

---

# Exploiting Training Regimens to Improve Learning

---

Peng Zang

Charles Isbell

Andrea Thomaz

Georgia Institute of Technology, Atlanta GA, 30308  
{pengzang,isbell,athomaz}@cc.gatech.edu

## 1. Introduction

Although theoretically elegant and powerful, approaches to reinforcement learning (RL) are sometimes criticized as unscalable to real-world environments. Here, we recognize that in many such environments, agents will be learning in close proximity to humans and can take advantage of human input to improve scalability and performance.

In particular, we define the notion of a *training regimen* for reinforcement learning, and seek to answer several questions: Does providing a training regimen improve learning? Are humans good at providing training regimens? Can we automatically generate training regimens?

## 2. Training regimens

While space does not permit a complete discussion, there are a variety of methods for leveraging humans for teaching machine agents. One common method is demonstration, where solutions to example problems are shown to the learner. Other methods have humans decompose problems so the learner need only solve a series of small problems (e.g., hierarchical decompositions (Dietterich, 1998), goal-based decomposition (Karlsson, 1997), or explicit training of skills (Stanley et al., 2005)). Still other methods, like reward shaping (Dorigo & Colombetti, 1994) leverage human input to guide agent exploration.

By contrast, we leverage human interaction by asking for a series of increasingly difficult problem instances, a *training regimen*. Here, an *instance* refers to solving an RL problem given a specific start and goal state pair, such as the starting and ending squares in a maze. Thus, providing a training regimen is much like providing demonstrations, but without the burden of having to provide solutions. As we will see, training regimens also provide decomposition; however, unlike previous approaches that are designed to exploit particular characteristics of certain domains and require specialized learners, regimens do so in a general fashion – providing wider applicability. A training regimen can also be seen as an alternative way of guiding exploration. Unlike reward shaping where one authors a potential-based shaping function (Ng et al., 1999), it guides exploration directly by giving samples of where the learner should focus.

Focus is important. Algorithms typically assume the goal is to solve all instances (to find a policy over all states), and that all instances are equally important. These assumptions are often false. For example, we do not care about solving all chess board positions, just those reachable. Thus, a properly-tailored regimen increases learning efficiency by maximizing generalization while minimizing the number of instances the learner must see and solve. For example one may provide a higher density of examples in complex or important regions and fewer examples elsewhere. In an interactive setting, the instance chosen can be tailored to learner performance. For example, it can be used to highlight errors in the learned function.

In addition to focus, providing instances benefits the learner by alleviating the need to sample. This is significant because the target instance distribution may be difficult to express. The domain may also have constraints that make sampling valid states expensive.

Finally, the ordering of instances guides the learner. By ordering instances such that more difficult ones build upon simpler ones, we can save the learner significant work. Solving each new problem instance should then only require incremental effort.

Our approach is to order instances by distance.<sup>1</sup> The insight is this: if we have learned to solve short (or “easy”) problem instances, when we come upon a more distant (or “harder”) one, we will not need to solve it entirely. We need only solve how to get close enough to the goal such that our prior solutions can be used. The effect is a decomposition into a series of short problems. This approach improves learning efficiency by relying on stitching value functions together rather than on specifics of any particular learning algorithm used.

## 3. Automation of training regimens

A training regimen provides some guidance that cannot be mechanized. For example, the specification of the target instance distribution cannot be automated because it is part of the problem definition; nevertheless, we can automate

---

<sup>1</sup>Without loss of generality, we assume a negative reward function, to speak more intuitively in terms of cost and distance.

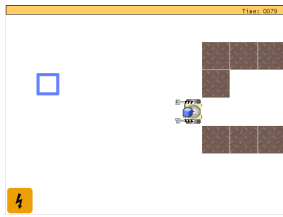


Figure 1: Walle domain

other portions.

First, we can partially automate the training regimen using an Extended Random Walk (ERW) distribution. An ERW works much like a normal random walk except it takes advantage of prior solutions where available to choose a distribution over actions that take it away from the start state. The effectiveness of this strategy depends on how closely the ERW distribution matches the target distribution.

A second strategy is training regimen augmentation. It augments an existing regimen by generating additional instances based on random walk perturbations and other transformations. Again, its effectiveness at magnifying the impact of a given regimen depends on how well the additional instances match the desired distribution. If they match well, they increase the number of training instances available to the system; otherwise, they waste computation by forcing the system to solve unimportant instances.

## 4. Experiments

We ran experiments comparing three algorithms. *BASE* is a simple approximate value iteration (AVI) algorithm, providing baseline performance. *HTRA* is AVI with interactive, human-provided training regimens and regimen augmentation. Finally, *ATR* is AVI using the automated training regimen strategy. In all cases a regression tree was used to represent the value function.

We used the Walle domain for our experiments (see Figure 1). In this domain, the agent, Walle, is tasked with finding a trash cube, moving it into the disposal, and docking with a charging station. The state space is the combination of all object locations and whether or not Walle is carrying the trash cube. Reward is uniformly -1.

In experiments, *BASE* performed poorly, unable to converge upon a good policy. On average, it successfully solves fewer than 10 percent of randomly generated instances. By contrast, *HTRA* was able to successfully solve Walle to near perfect accuracy. *ATR* was also unable to solve the Walle domain, learning a policy that is only correct when the trash cube is already in the disposal; that is, *ATR* only learns how to go to the charging station. This strange behavior is a result of the fact that ERWs are unlikely to move the trash cube. Thus, *ATR* never generates

instances where it must learn how to move the trash cube around, learning only how to move Walle. In other experiments, we tested *ATR* on simple maze domains where random walks are more likely to generate problem instances matching the target distribution. In those experiments, *ATR* performed well and was able to reach near perfect accuracy.

We also ran user studies (seven participants) with *HTRA* to see if non-expert humans are good at providing training regimens. Results show that they are: each participant was able to train Walle to perform nearly perfect on the tasks they trained it to perform. Interestingly, because we did not specify a target task, each participant trained Walle for something slightly different. In other words, when a participant trained Walle, s/he did not cover the whole space, but focused on some aspect of the domain, “personalizing” Walle. Taking the union all user given instances as the human “benchmark” set, we also looked at how well each “personalized” Walle performed on this larger set. On average, a personalized Walle was still able to solve about two thirds of those instances. This suggests that despite differences in how each human trains Walle, they have significant overlap. Thus, even without explicit direction, humans may be fairly good at providing a reasonable distribution of problem instances for an RL agent.

## 5. Conclusion

We explored the training regimen paradigm for RL. We showed how it could be used to effect a decomposition, and explored two automation strategies. Experiments show that training regimens are effective at improving learning, that humans are able to provide effective (yet personalized) training regimens, and that automation can work well but only when it closely matches the target distribution.

## References

- Dietterich, T. G. (1998). The MAXQ method for hierarchical reinforcement learning. *ICML*. Morgan Kaufmann.
- Dorigo, M., & Colombetti, M. (1994). Robot shaping: developing autonomous agents through learning. *Artif. Intell.*, 71, 321–370.
- Karlsson, J. (1997). *Learning to solve multiple goals*. Doctoral dissertation, Rochester, NY, USA.
- Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *ICML* (pp. 278–287). Morgan Kaufmann.
- Stanley, K. O., Bryant, B. D., & Miikkulainen, R. (2005). Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, 9, 653–668.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.