Useful Policy Invariant Shaping from Arbitrary Advice

by

Paniz Behboudian

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Reinforcement learning (RL) is a powerful learning paradigm in which agents can learn to maximize sparse and delayed reward signals. Although RL has had many impressive successes in complex domains, learning can take hours, days, or even years of training data. A major challenge of contemporary RL research is to discover how to learn with less data. Previous work has shown that domain information can be successfully used to shape the reward; by adding additional reward information, the agent can learn with much less data. Furthermore, if the reward is constructed from a potential function, the optimal policy is guaranteed to be unaltered. While such *potential-based reward shaping* (PBRS) holds promise, it is limited by the need for a well-defined potential function. Ideally, we would like to be able to take arbitrary advice from a human or other agent and improve performance without affecting the optimal policy. To achieve this, the current thesis presents a simple method called *policy invariant explicit shaping* (PIES). We further expose a technical flaw in the recently introduced dynamic potential based advice (DPBA) method and show theoretically and empirically that PIES is a simple alternative that succeeds where DPBA fails.

Preface

The work of this thesis was submitted as *Useful Policy Invariant Shaping from Arbitrary Advice* for the 29th International Joint Conference on Artificial Intelligence (IJCAI) 2020 which will be held in Yokohama, Japan. This submission was authored by Paniz Behboudian, Yash Satsangi, Matthew E. Taylor, Anna Harutyunyan, and Michael Bowling.

To my dear friends Pouneh and Arash
Whom we lost too soon on the never-landing flight 752. We always remember you by your loving memories that you left behind.

Acknowledgements

Foremost, I wish to express my sincere appreciation to Professor Michael Bowling for generously sharing his wisdom with me throughout my research. He taught me to express my goals clearly and precisely. Without his persistent guidance, the goal of this thesis would not have been realized. I wholeheartedly appreciate Professor Matthew Taylor, for his strong support and great advice which proved monumental towards the completion of this work. I wish to show my gratitude to Dr. Yash Satsangi whose valuable feedback and help, assisted me all along the way. As a young researcher, I was fortunate to be in a place to learn my first steps from such amazing supervisors.

I would like to recognize the invaluable assistance of Dr. Anna Harutyunyan whose research sprit inspired me a lot.

I am indebted to my bigger family at the University of Alberta, specially the RLAI lab members whose inspirational discussions motivated my academic life. Last but not least, I would like to thank my lovely family back in my hometown and my supportive friends here, specially my dear husband Pouya, for all the passion and love they devoted to me.

Contents

1 Introduction **1**

2 Background **5**
2.1 Reinforcement Learning 5
2.2 Reward Shaping 8
2.2.1 Potential-Based Reward Shaping 9
2.2.2 Dynamic Potential-Based Shaping 10

3 The Flaw in DPBA **12**
3.1 Original DPBA Experiments and Results 12
3.2 DPBA Can Affect the Optimal Policy 15
3.3 Empirical Validation: Unhelpful Advice 18
3.4 Empirical Validation: Helpful Advice 20

4 Policy Invariant Explicit Shaping **22**
4.1 The Recipe of PIES 22
4.2 Empirical Validation of PIES 24

5 Related Work and Conclusion **29**
5.1 Related Work 29
5.2 Conclusion 31

References **32**

List of Tables

3.1 Parameters values for Figure 3.1 (b) 15
3.2 Parameters values for Figures 3.3 and 3.4 19

4.1 Parameters values for Figure 4.1 25
4.2 Parameters values for Figure 4.2a 26
4.3 Parameters values for Figure 4.2b 27

List of Figures

2.1 The backup diagrams indicating the update rule behind (a) Sarsa(0) and (b) Q-learning algorithms. 7

3.1 The y-axis shows the number of time steps taken to finish each episode (on x-axis) averaged over (a) 50 and (b) 30 runs. The shaped agent with DPBA is compared with a Sarsa learner without shaping in a) grid-world and b) cart-pole domains. Shaded areas correspond to the standard error. 14
3.2 The toy example domain with advised transitions indicated by blue arrows. The bad expert in (a) tries to keep the agent away from the goal while the good expert in (b) rewards transitions towards the goal. 18
3.3 The y-axis shows number of time steps taken to finish each episode with the bad expert. The shaped agent with corrected DPBA is compared with the shaped agent with DPBA and an unshaped Sarsa agent. Shaded areas correspond to the standard error averaged over 50 runs. 19
3.4 The y-axis shows the number of time steps taken to finish each episode with the good expert. The shaped agent with corrected DPBA is compared with the shaped agent with DPBA and an unshaped Sarsa agent. Shaded areas correspond to the standard error averaged over 50 runs. 21

4.1 The y-axis shows the number of steps taken to finish each episode in the toy example. The figures compare PIES with the corrected DPBA and a Sarsa learner without shaping when the advice is a) bad and b) good. Shaded areas correspond to the standard error over 50 runs. 25
4.2 The length of each episode as the number of steps in (a) the grid-world, and (b) the cart-pole domain. The plot depicts PIES versus the corrected DPBA and a Sarsa learner without shaping. Shaded areas correspond to the standard error over (a) 50 and (b) 30 runs. 27

Chapter 1

Introduction

Learning to maximize rewards is what a *reinforcement learning* (RL) agent tries to do, and the agent accomplishes this through seeking a policy that is a mapping from states to actions [23]. One challenging scenario is when the reward signal is sparse and delayed often resulting in an agent spending hours, days, or even years to learn a good policy. For example, the Open AI Five agent [16] required 180 years worth of game experience per day of training; similarly, the grand-master level StarCraft agent AlphaStar [24], required 16,000 matches of training data. A means to improve learning with such a sparse reward is to augment the reward function with an external source of advice, using *reward shaping*. Reward shaping is the practice of providing an RL agent with additional rewards to encourage intermediate behaviours that may accelerate learning; the additional reward is called the shaping reward. However, naively augmenting the original reward function with shaping is prone to be *policy-variant*: the optimal policy for maximizing the shaped reward might be different than that of the original reward function. For example, Randløv and Alstrøm [18] showed how an additional shaping reward that seems reasonable can in fact change the optimal policy. For an agent learning how to ride a bicycle toward a goal, they added a shaping reward in order to encourage transitions toward the goal. What they observed was that the agent got distracted: instead of pursuing the goal, it preferred to ride in a loop, repeatedly collecting the shaping reward.

Potential based reward shaping (PBRS) [14], [26], [27] allows an RL agent to

incorporate external advice without altering its optimal policy by deriving the shaping reward from a potential function. Given a static potential function, PBRS defines the shaping reward as the difference in the potentials of states (or state-action pairs) when an agent makes a transition from one state to another. Ng, Harada, and Russell [14] showed that PBRS is guaranteed to be policy invariant: using PBRS does not alter the optimal policy.

While PBRS provides a reliable shaping method that is policy invariant, it may be difficult or impossible for a person or agent to express their advice as a potential-based function. Instead, it would be preferable to allow the use of more direct or intuitive advice in the form of an arbitrary function. The ideal reward shaping method then would have three properties:

1. Be able to use an arbitrary reward function as advice.
2. Keep the optimal policy unchanged in the presence of the additional advice.
3. Improve the speed of learning of the RL agent when the advice is good.[1]

Harutyunyan, Devlin, Vrancx, *et al.* [6] attempted to tackle the same problem by proposing the framework of *dynamic potential-based advice* (DPBA), where the idea is to dynamically learn a potential function from arbitrary advice, which can be used to define the shaping reward concurrently. Importantly, the authors claimed that if the potential function is initialized to zero then DPBA is guaranteed to be policy invariant. We show in this work that this claim is unfortunately not true, and hence, their approach is *not* policy invariant. We confirm our finding theoretically and empirically. We then propose a fix to the method by deriving a correction term for correcting the policy, and show that the result is theoretically sound, and empirically policy-invariant. However, our empirical analysis shows that the *corrected* DPBA fails to accelerate the learning of an RL agent with good external advice (e.g., the advice was useful for the DPBA method). Therefore, the corrected DPBA does not satisfy all three of the properties of ideal reward shaping.

[1] If the advice is poor, or even adversarial, one would expect the initial learning to suffer, even if the agent can eventually discover the optimal policy.

In the attempt to find a method that meet all three properties, we introduce a simple algorithm, *policy invariant explicit shaping* (PIES) and show that it can achieve all of the goals for a reward shaping method: PIES can allow for arbitrary advice, is policy invariant, and can accelerate the learning of an RL agent. PIES biases the agent's policy toward the advice at the start of the learning, when the agent is the most in need of guidance. Over time, PIES gradually decays this bias and relies on the original value function,ensuring policy invariance. Several experiments confirm that PIES ensures convergence to the optimal policy when the advice is misleading and also accelerates learning when the advice is useful.

Concretely, this thesis makes the following contributions:

1. Identifies an important flaw in a previous reward shaping method.

2. Verifies the flaw exists, both empirically and theoretically, by showing that the optimal policy can be altered by advice.

3. Provides a correction term to the method, but empirically shows that it introduces additional complications, where good advice no longer improves learning speed.

4. Introduces an approach that achieves the goals of the original method, verifying empirically and theoretically that it meets the three criteria desired from a reward shaping method.

This thesis is structured in four more chapters. In Chapter 2, we provide preliminary background materials including RL and reward shaping. Chapter 3 is dedicated to describing the DPBA paper flaw and how its corrected version behaves. In this chapter before revealing the problem with DPBA, we explain the original paper's test-bed and make an attempt to replicate its results. Further we provide our theoretical findings related to the flaw followed by supporting empirical results. After fixing DPBA with the correction term derived from the theoretical results, we show how the corrected version of the DPBA behaves in different scenarios. In chapter 4, we introduce PIES, the proposed alternative to DPBA, and show how it can achieve the three goals of

ideal shaping theoretically and empirically. Finally, the last chapter, Chapter 5, briefly expands on other related work and summarizes all the works done in this thesis.

Chapter 2

Background

In this section we provide some background material needed for a detailed discussion of the thesis contributions. We start with a general overview of RL and then we dive deeper into the reward shaping literature.

2.1 Reinforcement Learning

An RL agent interacts with its surrounding environment by taking actions and receiving rewards and observations in return. Unlike supervised learning that deals with a set of labeled examples, the RL agent is not provided with labels: it learns which action gives rise to the maximum reward only by trying different actions. One challenge is that actions might have long-term effects and affect future rewards. In this thesis we are interested in RL problems that can be formulated as a *Markov Decision Process* (MDP) [17]. Such an MDP is described by the tuple $\langle S, A, T, \gamma, R \rangle$. At each time step, the environment is in a state $s \in S$, the agent takes an action $a \in A$, and the environment transitions to a new state $s' \in S$, according to the transition probabilities $T(s, a, s') = \Pr(s'|s, a)$. Additionally, the agent (at each time step) receives a reward for taking action a in state s according to the reward function $R(s, a)$. Finally, γ is the discount factor, specifying how to trade off future rewards and current rewards.

A deterministic policy π is a mapping from states to actions, $\pi : S \to A$, that is, for each state, s, $\pi(s)$ returns an action, $a = \pi(s)$. The *state-action value function* $Q^\pi(s, a)$ is defined as the expected sum of discounted rewards

the agent will get if it takes action a in state s and follows the policy π thereafter.

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \middle| s_t = s, a_t = a, \pi\right].$$

The agent aims to find the optimal policy denoted by π^* that maximizes the expected sum of discounted rewards, and the state-action value function associated with π^* is called the optimal state-action value function, denoted by $Q^*(s,a)$. Given the optimal value function $Q^*(s,a)$, the agent can retrieve the optimal policy by acting greedily with respect to the optimal value function:

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s,a),$$

and $Q^*(s,a)$ is defined as:

$$Q^*(s,a) = \max_{\pi \in \prod} Q^{\pi}(s,a),$$

where $\prod$ is the space of all policies.

The action value function for a given policy π satisfies the *Bellman equation*:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \mathbb{E}_{s',a'}[Q^{\pi}(s',a')],$$

where s' is the state at the next time step and a' is the action the agent takes on the next time step, and this is true for all policies.

The Bellman equation for the optimal policy π^* is called the *Bellman optimality equation*:

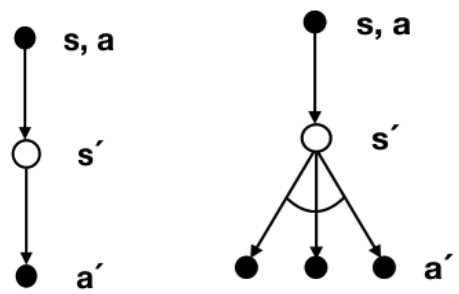
$$Q^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s',a'}[Q^*(s',a')].$$

The idea behind many reinforcement learning algorithms is to learn the optimal value function Q^* iteratively. For example Sarsa(0) learns Q-values with the following update rule, at each time step t (Q_0 can be initialized arbitrarily):

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \delta_t, \tag{2.1}$$

where

$$\delta_t = R(s_t, a_t) + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$



(a) Sarsa(0) (b) Q-learning

Figure 2.1: The backup diagrams indicating the update rule behind (a) Sarsa(0) and (b) Q-learning algorithms.

is the *temporal-difference* error (TD-error) [20], s_t and a_t denotes the state and action at time step t, Q_t denotes the estimate of Q^* at time step t, and α_t is the learning rate at time step t. The TD-error implies that the agent bootstraps from its current estimation of the value function for computing the target (the temporally successive estimates) in the error term. If all state-action pairs continue to be visited (for infinite number of times), with an appropriate learning rate and a bounded reward, these Q estimates are guaranteed to converge to Q^* for all s, a, and the policy converges to π^* [23], [25].

Q-learning is another RL approach to estimate Q^* with a different update rule:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t[R(s_t, a_t) + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)],$$

where s_t and a_t denotes the state and action at time step t, Q_t denotes the estimate of Q^* at time step t, and α_t is the learning rate at time step t. Q-learning belongs to the family of *off-policy* algorithms: the target policy for which the optimal value function is being learned differs from the behaviour policy which the agents follows to collect the learning samples. Here the target policy is greedy while the behaviour policy could be some different policy such as ϵ-greedy (which will be discussed shortly). The backup diagrams for Sarsa(0) and Q-learning are shown in Figure 2.1 (a) and Figure 2.1 (b) respectively, to illustrate the basis of their updates.

While learning iteratively, a good policy should dedicate some time for

exploration to discover new states and actions as well as some time to *exploit* the knowledge it already gained—just like what humans do, *e.g.*, ordering a new dish in a restaurant or rather choosing from those that have been already tried, with maximum deliciousness. One such policy is ϵ-greedy under which the probability of choosing a random action is ϵ and the greedy action is $1 - \epsilon$.

2.2 Reward Shaping

Similar to a sculptor shaping clay to form the final statue, the psychologist B. F. Skinner proposed *shaping* for animal training: a complex task can be learned faster by first solving its simpler approximates [5], [19], [23]. Shaping the *reward* in RL is meant to facilitate learning by reinforcing the intermediate behaviours related to achieving the main goal. One of the oldest examples of applying reward shaping is the work by Gullapalli and Barto [5] where a simulated robot hand was trained to press a key via learning a set of successive approximations to the original task.

Randløv and Alstrøm [18] also utilized reward shaping to accelerate their RL agent trying to learn how to ride a bicycle. In that paper when they provided positive reinforcement for making transitions toward the goal, the agent was misguided to find a loop as the optimal behaviour, accumulating positive rewards over and over:

> "In our first experiments we rewarded the agent for driving towards the goal but did not punish it for driving away from it. Consequently the agent drove in circles with a radius of 20-50 meters around the starting point. Such behavior was actually rewarded by the reinforcement function ..."

Consequently, the addition of an arbitrary reward can alter the optimal policy of a given MDP, and therefore be policy-variant. Ng, Harada, and Russell [14] proposed a mathematical approach to address the policy-variance problem. More specifically, they derived the sufficient conditions such that the set of optimal policies are invariant to the change in reward function [15]. In the

following subsections we describe the work by Ng, Harada, and Russell [14] in more detail and several important successive works that built upon their paper to empower the strengths of reward shaping in RL.

2.2.1 Potential-Based Reward Shaping

Potential-based reward shaping (PBRS) addresses the problem of adding a shaping reward function F to an existing MDP reward function R, without changing the optimal policy by defining F to be the difference in the *potential* of the current state s and the next state s' [14]. Specifically, PBRS restricts the shaping reward to the following form: $F(s, s') \coloneqq \gamma\Phi(s') - \Phi(s)$, where $\Phi : S \to \mathbb{R}$ is the potential function. Ng, Harada, and Russell showed that expressing F as the difference of potentials is the sufficient condition for the agent to be *policy invariant*. That is, if the original MDP $\langle S, A, T, \gamma, R\rangle$ is denoted by M and the shaped MDP $\langle S, A, T, \gamma, R + F\rangle$ is denoted by M' (M' is same as M but offers the agent an extra reward F in addition to R) then the optimal value function of M and M' for any state-action pair (s, a) satisfy:

$$Q^*_{M'}(s, a) = Q^*_M(s, a) - \Phi(s)$$

where the term $\Phi(s)$ is the *bias term*. Given $Q^*_{M'}$, the optimal policy π^* can simply be obtained by adding the bias term:

$$\pi^*(s) = \arg\max_{a\in A} Q^*_M(s, a) = \arg\max_{a\in A}(Q^*_{M'}(s, a) + \Phi(s)).$$

Because the bias term only depends on the agent's state, the optimal policy of the shaped MDP M', does not differ from that of the original MDP M.

To also include the shaping reward on actions, Wiewiora et al. [27] extended the definition of F to state-action pairs by defining F to be: $F(s, a, s', a') \coloneqq \gamma\Phi(s', a') - \Phi(s, a)$, where Φ is dependent on both the state and the action of the agent. Now the bias term is also dependent on the action taken at state s, therefore the agent must follow the policy

$$\pi^*(s) = \arg\max_{a\in A}(Q^*_{M'}(s, a) + \Phi(s, a))$$

in order to be policy-invariant.

In the work by Wiewiora [26], potential-based reward shaping with static potentials (which do not change with time), proved to be equivalent to initializing the value function with the potential function, given that the learning algorithm is using a tabular temporal difference method with an advantage-based exploration policy and the same sequence of experience during learning. Thus far, none of the methods that we discussed admits an arbitrary form of advice, hence fulfills the ideal shaping goals provided in 1. In the next subsection we further explain some approaches with potentials that changes dynamically with time and whether dynamic potentials can be harnessed to achieve our shaping goals.

2.2.2 Dynamic Potential-Based Shaping

PBRS, as discussed in Section 2.2.1, is restricted to external advice that can be expressed in terms of a potential function. Therefore, it does not satisfy the first of the three goals discussed in Chapter 1, which is being able to directly use and arbitrary form of advice. Finding a potential function Φ that accurately captures the advice can be challenging. To allow an expert to specify an arbitrary function R^{expert} and still maintain all the properties of PBRS one might consider *dynamic PBRS*.

Dynamic PBRS uses a potential function Φ_t that changes over time to form a dynamic shaping reward F_t, where subscript t indicates the time over which F and Φ vary. Devlin and Kudenko [3] used dynamic PBRS as $F_{t+1}(s, s') := \gamma\Phi_{t+1}(s') - \Phi_t(s)$, where t and $t+1$ are the times that the agent arrives at states s and s', respectively. They derived the same guarantees of policy invariance for dynamic PBRS as static PBRS. To admit an arbitrary reward, Harutyunyan, Devlin, Vrancx, *et al.* [6] suggested learning a dynamic potential function Φ_t given the external advice in the form of an arbitrary bounded function, R^{expert}. To do so, the following method named *dynamic potential based advice* (DPBA) is proposed: define $R^{\Phi} := -R^{expert}$, and learn a *secondary value function* Φ via the following update rule at each time step:

$$\Phi_{t+1}(s, a) := \Phi_t(s, a) + \beta\delta_t^{\Phi} \tag{2.2}$$

where $\Phi_t(s,a)$ is the current estimate of Φ, β is the Φ function's learning rate, and

$$\delta_t^{\Phi} \coloneqq R^{\Phi}(s,a) + \gamma\Phi_{t+1}(s',a') - \Phi_t(s,a)$$

is the Φ function's TD error. All the while, the agent learns the Q values using Sarsa (*i.e.*, according to Equation 2.1). In addition to the original reward $R(s,a)$ the agent receives a shaping reward given as:

$$F_{t+1}(s,a,s',a') \coloneqq \gamma\Phi_{t+1}(s',a') - \Phi_t(s,a), \tag{2.3}$$

that is, the difference between the consecutively updated values of Φ.

Harutyunyan, Devlin, Vrancx, *et al.* [6] suggested that with this form of reward shaping, $Q^*_M(s,a) = Q^*_{M'}(s,a) + \Phi_0(s,a)$ for every s and a and therefore to obtain the optimal policy π^*, the agent should be greedy with respect to $Q^*_{M'}(s,a) + \Phi_0(s,a)$ by the following rule:

$$\pi^*(s) = \arg\max_{a\in A}(Q^*_{M'}(s,a) + \Phi_0(s,a)), \tag{2.4}$$

and thus if $\Phi_0(s,a)$ is initialized to zero, the above biased policy in Equation 2.4, reduces to the original greedy policy:

$$\pi^*(s) = \arg\max_{a\in A} Q^*_{M'}(s,a) = \arg\max_{a\in A} Q^*_M(s,a).$$

In the next chapter we first try to replicate the original empirical results of DPBA and then explain the flaw in their policy invariance derivation and provide empirical evidence that it is not policy invariant. In other words, we show cases where incorporating advice through DPBA leads the agent to find non-optimal policies.

Chapter 3

The Flaw in DPBA

The previous chapter described DPBA, a method that can incorporate an arbitrary expert's advice in a reinforcement learning framework by learning a potential function Φ iteratively and concurrently with the shaped state-action values, $Q_{M'}$. Harutyunyan, Devlin, Vrancx, *et al.* [6] claimed that if the initial values of Φ, Φ_0, are initialized to zero, then the agent can simply follow a policy that is greedy with respect to $Q_{M'}$ to achieve policy invariance. In this chapter we show that this claim is unfortunately not true: initializing $\Phi_0(s, a)$ to zero is not sufficient to guarantee policy invariance. But before proving our claim theoretically and empirically, we replicate the same results from the original paper with a description of their test-bed (which is also used in the next chapter).

3.1 Original DPBA Experiments and Results

DPBA was empirically evaluated on two episodic tasks: a 20×20 grid-world and a cart-pole problem. In the grid-world experiment, the agent starts each episode from the top left corner until it reaches the goal located at the bottom right corner, within a maximum of 10,000 steps. The agent can move along the four cardinal directions and the state is the agent's coordinates (x, y). The reward function is +1 upon arrival at the goal state and 0 elsewhere. The advice, R^{expert}, for any state action is:

$$R^{expert}(s, a) := \begin{cases} +1 & \textit{if } a \textit{ is } \text{right } \textit{or } \text{down} \\ 0 & \textit{otherwise} \end{cases}.$$

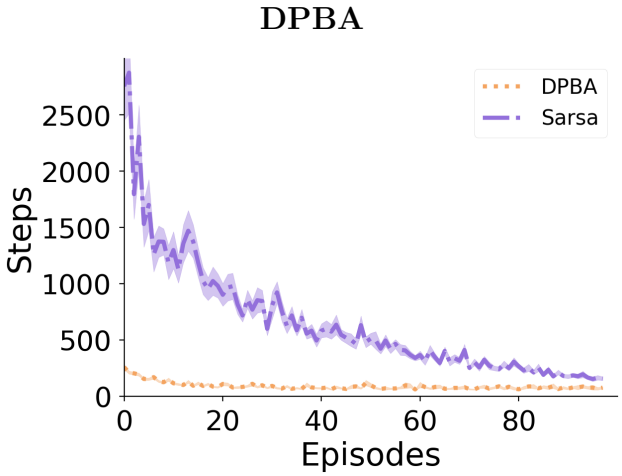
In the cart-pole task [13], the goal is to balance a pole on top of a cart as long as possible. The cart can move along a track and each episode starts in the middle of the track with the pole upright. There are two possible actions: applying a force of $+1$ or -1 to the cart. The state consists of a four dimensional continuous vector, indicating the angle and the angular velocity of the pole, and the position and the velocity of the cart. An episode ends when the pole has been balanced for 200 steps or the pole falls, and the reward function encourages the agent to balance the pole. To replicate this experiment in this thesis, we used the OpenAI Gym [2] implementation (cartpole-v0).[1] The advice for this task is defined as:

$$R^{expert}(s, a) := o(s, a) \times c,$$

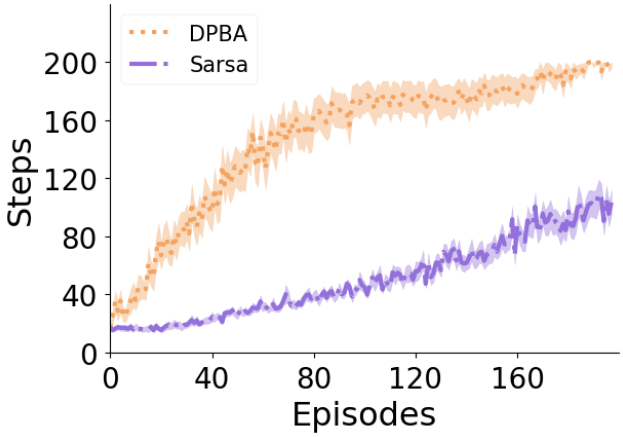
where $o : S \times A \to \{0, 1\}$ is a function that triggers when the pole direction is aligned with the force applied to the cart (*i.e.*, when the cart moves in the same direction as the pole is leaning towards, the agent is rewarded). We set $c = 0.1$.

Figure 3.1 shows the performance of the DPBA method, compared to a simple Sarsa learner not receiving any expert advice in the grid-world and the cart-pole domains. We used the same set of hyper-parameters as used by Harutyunyan, Devlin, Vrancx, *et al.* [6] for the grid-world. For learning the cart-pole task, the agent used linear function approximation for estimating the value function via Sarsa(λ) and a tile-coded feature representation [21] with the implementation from the open-source software provided in a blog post by Sutton [22]. The weights for Q and Φ were initialized uniformly at random between 0 and 0.001. For tile-coding, we used 8 tilings, each with 2^4 tiles (2 for each dimension). We used a wrapping tile for the angle of the pole for a more accurate state-representation. With a wrapping tile one can generalize over a range (*e.g.*, $[0, 2\pi]$) rather than stretching the tile to infinity, and then

[1] There are slight differences between our implementation of cart-pole and the version used in the DPBA paper [6], making the results not directly comparable. In that paper, 1) if the cart attempts to move beyond the ends of the track, the cart bounces back, and 2) there is a negative reward if the pole drops and otherwise the reward is zero. In contrast, in OpenAI Gym, 1) if the cart moves beyond the track's boundaries, the episode terminates, and 2) the reward function is $+1$ on every step the pole is balanced and 0 if the pole falls.



(a) Grid-World



(b) Cart-Pole

Figure 3.1: The y-axis shows the number of time steps taken to finish each episode (on x-axis) averaged over (a) 50 and (b) 30 runs. The shaped agent with DPBA is compared with a Sarsa learner without shaping in a) grid-world and b) cart-pole domains. Shaded areas correspond to the standard error.

wrap-around [22]. λ was set to 0.9 and γ to 1. For learning rates of Q and Φ value functions, α and β, we swept over the values $[0.001, 0.002, 0.01, 0.1, 0.2]$. The best parameter values according to the area under curve (AUC) of each line for Figure 3.1 (b) are reported in Table 3.1.

Note that in the grid-world task the desired behaviour is to reach the goal as quickly as possible. Consequently, for this task lower is better in plots (such as the ones in Figure 3.1) showing steps (y-axis) versus episodes (x-axis). In contrast, the nature of the goal in the cart-pole task dictates to take as much

Table 3.1: Parameters values for Figure 3.1 (b)

Agent	α	β
Sarsa	0.1	-
DPBA	0.02	0.1

steps as possible during each episode, as the desired behaviour. Therefore, for cart-pole the up is good in a plot showing the length of each episode through its axes as in Figure 3.1 (b). The results in Figure 3.1 agree with the prior work, showing that the agent using the DPBA method learned faster with this good advice, relative to not using the advice (*i.e.*, the DPBA line is converging faster to the optimal behaviour). These results show that the DPBA method satisfies criterion 1 (it can use arbitrary rewards) and criterion 3 (good advice can improve performance). However, as we argue in the rest of this chapter, a flaw in the proof of the original paper means that criterion 2 is not satisfied: the optimal policy can change, *i.e.*, advice can cause the agent to converge to a sub-optimal policy. This was not empirically tested in the original paper and thus this failure was not noticed. In the next section we prove our claim by re-deriving Equations 16 and 17 from the original paper correctly.

3.2 DPBA Can Affect the Optimal Policy

To prove our claim, we start by defining terms. We will compare Q-value estimates for a given policy π in two MDPs, the original MDP denoted by M described by the tuple $\langle S, A, T, \gamma, R\rangle$, and the MDP that is shaped by DPBA, M', described by the tuple $\langle S, A, T, \gamma, R + F_{t+1}\rangle$, where $F_{t+1}(s, a, s', a') = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$.

Let $R'_{t+1} := R + F_{t+1}$, given a policy π, at any time step t, $Q^{\pi}_{M'}$ can be defined as:

$$Q^{\pi}_{M'}(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R'_{t+k+1}(s_{t+k}, a_{t+k}, s_{t+k+1}, a_{t+k+1}) \middle| \begin{matrix} s_t = s, \\ a_t = a \end{matrix}\right].$$

By writing R' in terms of R and F:

$$= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \left(R(s_{t+k}, a_{t+k}) \right.\right. \\ \left.\left. + F_{t+k+1}(s_{t+k}, a_{t+k}, s_{t+k+1}, a_{t+k+1})\right) \middle| \begin{matrix} s_t = s, \\ a_t = a \end{matrix}\right].$$

The first term in the above expression (after separating the expectation) is the value function for the original MDP M for policy π so we can rewrite the expression as:

$$= \underbrace{\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \middle| \begin{matrix} s_t = s, \\ a_t = a \end{matrix}\right]}_{=Q_M^{\pi}(s,a)} \\ + \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k F_{t+k+1}(s_{t+k}, a_{t+k}, s_{t+k+1}, a_{t+k+1}) \middle| \begin{matrix} s_t = s, \\ a_t = a \end{matrix}\right]. \tag{3.1}$$

The second term in Equation 3.1 can be expanded based on Equation 2.3 as follows:

$$\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k F_{t+k+1}(s_{t+k}, a_{t+k}, s_{t+k+1}, a_{t+k+1}) \middle| \begin{matrix} s_t = s, \\ a_t = a \end{matrix}\right] \\ = \mathbb{E}\left[\sum_{k=0}^{\infty} \left(\gamma^{k+1} \Phi_{t+k+1}(s_{t+k+1}, a_{t+k+1}) \right.\right. \\ \left.\left. - \gamma^k \Phi_{t+k}(s_{t+k}, a_{t+k})\right) \middle| \begin{matrix} s_t = s, \\ a_t = a \end{matrix}\right]. \tag{3.2}$$

The two terms inside the infinite summation look quite similar, motivating us to rewrite one of them by shifting its summation variable k. This shift will let identical terms be cancelled out. However, we need to be careful. First, we rewrite the sums in their limit form. An infinite sum can be written as:

$$\sum_{i=i_0}^{\infty} x_i \coloneqq \lim_{W \to \infty} \sum_{i=i_0}^{W} x_i.$$

Using this definition, in Equation 3.2 we can shift the first term's iteration

variable as:

$$
\begin{aligned}
=&\mathbb{E}\left[\lim_{W\to\infty}\left(\sum_{k=1}^{W+1}\gamma^k\Phi_{t+k}(s_{t+k},a_{t+k})\right.\right.\\
&\qquad\left.\left.-\sum_{k=0}^{W}\gamma^k\Phi_{t+k}(s_{t+k},a_{t+k})\right)\middle|\begin{matrix}s_t=s,\\a_t=a\end{matrix}\right]\\
=&\mathbb{E}\left[\lim_{W\to\infty}\left(\gamma^{W+1}\Phi_{t+W+1}(s_{t+W+1},a_{t+W+1})\right.\right.\\
&\qquad\left.\left.-\ \gamma^0\Phi_t(s_t,a_t)\right)\middle|\begin{matrix}s_t=s,\\a_t=a\end{matrix}\right].
\end{aligned}
\tag{3.3}
$$

In Equation 3.3, if $\Phi_t^\pi(s,a)$ is bounded, then the first term inside the limit will go to 0 as W approaches infinity[2], and the second term does not depend on W and can be pulled outside the limit:

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k F_{t+k+1}(s_{t+k},a_{t+k},s_{t+k+1},a_{t+k+1})\middle|\begin{matrix}s_t=s,\\a_t=a\end{matrix}\right]\\
&=\mathbb{E}\left[-\Phi_t(s_t,a_t)\middle|\begin{matrix}s_t=s,\\a_t=a\end{matrix}\right]=-\Phi_t(s,a).
\end{aligned}
\tag{3.4}
$$

Back to Equation 3.1, if we apply Equation 3.4, we will have:

$$Q^\pi_{M'}(s,a)=Q^\pi_M(s,a)-\Phi_t(s,a),$$

for all π (given that $s_t=s$ and $a_t=a$). Thus, for an agent to retrieve the optimal policy π^*_M given $Q^*_{M'}(s,a)$, it must act greedily with respect to $Q^*_{M'}(s,a)+\Phi_t(s,a)$:

$$\pi^*_M(s)=\arg\max_{a\in A}\left(Q^*_{M'}(s,a)+\Phi_t(s,a)\right). \tag{3.5}$$

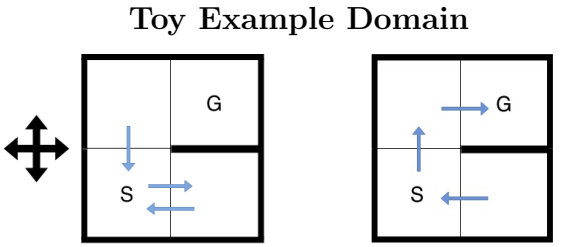
Equation 3.5 differs from Equation 2.4 (corresponding to Equation 17 in Harutyunyan, Devlin, Vrancx, *et al.* [6]) in that the bias term is Φ_t and not Φ_0. In other words, at every time step the Q values of the agent are biased by the *current estimate* of the potential function and not by the initial value of the potential function. The derived relation in Equation 17 of Harutyunyan, Devlin, Vrancx, *et al.* [6] is only valid for the first state-action pair that the

[2]Note that this term will go to zero only for infinite horizon MDPs. In practice, it is common to assume a finite horizon MDP with a terminal state, in such cases, this extra term will remain and must be removed, for example, by defining the potential of the terminal state to be zero.

agent visits (at $t = 0$). For the rest of the time steps, it is not accurate to use the first time step's bias term to compensate the shaped value function. Thus, the zero initialization of Φ is not a sufficient condition for the agent to recover the true Q values of the original MDP, and it cannot be used to retrieve the optimal policy of the original MDP.

3.3 Empirical Validation: Unhelpful Advice

We empirically validate the result above with a set of experiments. First, consider a deterministic 2×2 grid-world which we refer to it as the *toy example*, depicted in Figure 3.2. The agent starts each episode from state S and can move in the four cardinal directions (as depicted in the figure) until it reaches the goal state G (within a maximum of 100 steps). Moving towards a wall (indicated by bold lines), causes no change in the agent's position. The reward is 0 on every transition except the one ending in the goal state, resulting in a reward of +1 and episode termination. For advice, we assume that the "expert" rewards the agent for making transitions away from the goal. Blue arrows inside the grid in Figure 3.2 (a) represent the expert advised state-transitions. The agent receives a +1 from the expert by executing the advised transitions. Because this advice is encouraging poor behavior, we expect that it would slow down the learning (rather than accelerate it), but if a shaping method is policy invariant, the agent should still eventually converge to the optimal policy.



(a) Bad expert (b) Good expert

Figure 3.2: The toy example domain with advised transitions indicated by blue arrows. The bad expert in (a) tries to keep the agent away from the goal while the good expert in (b) rewards transitions towards the goal.

The learner uses Sarsa(0) to estimate Q values with $\gamma = 0.3$. We ran the experiment for both the learner with *corrected DPBA*, Equation 3.5, and the learner with *DPBA*, Equation 2.4, using an ϵ-greedy policy. Φ and Q were initialized to 0 and ϵ was decayed from 0.1 to 0. For the learning rates of the Q and Φ value functions, α and β, we swept over the values $[0.05, 0.1, 0.2, 0.5]$. The best values, according to the AUC of the each line are reported in Table 3.2.

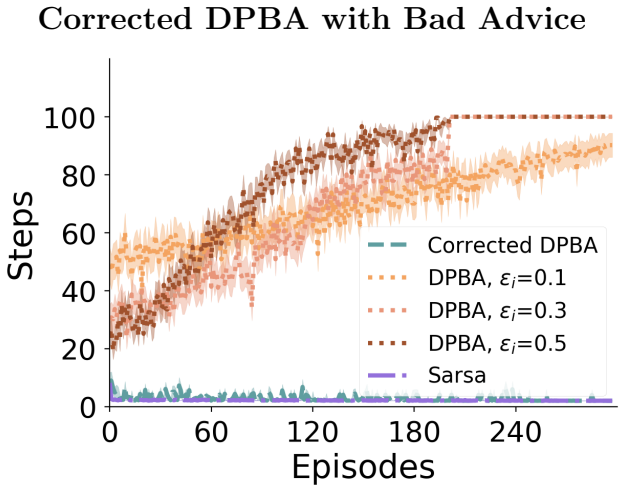


Figure 3.3: The y-axis shows number of time steps taken to finish each episode with the bad expert. The shaped agent with corrected DPBA is compared with the shaped agent with DPBA and an unshaped Sarsa agent. Shaded areas correspond to the standard error averaged over 50 runs.

Table 3.2: Parameters values for Figures 3.3 and 3.4

Agent	α	β
Sarsa	0.05	-
DPBA, good advice	0.2	0.5
DPBA, bad advice	0.2	0.5
corrected DPBA, good advice	0.2	0.1
corrected DPBA, bad advice	0.05	0.2

Figure 3.3 depicts the length of each episode as the number of steps taken to finish the episode, therefore, lower lines indicate better performance. The results were averaged over 50 independent runs. Figure 3.1 (a), The *Sarsa* line indicates the learning curve for a Sarsa(0) agent without shaping. Figure 3.3

shows that DPBA does not converge to the optimal policy with this bad advice. This figure also confirms our result that Φ_t (and not Φ_0 as proposed in Harutyunyan, Devlin, Vrancx, *et al.* [6]) is the appropriate correction term to recover the optimal policy for maximizing the MDP's original reward. Finally, we verify that this is not simply an artefact of the agent exploiting too soon, and repeat the same experiments for different exploration rates, ϵ. We considered two more different values for initial exploration rate, ϵ_i: 0.3 and 0.5. The corresponding lines in Figure 3.3 confirm that even with more exploration DPBA cannot obtain the optimal policy.

3.4 Empirical Validation: Helpful Advice

The previous section showed that DPBA is not a policy invariant shaping method since initializing the values of Φ to zero is not a sufficient condition for policy invariance. We showed that DPBA can be corrected by adding the correct bias term and indeed it is then policy invariant. While the addition of the correct bias term guarantees policy invariance, we still need to test our third criterion for the desired reward shaping algorithm — does corrected DPBA accelerate the learning of a shaped agent with good expert advice?

Figure 3.4 shows the results for repeating the same experiments as the previous section but with the good advice which is shown in Figure 3.2 (b) (*i.e.*, from each state the expert encourages the agent to move towards the goal). Here, since the expert is encouraging the agent to move towards the goal, we expect the shaped agent to learn faster than the agent that is not receiving a shaping reward. However, Figure 3.4 shows that the corrected agent does not learn faster with good advice. To our surprise, the advice actually slowed down the learning, even though the corrected DPBA agent did eventually discover the optimal policy, as expected. To explain the corrected DPBA behaviour, one needs to look closely at how the Q and Φ estimates are changing. The corrected DPBA adds the latest value of Φ at each time step, to correct the shaped Q value; however, the Φ value which has been used earlier to shape the reward function might be different than the latest value.

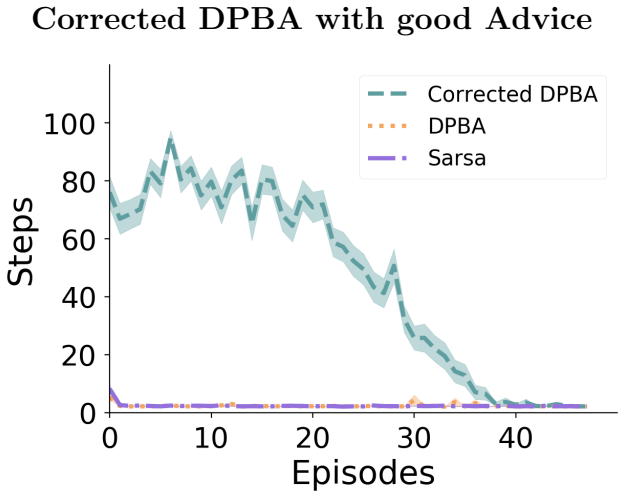


Figure 3.4: The y-axis shows the number of time steps taken to finish each episode with the good expert. The shaped agent with corrected DPBA is compared with the shaped agent with DPBA and an unshaped Sarsa agent. Shaded areas correspond to the standard error averaged over 50 runs.

Let us consider the case that Φ has been initialized to zero and the advice is always a positive signal, enforcing the Φ values to be negative. With such a Φ the latest Φ values are likely more negative than the earlier values used for shaping the reward, which in fact discourages the desired behaviour.

While the corrected DPBA guarantees policy invariance, it fails in satisfying the third goal for ideal reward shaping (*i.e.*, speed up learning of an agent with a helpful advice). We further depict the performance of the corrected DPBA in the original testbeds (grid-world and cart-pole) in the next chapter. The main conclusion of this chapter is that none of the mentioned methods for incorporating expert advice satisfies the three goals outlined in the introduction. DPBA as proposed in Harutyunyan, Devlin, Vrancx, *et al.* [6] can lead to faster learning if the expert offers good advice but it is not policy invariant. The corrected DPBA proposed in this chapter is provably policy invariant but it can lead to slower learning even when provided with good advice.

Chapter 4

Policy Invariant Explicit Shaping

In this chapter, we introduce the *policy invariant explicit shaping* (PIES) algorithm that satisfies all of the goals we specified in Chapter 1 for reward shaping. We also demonstrate that it stands out in all of the experiments mentioned so far.

4.1 The Recipe of PIES

PIES is a simple algorithm that admits arbitrary advice, is policy invariant, and speeds up the learning of the agent depending on the expert advice.

Unlike potential based reward shaping, with PIES the underlying idea is to use the expert advice explicitly without modifying the original reward function. Not changing the reward function is the principal feature that both simplifies PIES and makes analysing how it works easier. The PIES agent learns the original value function Q_M, while concurrently learning a secondary value function Φ that accumulates the expert advice, *i.e.* $R^{\Phi} = R^{expert}$. For an agent learning with Sarsa(0) and PIES the update for Q would be the same as Equation 2.1 and for Φ would be:

$$\Phi_{t+1}(s,a) \coloneqq \Phi_t(s,a) + \beta\delta_t^{\Phi},$$

where $\Phi_t(s,a)$ is the current estimate of Φ, β is the Φ function's learning rate, and

$$\delta_t^{\Phi} \coloneqq R^{\Phi}(s,a) + \gamma\Phi_{t+1}(s',a') - \Phi_t(s,a)$$

is the Φ function's TD error. To exploit the arbitrary advice, the agent explicitly biases the agent's policy towards the advice by adding Φ to Q_M at each time step t, weighted by a parameter ξ_t, where ξ_t decays to 0 before the learning terminates[1]. For example, for a Sarsa(0) agent equipped with PIES, when the agent wants to act greedily, it will pick the action that maximizes $Q_t(s,a) + \xi_t \Phi_t(s,a)$ at each time step. Thus, the optimal policy would be:

$$\pi^*_M(s) = \arg\max_{a \in A} \left(Q^*_M(s,a) + \xi_t \Phi_t(s,a) \right).$$

The parameter ξ_t controls that to what extent the agent's current behaviour is biased towards the advice. Decaying ξ_t to 0 over time removes the effect of shaping, guaranteeing that the agent will converge to the optimal policy, making PIES policy-invariant. The speed of decaying ξ determines how long the advice will continue to influence the agent's learned policy. Choosing the

Algorithm 1 PIES with Sarsa(0) updates for Q, Φ

Algorithm parameters: step sizes α for Q and β for Φ, small $\epsilon > 0$, decay factor ξ
Initialize $Q(s,a)$ and $\Phi(s,a)$ for all $s \in \mathbf{S}$, $a \in \mathbf{A}$ arbitrarily except that $Q(terminal, .) = 0$, $\Phi(terminal, .) = 0$, $\xi = 1$
for each episode **do**
    Initialize S
    Choose A from S using policy derived from $Q + \xi\Phi$ (*e.g.*, ϵ-greedy)
    for each step of episode until S is terminal **do**:
        Take action A, observe R, R^{expert}, S'
        Choose A' from S' using policy derived from $Q + \xi\Phi$ (*e.g.*, ϵ-greedy)
        if $\xi_t > 0$ **then**
            $\Phi(S,A) \leftarrow \Phi(S,A) + \beta[R^{expert} + \gamma\Phi(S',A') - \Phi(S,A)]$
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
        $S \leftarrow S'; A \leftarrow A';$
    $\xi \leftarrow next\_\xi(\xi)$

decay speed of ξ can be related to how beneficial it is to utilize the advice and can be done in many different ways. For this paper, we only decrease ξ at

[1]Unlike DPBA that accumulates the negation of R^{expert}, in PIES Φ is accumulating the R^{expert}. Therefore, in PIES we add Φ to shape the Q values for the policy. One can keep the Φ reward function, R^{Φ}, as same as DPBA by easily reverting the sign to set $R^{\Phi} = -R^{expert}$ and add $-\Phi$ to Q to shape the agent with PIES.

the end of each episode with a linear regime. More specifically, the value of ξ during episode e is:

$$\xi_e := \begin{cases} \xi_{e-1} - \frac{1}{C} & \text{if } \xi_{e-1} \text{ is not } 0 \\ 0 & \text{otherwise} \end{cases}, \tag{4.1}$$

where, $\xi_1 = 1$, and C is a constant that determines how fast ξ will be decayed; *i.e.*, the greater C is the slower the bias decays. The pseudo-code for a Sarsa(0) agent with PIES is provided in Algorithm 1. In the next section we show the performance of PIES in all of the experimental settings mentioned for variants of DPBA.

4.2 Empirical Validation of PIES

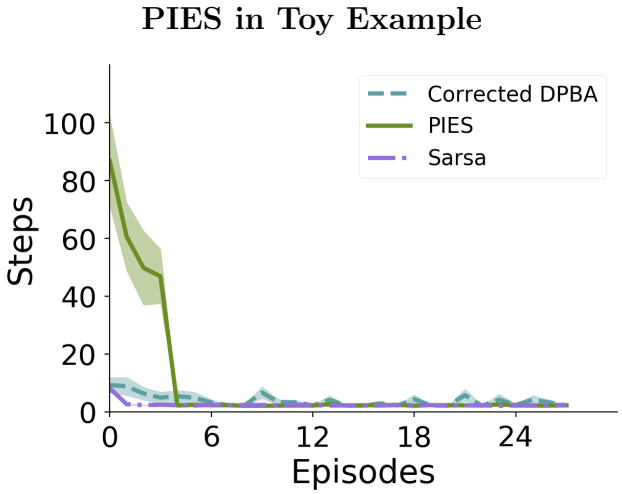
We first demonstrate empirically that PIES fulfills all three goals in the toy example. We then show how it performs against previous methods in the grid-world and the cart-pole problems (which were originally tested with DPBA), when provided with good advice. All the domains specifications are the same as before.

The plot of the agents' performance in the toy example in Figure 4.1 shows learning curves of the corrected DPBA, PIES, and the Sarsa learner. Figure 4.1 (a) is for the bad expert shown in 3.2 (a) and Figure 4.1 (b) is for the good expert as in 3.2 (b). The curves are averaged over 50 independent runs. Sarsa(0) was used to estimate state-action values with $\gamma = 0.3$ and an ϵ-greedy policy. Φ and Q were initialized to 0 and ϵ decayed from 0.1 to 0. For learning rates of Q and Φ value functions, α and β, we swept over the values $[0.05, 0.1, 0.2, 0.5]$. The values studied for setting C were $[5, 10, 20, 50]$. It is worth mentioning that the best value of the decay speed of ξ (*i.e.* choice of C) depend on the quality of the advice; *i.e.*, a smaller C for the bad advice was better as it decayed the effect of the adversarial bias faster while a larger C was better with the good advice as it slowed down the decay. The best values were used, according to the AUC of each line. The learning parameters are reported in Table 4.1.

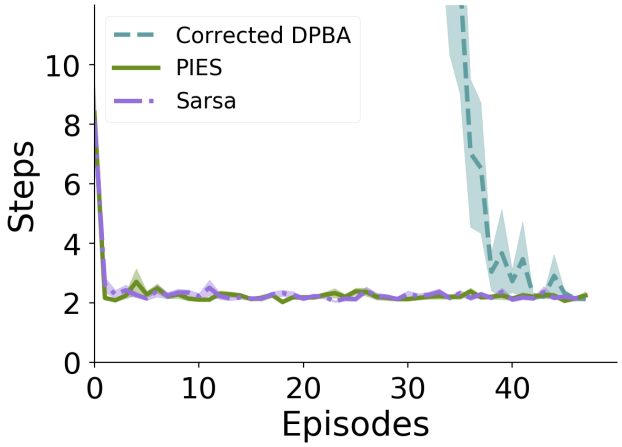
As expected, with PIES the agent was able to find the optimal policy

Table 4.1: Parameters values for Figure 4.1

Agent	α	β	C
Sarsa	0.05	-	-
corrected DPBA, good advice	0.2	0.1	-
corrected DPBA, bad advice	0.05	0.2	-
PIES, good advice	0.05	0.2	50
PIES, bad advice	0.1	0.2	5



(a) Advice from the bad expert



(b) Advice from the good expert

Figure 4.1: The y-axis shows the number of steps taken to finish each episode in the toy example. The figures compare PIES with the corrected DPBA and a Sarsa learner without shaping when the advice is a) bad and b) good. Shaded areas correspond to the standard error over 50 runs.

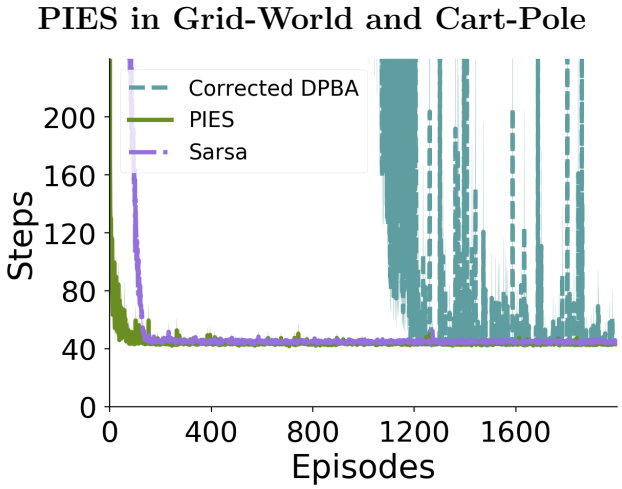
even with the bad expert. As discussed in Section 3.3, the corrected DPBA behaved similarly to PIES with the adversarial advice. In the case of the beneficial advice, PIES enabled the agent to learn the task faster. The speed-up, though, is not remarkable in the toy problem, as the simple learner is also able to learn in very few episodes. Unlike PIES, the corrected DPBA as discussed in Section 3.4 was not able to accelerate learning with the good advice.

Figure 4.2 better demonstrates how PIES boosts the performance of the agent learning with good advice in two more complex tasks: the grid-world and the cart-pole domains which were described in Section 2.2.2. The results for the grid-world task are depicted in Figure 4.2 (a). For this task the learner used Sarsa(0) with $\gamma = 0.99$. Φ and Q were initialized to 0 and the agent selected actions according to an ϵ-greedy policy with $\epsilon = 0.1$. The learning rates of Q and Φ value functions, α and β, were chosen as the best values over $[0.05, 0.1, 0.2, 0.5]$. In the cart-pole task for Figure 4.2 (b), learning was done with linear function approximation via Sarsa(λ) and a tile-coded feature representation. The weights for Q and Φ were initialized uniformly at random between 0 and 0.001. For tile-coding, we reused the parameters setting from Figure 3.1 (b). λ was set to 0.9 and γ to 1. To choose the best learning rates of Q and Φ value, α and β, the values $[0.001, 0.002, 0.01, 0.1, 0.2]$ were swept over. For both tasks, the values studied for setting C were $[50, 100, 200, 300]$. As before, the best parameter values according to the AUC of each line for Figures 4.2 (a) and 4.2 (b), and are reported in Tables 4.2 and 4.3, respectively.

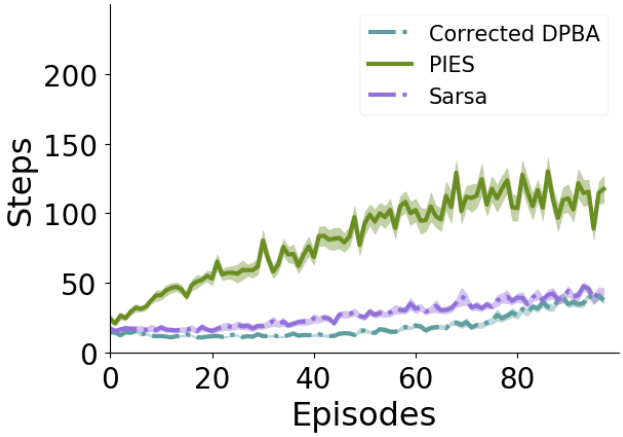
Table 4.2: Parameters values for Figure 4.2a

Agent	α	β	C
Sarsa	0.05	-	-
corrected DPBA	0.1	0.01	-
PIES	0.05	0.5	100

Just like before, in the cart-pole task's plot the higher lines indicate better performance whereas for the grid-world the lower the lines the better. PIES



(a) Grid-world



(b) Cart-pole

Figure 4.2: The length of each episode as the number of steps in (a) the grid-world, and (b) the cart-pole domain. The plot depicts PIES versus the corrected DPBA and a Sarsa learner without shaping. Shaded areas correspond to the standard error over (a) 50 and (b) 30 runs.

Table 4.3: Parameters values for Figure 4.2b

Agent	α	β	C
Sarsa	0.1	-	-
corrected DPBA	0.02	0.1	-
PIES	0.2	0.5	200

correctly used the good advice in both domains and improved learning over the Sarsa learner, without changing the optimal policy (*i.e.*, PIES approached the optimal behaviour with a higher speed compared to the Sarsa learner). PIES

performed better than the corrected DPBA as expected, since the corrected DPBA is not capable of accelerating the learner with good advice.

PIES is a reliable alternative for DPBA when we have an arbitrary form of advice, regardless of the quality of the advice. As shown in the experiments, PIES satisfies all three desired criteria.

Chapter 5

Related Work and Conclusion

In the final chapter, we discuss briefly some related work with common interests from this thesis. We then summarize the contributions of the thesis.

5.1 Related Work

The most similar line of work to our algorithm, PIES, is the TAMER framework [7]–[9]. TAMER aims to improve RL agents' learning with the help of interactive human advice [7]–[9]. With TAMER the agent models the human reward, H, with supervised learning, and all the while acts to maximize the expected immediate modeled reward, $\hat{H}$. In addition to TAMER, TAMER+RL [10] also takes the environmental reward into consideration. For combining the MDP's reward with human reinforcements, Knox and Stone [10] used a pre-trained TAMER agent with a learned model of the immediate human reward. They proposed eight different techniques for combining $\hat{H}$ with MDP's reward. Three of the eight methods use either reward or value function augmentation with $\hat{H}$ which are the most relevant methods to PIES: method one augments the reward function with $\hat{H}$, method four augments the value function, and method six augments the value function only during the action selection (similar to adding Φ to Q in PIES). In all of the mentioned methods, $\hat{H}$ is weighted by an annealing factor, decaying over time (which plays a similar role as β in PIES). While TAMER+RL uses a pre-learned model for the human reward (learned offline), PIES does not need a prior phase of interacting with the environment for approximating Φ. Moreover, $\hat{H}$ is esti-

mating the immediate human advice, ignoring the long-term effect of actions, while PIES estimates the expected return of the advice. We should mention that Knox and Stone [10] used PBRS in their eighth method, substituting the static potential function with $\max_a H(s, a)$. Since this potential function is static and depends only on states, their eighth method is not relevant to PIES. Interestingly, the results from TAMER+RL established that the sixth method, which resembles PIES the most, performed the best amongst all of the techniques investigated.

Further extending TAMER+RL, Knox and Stone [11] later provided a more thorough empirical study for the four-most successful methods out of the eight and also adjusted the framework to learn from human advice and the MDP's reward simultaneously (unlike the previous work that learns $\hat{H}$ beforehand); however, the framework still remains myopic. Once again, their empirical results confirmed that using $\hat{H}$ in action biasing (analogous to PIES) was the best method (along with the action control method) among the top-four methods. In other words, based on the observed evidence, they concluded that the less directly the advice affects the Q update the better. As opposed to Q, the more directly the advice affects the action selection the better.

A prior work to TAMER, Heuristically Accelerated Q–Learning (HAQL) by Bianchi, Ribeiro, and Costa [1], suggested a similar framework as PIES to incorporate a heuristic function in order to accelerate learning. The proposed heuristic function dynamically changes with time through an update rule, dependent on the current estimate of Q value function. They also mention a weight variable (analogous to ξ) to influence the effect of the heuristic on the policy. They showed that under certain assumptions, HAQL's Q value function converges to the optimal value function and thus it preserves the optimal policy. However there was no explicit discussion on the role of the heuristic function weight variable for policy invariance. Although PIES uses a similar approach to bias the policy toward the advice, instead of a heuristic function, PIES learns a value function from the advice (independent of the Q value function).

There is another line of research dedicated to approximating a good poten-

tial function; for example by solving an abstract MDP and computing its final value function [12], or by learning a more generalized value function of the MDP with state aggregation [4]. Although this line of work has some common interests with PIES, it does not consider an external source of advice in the problem setting.

5.2 Conclusion

This thesis exposed a flaw in DPBA, a previously published algorithm with the aim of shaping the reward function with arbitrary advice without changing the optimal policy. We used empirical and theoretical arguments to show that DPBA is not policy invariant, a key criterion for ideal reward shaping. Further, we derived the corrected DPBA algorithm with a corrected bias component. However, based on our empirical results the corrected algorithm fails to improve learning when leveraging useful advice resulting in a failure to satisfy the speed-up criterion. To overcome these problems, we proposed a simple approach, called PIES. We show theoretically and empirically that it guarantees the convergence to the optimal policy for the original MDP, agnostic to the quality of the arbitrary advice while it successfully speeds up learning from good advice. Therefore PIES satisfies all of the goals for an ideal reward shaping method.

References

[1] R. A. Bianchi, C. H. Ribeiro, and A. H. Costa, "Heuristically accelerated Q–learning: A new approach to speed up reinforcement learning," in *Brazilian Symposium on Artificial Intelligence*, Springer, 2004, pp. 245–254. 30

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016. 13

[3] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 433–440. 10

[4] M. Grześ and D. Kudenko, "Online learning of shaping rewards in reinforcement learning," *Neural Networks*, vol. 23, no. 4, pp. 541–550, 2010. 31

[5] V. Gullapalli and A. G. Barto, "Shaping as a method for accelerating reinforcement learning," in *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, 1992, pp. 554–559. 8

[6] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé, "Expressing arbitrary reward functions as potential-based advice," in *The Association for the Advancement of Artificial Intelligence*, 2015, pp. 2652–2658. 2, 10–13, 17, 20, 21

[7] W. B. Knox, I. R. Fasel, and P. Stone, "Design principles for creating human-shapable agents.," in *The Association for the Advancement of Artificial Intelligence Spring Symposium: Agents that Learn from Human Teachers*, 2009, pp. 79–86. 29

[8] W. B. Knox and P. Stone, "TAMER: Training an agent manually via evaluative reinforcement," in *2008 7th IEEE International Conference on Development and Learning*, 2008, pp. 292–297. 29

[9] ——, "Interactively shaping agents via human reinforcement: The TAMER framework," in *Proceedings of the Fifth International Conference on Knowledge Capture*, ACM, 2009, pp. 9–16. 29

[10] ——, "Combining manual feedback with subsequent MDP reward signals for reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1*, 2010, pp. 5–12. 29, 30

[11] ——, "Reinforcement learning from simultaneous human and MDP reward," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 2012, pp. 475–482. 30

[12] B. Marthi, "Automatic shaping and decomposition of reward functions," in *Proceedings of the 24th International Conference on Machine Learning*, ACM, 2007, pp. 601–608. 31

[13] D. Michie and R. A. Chambers, "Boxes: An experiment in adaptive control," *Machine Intelligence*, vol. 2, no. 2, pp. 137–152, 1968. 13

[14] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning*, vol. 99, 1999, pp. 278–287. 1, 2, 8, 9

[15] A. Y. Ng and M. I. Jordan, "Shaping and policy search in reinforcement learning," PhD thesis, University of California, Berkeley, 2003. 8

[16] OpenAI, *OpenAI Five*, `https://blog.openai.com/openai-five/`, 2018. 1

[17] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 5

[18] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," in *International Conference on Machine Learning*, vol. 98, 1998, pp. 463–471. 1, 8

[19] B. F. Skinner, "Reinforcement today," *American Psychologist*, vol. 13, no. 3, p. 94, 1958. 8

[20] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988. 7

[21] ——, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems*, 1996, pp. 1038–1044. 13

[22] ——, */incompleteideas.net/*. [Online]. Available: `http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/RLtoolkit/tiles.html`. 13, 14

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018. [Online]. Available: `http://incompleteideas.net/book/the-book-2nd.html`. 1, 7, 8

[24] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps,

and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, 2019, ISSN: 1476-4687. DOI: 10.1038/s41586-019-1724-z. [Online]. Available: https://doi.org/10.1038/s41586-019-1724-z. 1

[25] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992. 7

[26] E. Wiewiora, "Potential-based shaping and Q-value initialization are equivalent," *Journal of Artificial Intelligence Research*, vol. 19, pp. 205–208, 2003. 1, 10

[27] E. Wiewiora, G. W. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 792–799. 1, 9