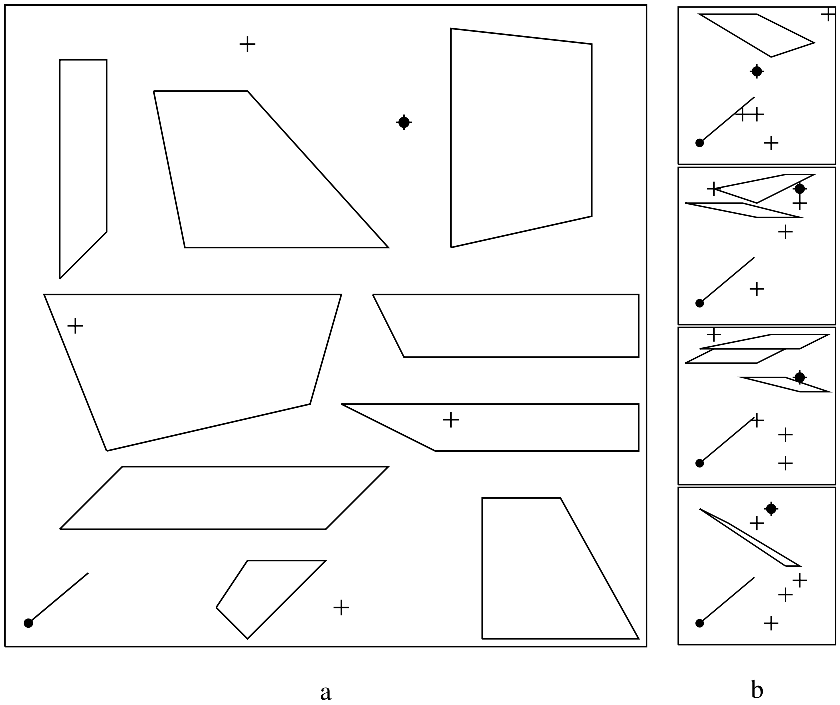

Figure 2: The homing experiment 40x40 test rodworld (a) and four sample 10x10 training rodworlds (b). Beacon locations are shown as crosses, and the goal is shown as a large dot. Note that one of the beacons is on the target in each world.

(in the upper left corner) because their performance does not vary with the number of training episodes.

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

Figure 4 shows the average (again over 50 runs) number of steps required to reach the goal as the agents repeat episodes in the test world, after having been allowed 20 training episodes (L is still never updated in the test world), as well as the number required by agents with value tables uniformly initialised to 0 and 500. This illustrates the overall learning performance of the agents over time on a single new task once they have been fully trained 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

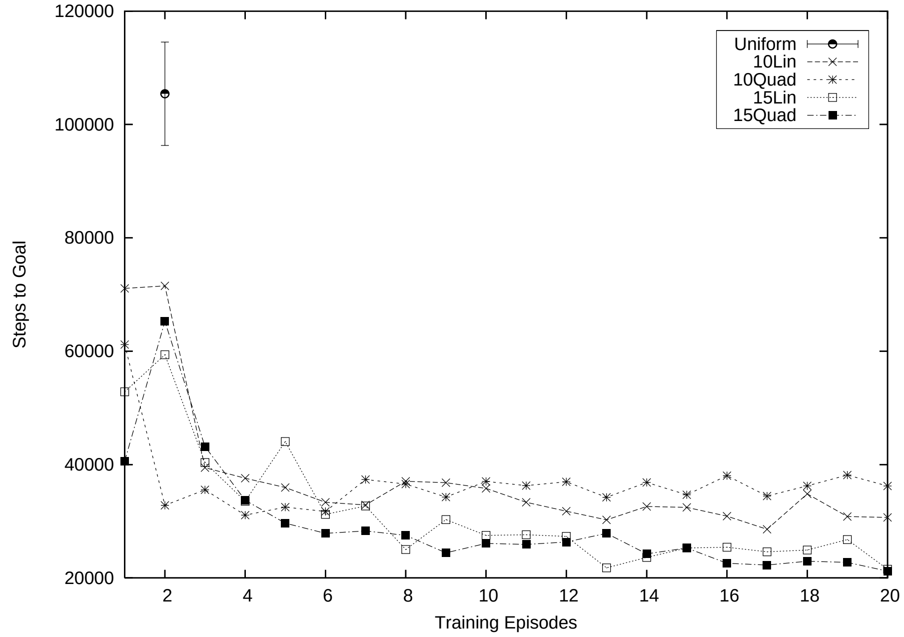

Figure 3: The average number of steps required to first reach the goal in the homing task.

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 in this particular domain works well because it is a deterministic one and a low initial value discourages extended exploration, but such behavior is not always desirable.

4.3 Finding the Center of a Beacon Triangle

In the second experiment, we arranged the first three beacons in a triangle at the edges of the rodworld, 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 be a harder function to learn. Figure 5 shows the 10x10 sample training worlds given in Figure 2 after modification for the triangle experiment. The test world was similarly modified.

Figure 6 shows the number of steps initially required to reach the goal for the triangle experiment, again showing that even a single training episode 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 world. 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

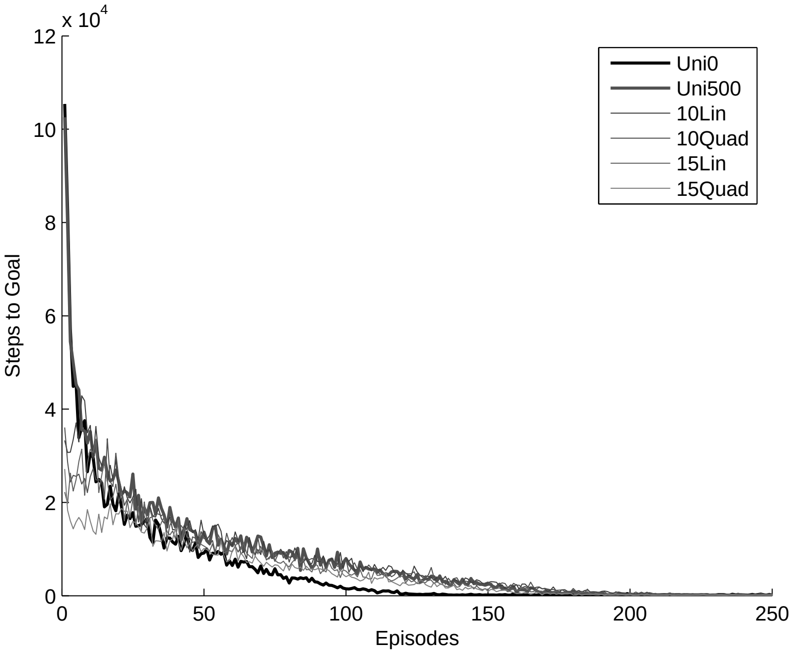

Figure 4: Steps to reward against episodes in the homing test world after 20 training episodes.

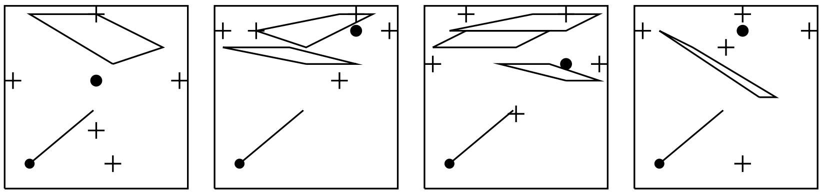

Figure 5: Sample 10x10 training worlds for the triangle experiment.

7 shows again that the initial few episodes of repeated learning in the test world 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 initialising its value table to 0 and one initialising it to 500.

4.4 Conclusions

The above two experiments show that an agent that can learn its own shaping rewards through training can find an initial solution to a problem in a space in which it has not trained much faster than an agent that uses uniform initial values, even after only a few training episodes. 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

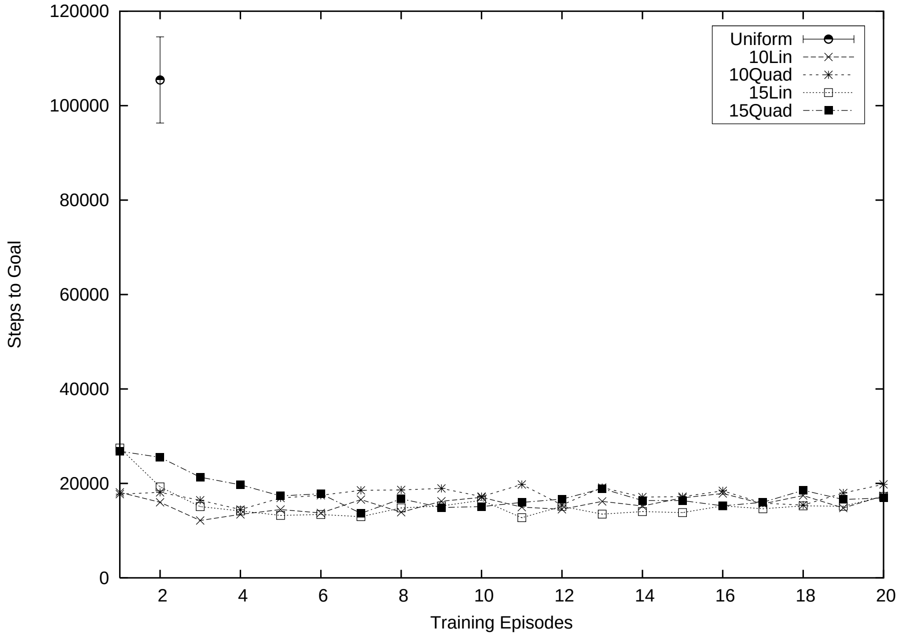

Figure 6: The average number of steps required to first reach the goal in the triangle task.

that its usefulness decreases as the reward signal 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 Discussion

The results given 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 and display more autonomy 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 and selecting an appropriate learning method for L creates a potentially difficult design problem. We expect that most of the difficulty will lie in choosing an appropriate c, which will then allow for the use of a relatively simple learning algorithm, which should in turn allow for rapid learning and short training times.

Another potential concern is the possibility that a maliciously chosen or unfortunate

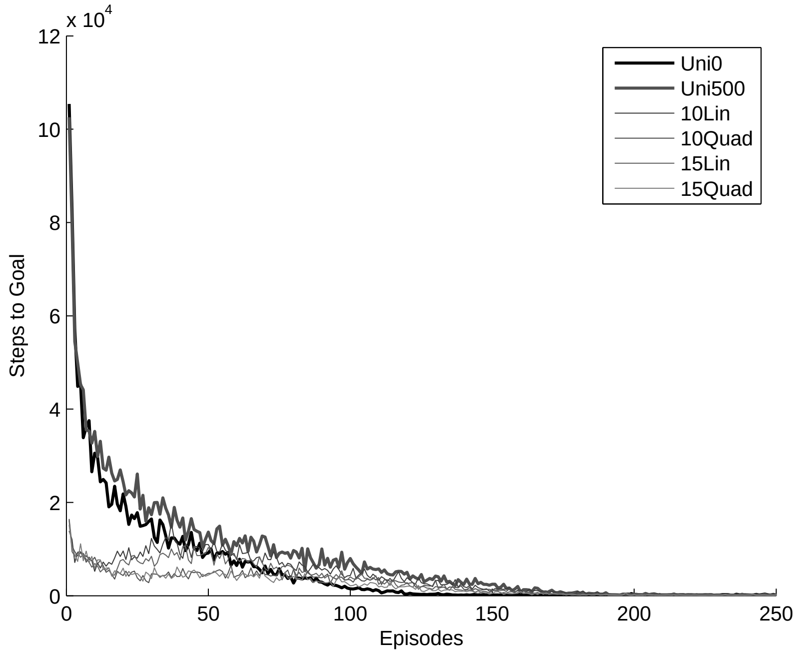

Figure 7: Steps to reward against episodes in the triangle test world after 20 training episodes.

set of training tasks could result in an agent that performs worse than one with no training, or becomes overly pessimistic. Alternatively, the learning algorithm chosen for L or the sensory patterns given to it might result in an agent which is completely unable to learn anything useful. However, we do not expect such an agent to do much worse than one without any shaping rewards at all.

5.1 Learned Shaping Rewards and Generalisation 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 generalisation, 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 generalisation 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 trained to represent values experienced at 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 generalises forwards, and not backwards: novel states are given initial values based on generalisation, but the values of previously encountered states are never disturbed. Therefore although learned shaping values do not generalise 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 generalisation.

Second, value function approximation usually only generalises within a single task. An approximated value function that is used to generalise over one MDP may not be applicable to another related but distinct MDP, because the semantics of each state descriptor may have changed (as a trivial example, consider two MDPs that are identical to each other but with different goal states), and because the size of the input to the approximator may have changed. A learned shaping function, by virtue of its split representation, can be used to generalise across a series 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 model of the underlying problem. Using sensor input as a state descriptor might facilitate generalisation, but it also often results in a state space that is both very large (thus necessitating generalisation) 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 generalisation.

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 discretisation of the map to generate a very small problem-space, and then use a separate learning element to generalise by learning to estimate novel state rewards from its remaining sensors.

5.2 Shaping Reward as a Search Heuristic

It is unlikely that in any useful scenario an agent will be able to learn an accurate measure of value in 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, rule-of-thumb measure for the distance between a particular node in the search space and the goal. This is combined with knowledge of how far that node is from the start node to obtain the total estimated distance from the start node, through the node under consideration, to the goal. During the search process, nodes which have not yet been considered are opened in order of increasing estimated total distance. Thus, guided search methods like A* use a heuristic to order the selection of unvisited nodes, whereas unguided search methods use some arbitrary ordering (e.g., a stack and a queue for depth- and breadth-first search, respectively) instead.

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 reaching the nodes open to it to determine where to go next, overwriting the heuristic values of visited nodes with that of the cost of reaching the nearby node with the smallest estimated total cost (heuristic plus transition cost). Since Real-Time Dynamic Programming (RTDP) is the stochastic generalisation of LRTA* (Barto et al., 1995), and shaping rewards act to order the selection of unvisited nodes and RTDP updates their values in exactly the same way, shaping rewards provide a heuristic initialisation of the value function in an embodied search problem when applied along with RTDP to goal-directed tasks. Therefore, agents solving embodied search problems that are able to learn their own shaping functions can effectively learn their own heuristics.

6 Conclusion

In this paper have introduced the use of learned shaping rewards in a sequences 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 which 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 episode.

Acknowledgements

We would like to thank Gillian Hayes, Colin Barringer, Sarah Osentoski and Özgür Şimşek for their useful comments. Andrew Barto was funded by NSF grant CCF 0432143 and a grant from DARPA's IPTO program.

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.

Mitchell, T. (1997). *Machine learning*. McGraw-Hill.

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).

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. (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.