What does Shaping Mean for Computational Reinforcement Learning?

Tom Erez and William D. Smart
Dept. of Computer Science and Engineering
Washington University in St. Louis
Email: {etom,wds}@cse.wustl.edu

Abstract—**This paper considers the role of shaping in applications of reinforcement learning, and proposes a formulation of shaping as a homotopy-continuation method. By considering reinforcement learning tasks as elements in an abstracted task space, we conceptualize shaping as a trajectory in task space, leading from simple tasks to harder ones. The solution of earlier, simpler tasks serves to initialize and facilitate the solution of later, harder tasks. We list the different ways reinforcement learning tasks may be modified, and review cases where continuation methods were employed (most of which were originally presented outside the context of shaping). We contrast our proposed view with previous work on computational shaping, and argue against the often-held view that equates shaping with a rich reward scheme. We conclude by discussing a proposed research agenda for the computational study of shaping in the context of reinforcement learning.**

I. Introduction

Behaviorist psychology explores the mechanisms of learning through reward and punishment, and provides inspiration and motivation to the machine learning paradigm of computational reinforcement learning (RL). As may be expected, the computational rendering of RL differs from the psychological theory of reinforcement learning in many ways, but preserves an important essence: success is achieved by maximizing future rewards.

Shaping is another notion, central to behaviorist psychology, that has seen several computational renderings. In the discipline of psychology, the word *shaping* refers to a conditioning procedure used to train subjects to perform tasks that are too difficult to learn directly. Through shaping, the trainer induces the desired behavior by means of *differential reinforcement of successive approximations* to that behavior. This means that the subject is brought to perform the *task of ultimate interest* by mastering a series of related tasks. The successful learning of one task in the *shaping sequence* guides the subject's behavior in the following task, and facilitates the learning of later tasks in the sequence. In essence, shaping is a developmental approach, where the subject is allowed to refine its skills at it masters the different tasks in the sequence as they become progressively harder.

In most modern shaping protocols, it is the reward/punishment scheme that changes from one iteration to the next. At first, the learning agent is rewarded for meeting a crafted subgoal, which in itself does not fulfill the desired behavior, but serves as a simpler approximation to it. Once this subgoal is mastered, the reward scheme changes, and the next task in the sequence challenges the learning agent to produce a better approximation to the desired behavior. This sequence of rewarded subgoal behaviors provides a behavioral scaffold, eventually bringing the agent to perform the task of ultimate interest.

However, reward shaping was not the first shaping technique to be considered, chronologically speaking. In the early days of behaviorist psychology, it was the experimental setting, and not the reward scheme, that was altered by the trainer. As Peterson points out [1], the first practical demonstration of the power of reward shaping (as part of a research aiming to use pigeons for missile navigation and control) took place five years *after* the publication of B.F. Skinner's seminal book "The Behavior of Organisms" in 1938 [2]. A closer look at the notion of shaping reveals that, despite the contemporary dominance of reward shaping, the concept itself extends well beyond merely choosing a rich reward scheme, and applies to any supervised modification of the learning task (see section IV). In this paper we wish to explore the different ways in which we can find a solution to a task that is too hard to master independently, by crafting a sequence of related, easier tasks. As the next section shows, modifying the reward function, or the experimental environment, are but two examples of employing shaping in RL.

The notion of successive approximations that converge to a desired outcome is a staple of computer science, and therefore there is reason to hope that one could identify a natural way to render shaping in computational terms. In this paper we propose that shaping can be associated with the class of algorithms known as *homotopy-continuation methods* [3]. We explore the different ways of shaping by bringing together a diverse body of works from the RL literature, and showing how our computational interpretation of shaping easily applies to all of them. The essence of shaping we wish to render in computational terms is that of a *supervised, iterative process*, whereby the learned task is repeatedly modified in some meaningful way by an external trainer, so as to eventually bring the learning agent to perform the behavior of ultimate interest.

In order to anchor the technical terms used in the rest of the paper, the next section is dedicated to a quick review of computational reinforcement learning. In section III we present an exhaustive list of the different ways shaping can be rendered computationally within the RL paradigm. For each such category we discuss what is being shaped and present prior work which demonstrated the merit of this shaping technique. Finally, we conclude with a broader discussion, and contrast our perspective with previous realizations of computational shaping.

II. Computational RL Terminology

This exposition offers a quick review of the computational theory of reinforcement learning; the reader familiar with RL vocabulary may safely skip ahead to section III without losing sight of the main argument.

In RL, we render the problem of behaving optimally as an optimisation problem: an *agent* interacts with an *environment*, and this interaction yields an instantaneous scalar *reward*, serving as a measure of performance. The agent's goal is to behave in a way that maximizes future rewards.[1] For example, the task of balancing a pole on one's hand could be described by rewarding the agent for keeping the pole perpendicular to the ground at its unstable equilibrium.

All the properties of the agent and the environment that may change through time are captured by the system's *state*.[2] The agent realizes its agency by choosing an *action* at every time step, which affects the next state. The current state of the system, together with the chosen action (and, of course, some domain-specific, fixed parameters), allow two key evaluations: first, the current state and action determine the instantaneous *reward*; second, the current state and action determine the system's next state. In the above-mentioned example, the system's state includes the pole's angle and angular velocity, as well as the hand position and velocity. The agent acts by moving its hand, and hence the pole's tip.

The computational representation of the progression of states in RL can take several forms: in the simplest case, time is discrete. In this case, an action is chosen every time step, and the next state is determined by the current state and the chosen action. If time is continuous, the mathematical formulation becomes more complicated, as both state and action are functions of time. If the dynamical system is stochastic, the current state and action determine the probability distribution of the next state.

In some cases, the state space is finite and discrete. For example, in the game of chess every piece can only be in one of 64 positions. In the case where mechanical systems are studied, the state space is often abstracted as a continuous vector field. The same holds for the *action space*.

The agent's behavior is described by the *policy function*, a mapping from state to action that describes the agent's action at a given state and time, perhaps probabilistically. Since the policy determines the agent's behavior, it is the object for learning and optimization. Broadly stated, the agent's goal is to find a policy that would maximize future rewards, or their expectations. These rewards may be summed (in the discrete case) or integrated (in the continuous case) over a finite or infinite future time horizon, perhaps using some form of discounting rewards in the far future.

One of the fundamental approaches in RL involves the *value function*. Given a fixed policy, the value of a state is the total sum of (expected) future rewards. Therefore, the agent's goal can be formally stated as finding the policy that maximizes the value of all states. Intuitively, the value of a state indicates how desirable this state is, based not on the greedy consideration of instantaneous reward, but in full consideration of future rewards. For example, let us consider two states of the pole-balancing domain with the pole being in the same non-perpendicular angle. In one, the pole's angular velocity is directed towards the balancing point; in the other, it is directed away from it. If the instantaneous reward depends only on the angle of the pole from the above example, both states would provide the same reward. Yet, the optimal value of these two states is different, the first state being clearly better than the second.

As interaction (and learning) commences, the agent is placed in an initial state (perhaps chosen probabilistically) and is equipped with an initial, suboptimal policy (perhaps a stochastic one). The agent learns through a particular interaction history (sequences of state-action-reward triplets), with the goal of devising a policy that would lead it through the most favorable sequences of states, and allow it to collect maximum rewards along the way.

A key concept in computational models is that of computational *representation*. In many cases, the same task can be described using different terms; for example, when describing a body pose of a limbed robot, we may specify the joint angles together with the position of the center-of-mass, or the position and orientation of each limb. Even if different representations of the same domain are mathematically equivalent, they may generate different computational constraints and opportunities. For example, consider the case where the reward is a function of the position of the center-of-mass. In the first case, this dependency would be clearly manifested through simple correlation, while in the other case this dependency would be more hidden. The choice of representation is known to be a subtle issue in all domains of machine learning, and is often the locus of intervention where the researcher can input some prior domain knowledge to facilitate solving the task.

For further reading on computational reinforcement learning we recommend the canonical book by Sutton and Barto [4], as well as the widely-cited survey paper by Kaelbling, Littman and Moore [5].

III. The different ways of shaping

As stated before, we view shaping as an iterative process, composed of a sequence of related tasks. At every iteration, the final solution to the previous task informs the initial solution in the present task. Delving deeper into mathematical abstraction,

[1] An alternative definition, of *minimizing* future *costs*, is ubiquitous in the Optimal Control literature, but since it is mathematically equivalent, we make the arbitrary choice of using the reward-based terminology.

[2] In the *fully-observable* case the agent has full knowledge of the current state, while in the *partially-observable* case it may have access only to a noisy and/or partial observation of the state. For clarity, in the remainder of this exposition we will refer only to the fully-observable case.

we can think of a shaping sequence as a *homotopy* in the space of all RL tasks, a path leading from an initial, simple task, to the task of ultimate interest, through a sequence of intermediate tasks. This is a powerful metaphor, because it allows us to associate between several distinct algorithms as different homotopy-paths that traverse different dimensions of task space. Since an RL task is characterized by several aspects (dynamical function, reward function, initial state, and so forth), the space of all tasks is high-dimensional, and there can be many different paths connecting any two tasks. This means that when designing a shaping protocol, many aspects of the task can be modified at each step.

In the following list we enumerate the different dimensions along which tasks can be altered. For most categories, we were able to find prior RL work which explored that particular rendering of the homotopic principle, even if most works considered didn't refer to their proposed technique by the term 'shaping'. While some categories have direct counterparts in the world of behaviorist psychology, others come about due to the way an RL task is represented computationally. It is important to note that the works we present as examples do not comprise a comprehensive list of all papers ever to employ that kind of shaping, and are mentioned merely for illustration purposes.

1) **Modifying the reward function**: This category includes cases where the reward function changes between iterations. For example, in order to reward the agent for achieving a sequence of subgoals that eventually converge to the behavior of ultimate interest. As mentioned before, this is the most straightforward rendering of behaviorist shaping in its common form. There have been several RL works that take this approach. One such example is by Gullapalli [6], where a simulated robot arm is trained to press a key through successive approximations of the final desired behavior.
   One important variant of this category is *chaining*, where subgoals follow each other sequentially to compose the behavior of ultimate interest. Singh [7] discusses learning compositional tasks made of elemental tasks in a simulated gridworld. Another interesting example, focusing on motor learning in non-symbolic domains, is Via-Point Representation [8], [9]. An impressive illustration of this approach is a real robotic 3-link chain of rigid bodies, trained to stand up by following a sequence of motions: first, it is trained to assume a sitting position (which is statically stable), then it is trained to reach a crouched position from the sitting position, and only then it is trained to reach a standing position from the crouched position.
2) **Modifying the dynamics**: this category includes cases where physical properties of the system change between iterations. These might be physical properties of the environment or of the agent itself. An example for the first case (changing the environment) is the gradual elevation of a lever, in order to bring a rat to stand on its hind legs. An example of the second case (changing the agent) is the way people train to walk on stilts: first, one learns to walk stably on short stilts, and then the length of the stilts is gradually extended. As mentioned before, Skinner's original experiments in the 1930s were along these lines, and the reward shaping is a later development from the 1940s.
   An early example for using this approach in machine learning is the early work of Selfridge, Sutton and Barto [10], where optimal control of the pole-balancing task is achieved by gradually modifying the task parameters from easier to harder variants (e.g. starting with a heavy pole and switching to a lighter one, or starting with a long track and gradually reducing its length). Randløv [11] proved that a shaping sequence that convergences to a final task entails convergence of the solution sequence to the final optimal solution for the case of global Q-function approximation over finite Markov Decision Processes.
   A different perspective on modifying the dynamics comes about when learning stochastic dynamics.[3] Assuming that the dynamics are strongly influenced by noise may lead to different responses in different algorithms: the lack of confidence in the generality of the particular history encountered may lead to a smoother estimation of the value [12], or, in case noise is modeled as an adversary in a noncooperative differential game [13], [14], increased noise means a stronger adversary. In the first case, it has been shown [15] that the smoothing effect of noise may serve as antidote against overfitting the value function, and by gradually *reducing* the noise, the exact value function may emerge. In the second case, it is considered better [16] to first learn the task assuming less noise (i.e. a weaker adversary), and gradually *increase* its effect.
3) **Modifying internal parameters**: Many algorithms are "parametric", in the sense that they require a-priori setting of some parameters (e.g. number of neurons in an artificial neural network), independently on the actual task data. For such algorithms, one can often find some heuristics to determine an appropriate value for such a parameter. However, in some cases greater efficiency can be achieved if a *schedule* is set to gradually alter the value of such a parameter. A classic case for this category is *simulated annealing*, where a "temperature" parameter is gradually decreased, allowing the system to overcome local extrema. In RL, scheduling the learning rate is considered a standard approach [17]. A more interesting example is the work of Fasel [18] which presents an additive boosting mechanism whereby new elements are gradually added to the parametrization of the likelihood model.

[3]This modeling choice often serves in cases where there is discrepancy between the training model and the runtime environment, for example in model-based learning for real, physical robots. In these cases, the unmodeled and mismodeled dynamical effects are regarded as noise.

4) **Modifying the initial state**: In many goal-based tasks, the optimal policy is simpler to learn when the initial state is close to the goal, and becomes progressively harder for states farther from the goal. This is the case because the effective delay between an action and its consequence is short. The optimal solution, then, can be learned by "growing" the solution out from the goal state towards the rest of the state space volume, or any portion of it. Once the optimal policy is learned in the immediate vicinity of the goal, another initial state, slightly farther from the goal, can be learned more easily - the agent can reach the goal by first reaching one of the previously-learned states, and following the previously-learned policy from there. Such a controlled interaction reduces the risk of prolonged exploration and aimless wandering. One algorithm that employed such an approach for value function approximation in deterministic MDP's was the GROW SUPPORT by Boyan and Moore [19].
5) **Modifying the action space**: Since the number of possible policies grows exponentially with the number of possible actions, limiting the set of possible actions generally makes for an easier optimization problem. Such a modification is often coupled with modifying the state space, and as such it is related to the second category above.
   One example from developmental psychology for this class of shaping is the way infants lock some joints in their arm when learning to do a certain motion, so as to temporarily limit the number of degrees of freedom of their arm, and make the task more tractable [20], [21]. In RL, Marthi [22] discusses an algorithm called *automatic shaping*, whereby both states and actions are clustered to form an *abstracted* task. The value function of the abstracted task then serves as a rich reward function for the original task.
6) **Extending the time horizon**: Since the formulation of the RL domain involves the integration of the reward into the future along some time horizon, we can consider a homotopy that gradually extends the time-horizon parameter (or, equivalently, decrease the discounting factor). This idea lies at the heart of the VALUE ITERATION algorithm [23], where the optimal value function is estimated using a series of evaluations of extending lengths. First, the value function for a single step through the dynamical system is evaluated; then, it is used in an evaluation of the value function for a two-step interaction; and so forth. Another application of this idea was presented in [15], where the pole-balancing task was solved by gradually extending the time horizon and re-approximating the value function, taking the previous value function as an initial guess.

IV. RELATED WORK

Our central claim in this paper is that shaping is an iterative, supervised process, and that this property should be preserved when this notion is rendered computationally. This view is under dispute in the RL community, and some of the most famous works on computational shaping consider only the static, one-task case of a rich reward function: Randløv and Alstrøm's work on bicycle riding [24] and Ng's work on autonomous helicopter control [25] both used the term 'shaping' to refer to a static, rich-reward problem. Optimal design of reward functions has been studied before (e.g. Matarić [26]), and it is clear that a well-designed reward function may facilitate learning, promote faster convergence, and prevent aimless wandering. Also, as pointed out by Laud [27], if the optimal value can be provided as a reward, the RL task successfully collapses to greedy action selection. However, we suggest that the notion of shaping goes well beyond merely using a rich reward scheme.

On the other hand, many other papers highlighted the iterative nature of shaping: Kaelbling, Littman and Moore [5], and Sutton and Barto [4] both consider shaping as an iterative scheme, independently from considerations of reward function design. Asada [28] referred to such iterative schemes as *learning from easy missions* (LEM).

Several other works are concerned with computational shaping. Dorigo and Colombetti, in their book "Robot Shaping" [29], suggested to import behaviorist concepts and methodology into RL, and discussed a model for automatic training of an RL agent. In the scheme they consider, the automatic trainer has an abstracted specification of the task, and it automatically rewards the agent whenever its behavior is a better approximation of the desired behavior. This work is often cited as a prime example of computational shaping, but its impact is nonetheless limited we could find no papers that actually employ their proposed method of employing an automatic trainer.

Kaplan et al. [30] discuss the case of human-robot interaction, where the reward is provided by a human observer/trainer. The trainers may alter their criteria for providing reward, so as to shape the desired behavior through successive approximations.

Konidaris and Barto [31] consider a case they title *autonomous shaping*, where the tasks in the shaping sequence share similar dynamics and reward scheme, but obstacle and goal position is altered. There is no deliberate ordering of the tasks from easy to hard, and the learning agent is meant to generalize from the different instances, and create a robust representation of the solution.

Finally, in Tesauro [32], learning took place through self-play between two copies of the same agent. This guaranteed that at every stage, the agent faced an opponent of optimal strength — not too strong, and not too weak. That work has been cited as an example for shaping, but we claim that it is better described as an adaptive exploration of task space, and as such, falls outside the realm of strictly supervised shaping.

V. RESEARCH AGENDA

One possible research direction is the identification of certain shaping invariances. Ng's method of potential function-based enriching of the reward function [25] was shown by Wiewiora [33] to be equivalent to a modification of the initial state distribution. While both these works consider the static, one-task case, it might be the case that such invariant relations exist between some of the categories discussed in the previous section.

Issues of designing shaping protocols have implications for representational considerations. In order for shaping to work, the solution to one domain has to inform the initial approach to the next domain in the sequence. This implies that the representation used should facilitate knowledge transfer between subsequent tasks. This subject is the focus of contemporary study [34].

We believe that this paper is the first to discuss the link between shaping and homotopy-continuation methods. The metaphor of a shaping sequence as a trajectory in the high-dimensional space of all RL tasks may allow the introduction of advanced continuation methods to the study of RL learning and shaping. Currently, the biggest gap between shaping and homotopy-continuation methods is that continuation methods often relate to the continuous nature of the homotopy trajectory, while all the shaping methods discussed above jump discretely between the tasks in the shaping sequence. We hope to bridge this gap in future work.

In conclusion, we believe that some sort of a continuation method (such as shaping) is *imperative* to tackle any RL domain of real-life difficulty, of high dimensionality and impoverished reward. Many of the interesting RL tasks are impossible to solve directly, and shaping may well be the only viable way to make them tractable.

ACKNOWLEDGMENT

The authors would like to thank Michael Dixon, Yuval Tassa and Robert Pless for envigorating discussions on this subject.

REFERENCES

[1] G. B. Peterson, "A day of great illumination: B. F. Skinner's discovery of shaping," *Journal of the Experimental Analysis of Behavior*, vol. 82, no. 3, p. 317328, 2004.

[2] B. F. Skinner, *The Behavior of Organisms*. Morgantown, WV: B. F. Skinner Foundation, 1938.

[3] E. L. Allgower and K. Georg, *Numerical continuation methods: An introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1990.

[4] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[5] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research (JAIR)*, vol. 4, pp. 237–285, 1996.

[6] V. Gullapalli, "Reinforcement learning and its application to control," Ph.D. dissertation, University of Massachusetts, 1992.

[7] S. P. Singh, "Transfer of learning by composing solutions of elemental sequential tasks," *Machine Learning*, vol. 8, pp. 323–339, 1992.

[8] H. Miyamoto, J. Morimoto, K. Doya, and M. Kawato, "Reinforcement learning with via-point representation," *Neural Networks*, vol. 17, no. 3, pp. 299–305, 2004.

[9] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems (RAS)*, vol. 36, no. 1, pp. 37–51, 2001.

[10] O. G. Selfridge, R. S. Sutton, and A. G. Barto, "Training and tracking in robotics," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 1985, pp. 670–672.

[11] J. Randløv, "Shaping in reinforcement learning by changing the physics of the problem," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 2000.

[12] W. Fleming and H. Soner, *Controlled Markov Processes and Viscosity Solutions*. New York: Springer Verlag, 1993.

[13] J. Morimoto and C. G. Atkeson, "Minimax differential dynamic programming: An application to robust biped walking," in *Advances in Neural Information Processing Systems (NIPS)*, 2002, pp. 1539–1546.

[14] T. Basar and P. Bernhard, *H-infinity Optimal Control and Related Minimax Design Problems*. Boston: Birkhauser, 1995.

[15] Y. Tassa and T. Erez, "Least Squares Solutions of the HJB Equation With Neural Network Value-Function Approximators," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 1031–1041, 2007.

[16] Chris Atkeson, private communication.

[17] G. Tesauro, "Extending Q-Learning to General Adaptive Multi-Agent Systems," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[18] I. R. Fasel, "Learning to detect objects in real-time: Probabilistic generative approaches," Ph.D. dissertation, UCSD, 2006.

[19] J. A. Boyan and A. W. Moore, "Generalization in Reinforcement Learning: Safely Approximating the Value Function," in *Advances in Neural Information Processing Systems (NIPS)*, 1994, pp. 369–376.

[20] M. Schlesinger, D. Parisi, and J. Langer, "Learning to reach by constraining the movement search space," *Developmental Science*, vol. 3, no. 1, p. 6780, 2000.

[21] B. Vereijken, R. V. Emmerik, H. Whiting, and K. Newell, "Free(z)ing degrees of freedom in skill acquisition," *Journal of Motor Behavior*, vol. 24, pp. 133–142, 1992.

[22] B. Marthi, "Automatic shaping and decomposition of reward functions," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007, pp. 601–608.

[23] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[24] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 463–471.

[25] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, 1999.

[26] M. J. Matarić, "Reward functions for accelerated learning," in *Proceedings of the Eleventh International Conference on Machine Learning (ICML)*, 1994, pp. 181–189.

[27] A. Laud and G. DeJong, "The influence of reward on the speed of reinforcement learning: An analysis of shaping," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 2003, pp. 440–447.

[28] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Vision-based reinforcement learning for purposive behavior acquisition," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1995.

[29] M. Dorigo and M. Colombetti, *Robot Shaping: An Experiment in Behavior Engineering*. MIT Press/Bradford Books, 1998.

[30] F. Kaplan, P.-Y. Oudeyer, E. Kubinyi, and A. Miklósi, "Robotic clicker training," *Robotics and Autonomous Systems (RAS)*, vol. 38, no. 3-4, pp. 197–206, 2002.

[31] G. Konidaris and A. G. Barto, "Autonomous shaping: knowledge transfer in reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 489–496.

[32] G. Tesauro, "Temporal Difference Learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[33] E. Wiewiora, "Potential-Based Shaping and Q-Value Initialization are Equivalent," *Journal of Artificial Intelligence Research (JAIR)*, vol. 19, pp. 205–208, 2003.

[34] M. E. Taylor and P. Stone, "Towards reinforcement learning representation transfer," in *AAMAS*, 2007, p. 100.