Evolved Intrinsic Reward Functions for Reinforcement Learning

Scott Niekum
University of Massachusetts Amherst
140 Governors Drive Amherst, MA 01003
Phone: (413)-545-1596 Email: sniekum@cs.umass.edu

The reinforcement learning (RL) paradigm typically assumes a given reward function that is part of the problem being solved by the agent. However, in animals, all reward signals are generated internally, rather than being received directly from the environment. Furthermore, animals have evolved motivational systems that facilitate learning by rewarding activities that often bear a distal relationship to the animal's ultimate goals. Such *intrinsic motivation* can cause an agent to explore and learn in the absence of external rewards, possibly improving its performance over a set of problems.

This raises an important question: given a problem, can we find an alternate reward function that the agent can use to learn faster than it would in the original setting? Singh, Lewis, and Barto (2009) demonstrated that an entire spectrum of reward functions exist that outperform the natural task-specific reward function, even for simple problems. These alternate reward functions were found through a brute force search over a small, discretized set of possible scalar reward functions defined on a subset of the state variables from the original problem. Their objective was to examine questions about the nature and source of reward functions, and not to suggest that a search over reward-function space was a potentially useful methodology for solving difficult problems. Nonetheless, their results suggest that such a search may indeed be beneficial if carried out efficiently.

Other work has been done exploring alternate reward functions in various contexts. Ng, Harada, and Russell (1999) explored shaping adjustments to the task-based reward function that would not disturb the optimal policy. By contrast, we aim to find reward functions that define a new problem, which has a solution closely related to the original problem, but is easier to solve. Shaping rewards do not redefine the problem and can be 'learned away'; they were later shown to be equivalent to using initial value functions (Wiewiora 2003). Ackley and Littman (1991) and Uchibe and Doya (2008) experimented with using evolutionary methods to search for reward functions, but analyzed their results only in the context of evolutionary biology in the former case, and only explored shaping rewards in the latter.

We propose applying genetic programming methods—a class of efficient, general search procedures that search over the space of *programs*—to search for reward functions. These reward functions operate over the entire state space of a reinforcement learning problem and, if successful, will be able to quickly and automatically identify relevant variables and features of the problem. This will allow the agent to outperform an agent that uses the obvious task-based reward function. The use of genetic programming methods may alleviate the difficulty of scaling reward function search and provide a natural way to search through a very expressive space of such functions. For this task, we selected PushGP, a stack-based evolutionary programming system that has previously produced human-competitive results in multiple domains and has the capability to search over the space of all computable functions (Spector, Klein, and Keijzer 2005).

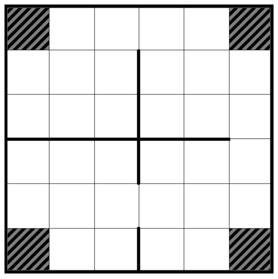
Figure 1: The Hungry-Thirsty domain. Thick lines are walls, striped squares denote possible food or water sites.

This methodology is evaluated using the Hungry-Thirsty domain, proposed by Singh et al. (2009) and shown in Figure 1. In each instance of this domain, food and water locations are picked randomly from two of the four corners and held fixed for the agent's lifetime. The agent's objective is to eat as much food as possible, but the agent cannot eat when it is thirsty. Drinking makes the agent not-thirsty for one time step, but at each time step thereafter, the agent becomes thirsty again with probability 0.1. When the agent successfully eats, it becomes not-hungry for one time step. For each time step that the agent is not-hungry, it increments its fitness score, a simple surrogate for evolutionary fitness. We intentionally chose this small domain to simplify the analysis of our methodology and the evolved reward functions. In

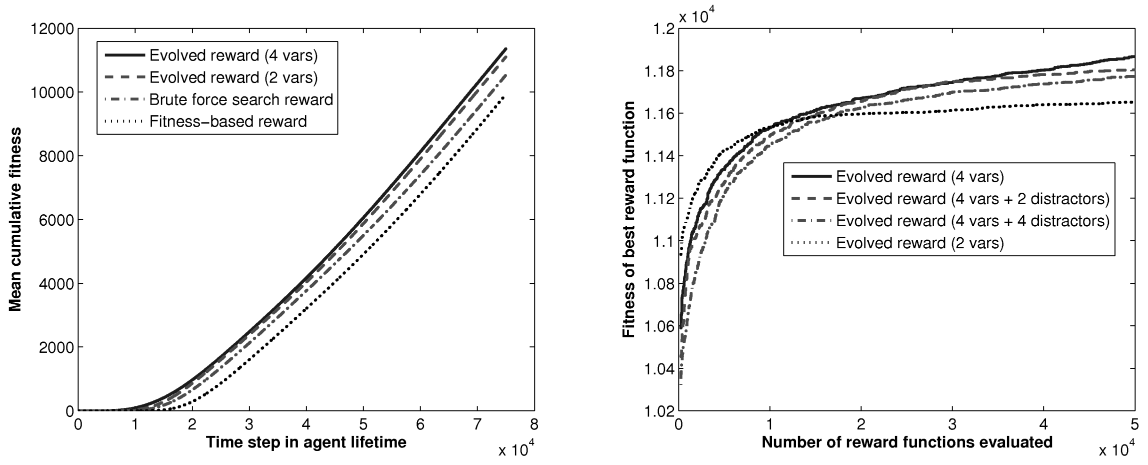

Figure 2: Agent fitness (left) and evolutionary progress (right) over a distribution of environments.

our experiments, generations of agents are evaluated based on their fitness across a distribution of environments within this domain. Thus, a reward function emerges from the evolutionary process that maximizes an agent's fitness across this distribution of environments, capturing the most salient common features.

Early results in the Hungry-Thirsty domain suggest that PushGP is a promising system for efficiently searching for reward functions. The left panel of Figure 2 compares the cumulative fitness over an agent's lifetime when using the fitness-based reward, the best brute force search reward from Singh et al. (2009), and rewards evolved using PushGP. In the two variable case, the evolved reward function was given as input the hunger and thirst status of the agent, the same two state variables that the brute force method was given. Here, the evolved function performs better; the evolutionary search did not have to search over a discretized space, and thereby found a more refined solution. In the four variable case, we provided the reward function with two additional state variables, the agent's X and Y position, and the evolutionary process produced a slightly better reward function. At the very least, this shows that evolution can extract salient features of a problem, even when searching over a very large space of functions, affording better performance than the simple brute force method.

The right panel of Figure 2 shows the fitness of the best reward function evaluated so far as a function of time during the evolutionary process. Despite the fact that adding additional relevant variables significantly increases the size of the search space, we see that the four variable curve only lags behind the two variable curve until around the tenth generation, and then shows better performance thereafter. In the other two cases, we added additional 'distractor' variables (two and four, respectively) that were uniformly distributed integers between 0 and 4. It appears that evolutionary search quickly learned to ignore these irrelevant variables, as evolutionary progress slowed relatively little with their addition. This suggests that evolutionary methods such as this may generalize well to larger, high-dimensional problems.

Alternate reward functions can improve the learning speed and fitness of agents in a domain, but they may also have more powerful uses in the future. As mentioned earlier, a properly evolved reward function captures the common features across environments that the agent faces. Thus, such a reward function may be able to aid an agent in autonomous skill identification and learning, allowing the agent to hierarchically break down and solve much larger problems than are currently feasible; it may also be able to capture the similarities across a collection of tasks and thus facilitate transfer.

Acknowledgements

This work was done under the supervision of professors Andrew Barto and Lee Spector.

References

Ackley, D., and Littman, M. 1991. Interactions between learning and evolution. In *Artificial Life II*. SFI Studies in the Sciences of Complexity.

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

Singh, S.; Lewis, R.; and Barto, A. 2009. Where do rewards come from? In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, 2601–2606.

Spector, L.; Klein, J.; and Keijzer, M. 2005. The push3 execution stack and the evolution of control. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, 1689–1696.

Uchibe, E., and Doya, K. 2008. Finding intrinsic rewards by embodied evolution and constrained reinforcement learning. *Neural Networks* 21(10):1447–1455.

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