REINFORCEMENT LEARNING WITH UNKNOWN REWARD FUNCTIONS

Benjamin Eysenbach*
Carnegie Mellon University
beysenba@cs.cmu.edu

Jacob Tyo*
Carnegie Mellon University
jtyo@cs.cmu.edu

Shane Gu
Google Brain
shanegu@google.com

Ruslan Salakhutdinov
Carnegie Mellon University
rsalakhu@cs.cmu.edu

Zachary Lipton
Carnegie Mellon University
zlipton@cmu.edu

Sergey Levine
University of California, Berkely
svlevine@berkely.edu

ABSTRACT

In practical reinforcement learning (RL) scenarios, algorithm designers might express uncertainty over which reward function best captures real-world desiderata. However, academic papers typically treat the reward function as either (i) exactly known, leading to the standard reinforcement learning problem, or (ii) unknown, motivating a body of work on intrinsically-motivated exploration, where agents learn the dynamics of their environment and visit diverse states, often as a pre-training step to task-specific learning. We propose a framework for reinforcement learning given *a distribution over possible reward functions*. Our contributions include derivations of the Bayes-optimal and minimax policies in this setting as well as efficient algorithms for approximating these policies.

1 INTRODUCTION

While most research on reinforcement learning (RL) addresses how to learn a policy given a Markov decision process (MDP), how to properly design reward functions in the first place is a notoriously difficult task. Well-known failures include reward hacking (Clark & Amodei, 2016; Russell & Norvig, 2016), side effects (Krakovna et al., 2018), and the difficulty of learning when rewards are sparse (Ng et al., 1999). Recent work addressing these issues includes inverse reward design (Hadfield-Menell et al., 2017) and intrinsic motivation (Chentanez et al., 2005).

In this paper, we define the *unknown reward setting*, where although a precise reward function *is not known*, a distribution over *possible reward functions* is. Our goal is to produce a policy that during testing, will achieve high reward on an *unknown* task, sampled from the task distribution. To that end, we address both the maximization of *expected* and of *worst-case* return. To motivate our setting, consider vehicle operation. While the physics of the environment would not change for different agents, their preferences might (e.g., relative priority of speed, safety, and comfort).

We contribute the following: First, we derive Bayes-optimal and minimax policies for the unknown reward setting. We present a toy example to build intuition, and then extend our results to commplex continuous-control tasks. Second, we propose efficient algorithms for approximating these policies. Recovering the minimax policy involves determining the least favorable prior, and here we make this efficient using fictitious play (Brown, 1949) and a variant of universal value function framework (Schaul et al., 2015). Third, our experiments demonstrate our method's effectiveness compared to competitive baselines.

2 POLICY SEARCH AS DECISION THEORY

Formally, we define the unknown reward setting as a distribution of MDPs $\mathcal{M}$, such that each MDP is identical but for the reward function:

$$\mathcal{M} \triangleq \{\langle \mathcal{S}, \mathcal{A}, f, r_i \rangle \mid r_i \in \mathcal{R}\} \tag{1}$$

*Equal contribution.

Algorithm 1 Bayes-Optimal Policy Learning

function LEARNBAYESPOLICY($\mathcal{M}, q$)
  Initialize policy π_θ with parameters θ
  Define $r_q(s,a) = \mathbb{E}_{r_i \sim q}[r_i(s,a)]$
  while not converged **do**
    Collect transitions with π, $\mathcal{M}$, and r_q.
    Update π_θ to maximize r_q with RL.
  return π_θ

Algorithm 2 Minimax Policy Learning

function LEARNMINIMAXPOLICY($\mathcal{M}$)
  Initialize $q_0(r)$, policy $\pi_0(a \mid s)$
  while not converged **do**
    $\bar{q}_{t-1} \leftarrow \frac{1}{t-1} \sum_{i=1}^{t-1} q_i$
    $\pi_t \leftarrow$ LEARNBAYESPOLICY$(\bar{q}_{t-1})$
    $\bar{\pi}_t \leftarrow \frac{1}{t} \sum_{i=1}^{t} \pi_i$
    $q_t \leftarrow \min_q \mathbb{E}_{\bar{\pi}_t, q}[r(s,a)]$
  $\bar{q} \leftarrow \frac{1}{t} \sum_{i=1}^{t} q_t$ ▷ Least-favorable prior
  $\pi_{\text{minimax}} \leftarrow$ LEARNBAYESPOLICY$(\bar{q})$
  return π_{minimax}

Figure 1: **Bayes-optimal and Minimax Policy Learning**: (left) we find the Bayes-optimal policy by using RL to maximize the expected reward under the prior; (right) to obtain the minimax policy, the policy and prior play a 2-player, zero-sum game. The average prior $\bar{q}_t$ converges to the least favorable prior; the Bayes-optimal policy for this prior is the minimax policy.

where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $f : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition function, $\mathcal{R}$ is a distribution of reward functions, and the prior over the MDPs is inherited from the prior over reward functions. Each reward function $r_i \sim p(r)$ is drawn from a distribution p with support $\mathcal{R}$. However, because we have multiple reward functions which cannot all be simultaneously maximized, we use the Bayes-optimal and minimax (Berger, 2013) approaches to maximize the *expected* reward and the *worst-case* reward respectively.

We consider distinct *training* and *testing* phases: while training, we have access to the task distribution, but do not know which task we will face at test time. The decision problem is to choose a policy π from a parametric class of policies $\Pi \triangleq \{\pi_\theta \mid \theta \in \Theta\}$ that will perform well during testing with respect to the Bayes-optimal or minimax notions of performance.

2.1 THE BAYES-OPTIMAL POLICY

The Bayes-optimal policy maximizes the expected reward, where expectation is taken over both the uncertainty in the reward function and any stochasticity in the policy and MDP:

$$\pi^p_{\text{Bayes}} \triangleq \arg\max_{\pi \in \Pi} B(\pi, p), \quad \text{where} \quad B(\pi, p) \triangleq \mathbb{E}_{r_i \sim p}\left[\mathbb{E}_{\substack{s_{t+1} \sim f(s_t, a_t), \\ a_t \sim \pi(a_t \mid s_t)}}[r_i(s_t)]\right]$$

By linearity of expectation, we can rewrite this problem as a standard RL objective:

$$B(\pi, p) = \mathbb{E}_{\substack{s_{t+1} \sim f(s_t, a_t), \\ a_t \sim \pi(a_t \mid s_t)}}[r_{\text{Bayes}}(s_t)], \quad \text{where} \quad r_{\text{Bayes}} \triangleq \mathbb{E}_{r_i \sim p}[r_i(s_{t+1})].$$

Thus, we have reduced the decision problem of choosing the Bayes-optimal policy into an RL problem, solvable with standard algorithms (see Alg. 1).

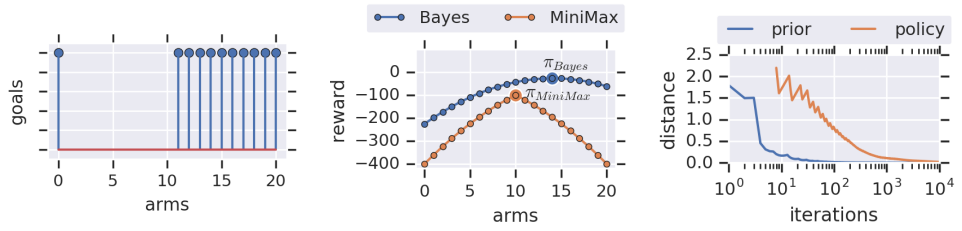
2.2 THE MINIMAX POLICY

The minimax policy maximizes the worst-case expected reward

$$\pi^*_{\text{minimax}} \triangleq \arg\max_{\pi \in \Pi} \min_{r \in \mathcal{R}} \mathbb{E}_{\substack{s_{t+1} \sim f(s_t, a_t), \\ a_t \sim \pi(a_t \mid s_t)}}[r(s_t)],$$

which is more difficult to optimize than the Bayes-optimal case. Note that simply optimizing a policy by considering at each step only its current worst-case reward function does not converge to the best worst-case policy (see Section 2.3). Instead, our approach leverages the fact that the minimax estimator is equivalent to the Bayes-optimal estimator w.r.t. the least-favorable prior:

Theorem 1 (Berger (2013)) *Let $R(\theta, \delta)$ be the risk associated with using with decision rule δ when the true parameter is θ. Assume p is a prior distribution and δ_p is the Bayes-optimal estimator w.r.t. p. If p is least favorable (i.e., $\mathbb{E}_p[R(\theta, \delta_p)] = \sup_\theta R(\theta, \delta_p)$), then δ_p is minimax.*



(a) Distribution of goals. (b) Bayes and minimax rewards. (c) Fictitious Play.

Figure 2: **Illustrative Example:** (a) A 21-armed bandit with uniform distribution over goal states. (b) The Bayes-optimal and minimax return for choosing each arm. (c) The KL-divergence between (i) the current prior and the least favorable prior, and (ii) the current policy and the minimax policy converges to zero with fictitious play.

We propose a method that recovers the *least favorable prior* distribution $q \in \mathcal{P}^d$, where $\mathcal{P}^d$ is the probability simplex with dimension equal the number of reward functions:

$$q_{\text{LFP}} \triangleq \arg\min_{q \in \mathcal{P}^d} B(\pi, q) \tag{2}$$

The minimax policy is the Bayes-optimal policy w.r.t. this least favorable prior:

$$\pi^*_{\text{minimax}} = \pi^{P_{\text{LFP}}}_{\text{Bayes}} = \arg\max_{\pi \in \Pi} B(\pi, q_{\text{LFP}}) \tag{3}$$

In Section 3, we show how to determine the least favorable prior, and we can then use Algorithm 1 to obtain the corresponding Bayes-optimal policy.

2.3 Illustrative Example

To build intuition for the difference between a Bayes-optimal and minimax policy, consider a 21-armed bandit with multiple reward functions. Let $\mathcal{G} \triangleq \{0\} \cup \{11, 12, \cdots, 20\}$ be a set of goal states, and let the reward function be the negative squared distance from a specific goal (i.e. a specific reward function for each goal state):

$$\mathcal{R} \triangleq \left\{ r_g \middle| r_i(s) = -(g-s)^2, g \in \mathcal{G} \right\}$$

Now assume a uniform prior over the reward functions, which results in the goal state distribution shown in Figure 2a. In this setting, the Bayes-optimal policy will select the arm that gives the highest reward on average, therefore selecting arm 16. While the Bayes-optimal estimator will get low reward when goal state 0 is selected, this event occurs rarely enough under the enough prior that it does not outweigh the benefit of being close to the remaining arms. The minimax policy will select the arm that maximizes the worst-case reward, therefore selecting arm 11. This aligns with our intuition: if an arm other than the middle were selected, the worst-case reward could be made lower by selecting the goal state on the furthest arm.

As a straw man, consider an *adversarial* algorithm (similar to Pinto et al. (2017)) that selects the worst-case reward function at every episode. If the initial policy chooses arm 0, then adversarial training dictates that reward function r_{20} be selected for the next episode. The policy then selects arm 20, followed by the selection of reward function r_0. The policy will cycle infinitely between choosing arms 0 and 20, never converging to the minimax policy. The algorithm that we develop in the next section avoids this cyclic behavior by historical averaging.

3 Least Favorable Priors from Nash Equilibria

In the minimax setting, we are maximizing the Bayes reward $B(\pi, q)$ w.r.t. the policy while simultaneously minimizing it w.r.t. the prior over rewards, q. We can formalize this as a two-player, zero-sum game between a *policy player*, whose strategies are to selecting policy parameters, and a *prior player*, whose strategies are to select priors over rewards. Note that we now have two sets of actions to distinguish between: (i) those due to traditional RL problem (*actions*) and (ii) those due to the decision problem (*strategies*). Strategies correspond to select policy parameters and a prior over the reward functions in our task distribution. Our goal is to recover the strategy of the prior

player, as this corresponds to the least favorable prior. The Nash Existence Theorem proves that such a stationary point always exists:

Theorem 2 (Nash (1951)) *Every two-player, zero-sum game with finite actions has a mixed strategy equilibrium point.*

We emphasize that, while the number of strategies for both players is large, the finite precision of computers dictates that the number of pure strategies (all unique combinations of neural network parameters) be finite: $|\Theta| < \infty$. While Theorem 2 proves that a stationary point exists, it fails to provide a method for finding it. Fictitious play (Brown, 1949) is a slight variant on the naïve adversarial training procedure, that says to choose the best strategy in response to the *historical average* of the opponent's strategies (Daskalakis, 2011). Robinson (1951) proved that fictitious play always converges to a Nash equilibrium:

Theorem 3 (Robinson (1951)) *If two players use fictitious play in a zero-sum game, the historical average of their strategies will converge to a Nash equilibrium in the limit.*

Moreover, Robinson (1951) suggests and Daskalakis & Pan (2014) further explain how fictitious play will converge to achieve a value within ϵ of a Nash Equilibrium within finite time.[1]

The historical average over strategies is a mixed strategy for both the policy player and the prior player. Because the set of pure strategies for the prior player, $\mathcal{P}^d$, is closed under convex combination, it is sufficient to consider pure strategies for the prior player. For the policy player, it is sufficient to keep track of the *rewards* of the policy at each iteration, rather than the *parameters*. To compute the expected reward of the historical average of policies, we can simply average the rewards from each iteration.

Algorithm 2 summarizes our algorithm derived from this analysis: we use fictitious play to obtain a least favorable prior and then compute the corresponding Bayes-optimal policy to obtain a minimax policy. We apply the algorithm to the toy problem in Figure 2a and show that it obtains the least favorable prior, and therefore the minimax policy at convergence in Figure 2c.

3.1 Efficient Implementation with Prior-Conditioned Policies

While Algorithm 2 will obtain the minimax policy, it requires solving an RL problem to convergence at each iteration. To reduce this computational expense, a naive approach of performing only a few gradient updates instead of training the policy to convergence at each step can be used. However, this is a non-stationary problem due to the changing rewards at each episode. In this case, the non-stationarity is solely caused by the prior, therefore *we can make the problem stationary by conditioning the policy (and associated value functions) on the prior*. We use a variant of the universal value function framework (Schaul et al., 2015) where we condition on prior distributions (Alg. 3). We use an off-policy algorithm, TD3 (Fujimoto et al., 2018), to leverage transitions collected at previous iterations by different policies.

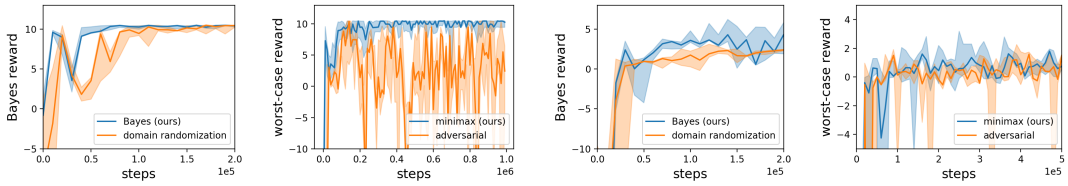
Algorithm 3 Efficient Minimax Policy Learning

Initialize prior-conditioned policy $\pi_\theta(a \mid s, q)$
Initialize prior q_0
while not converged **do**
  Collect trajectories with $\pi(a \mid s, q_t)$
  Sample (s_t, a_t, s_{t+1}) from buffer.
  Sample K reward functions $r_i \sim q_t(r)$.
  $r_t \leftarrow \frac{1}{K} \sum_{i=1}^{K} r_i(s, a)$
  Update $\pi_\theta(\cdot \mid \cdot, q_t)$ to maximize r_t with RL.
  $q_{t+1} \leftarrow \arg\min_Q \mathbb{E}_{r_i \sim q} \left[\mathbb{E}_{\pi(\cdot|\cdot, q_t)}[r_i(s)] \right]$
$\bar{q} \leftarrow \frac{1}{t} \sum_{i=1}^{t} q_t$
return $\pi_\theta(\cdot \mid \cdot, \bar{q})$ ▷ Least favorable prior.

4 Experiments

Our experiments focus on understanding when our method works and how it compares to various baselines. We apply our method to two continuous 2D navigation tasks: a point mass and a non-holonomic car (Ghosh et al., 2018) on the Cartesian plane. In both cases, the agent starts at $(-4, -4)$

[1] The time to converge grows exponentially in the number of pure strategies. In our setting, pure strategies correspond to neural networks. With 10^6 parameters, each represented as a 32-bit number, we need $O((1/\epsilon)^{32^{1e6}})$ iterations to converge.



(a) Bayes-optimal for point. (b) Minimax for point. (c) Bayes-optimal for car. (d) Minimax for car.

Figure 3: **Evaluating Bayes and minimax policy estimators**: We evaluate out method on two navigation tasks: a 2-d point in (a) and (b) and a car in (c) and (d). In both cases, we observe that our Bayes policy achieves a higher average reward than the domain randomization baseline. For point navigation, our minimax method quickly maximizes the worst case return, while the adversarial baseline never converges. Both methods struggle to learn a minimax policy on the car task.

tasked with navigating to a fixed goal location at $g = (4, 4)$. Nine zones, $z_1, \ldots, z_9$, are added between the agent's starting location and the goal location (see Fig. 4) and each zone can positively or negatively effect the return. The distribution of reward functions consists of 18 reward functions:

$$r_i^{\pm}(s_t, a_t, s_{t+1}) = d(s_t, g) - d(s_{t+1}, g) \pm 10 \cdot \mathbb{1}(s_{t+1} \in z_i)$$

where $d(s, g)$ is the $L2$ distance between the agent's center of mass and the goal.

First, we apply our method to obtain the Bayes-optimal policy for both tasks. Our prior over reward functions is uniform on the nine reward functions with negative reward bonuses. We compare our method (Alg. 1) to the *domain randomization* baseline, inspired by Tobin et al. (2017), that samples a reward function for each episode, rather than computing the expectation over reward functions. On the point navigation task, both methods achieve the maximum possible cumulative reward (Fig. 3a). On the more challenging car navigation task (Fig 3c), neither method reliably reaches the goal. Though the effect is small, our algorithm learns faster than the domain randomization baseline on both tasks, likely because of the lower variance in computing the expected reward.

Second, we obtain minimax policies for both tasks. We compare our method (Alg. 3) to the adversarial baseline. On the point navigation task, our minimax algorithm successfully obtains a minimax policy that maximizes the worst-case reward (Fig 3b). In contrast, the adversarial baseline experiences the same cyclic behavior as in Section 2.3 and fails to converge. On the car navigation task, neither method solves the task (Fig. 3d). This is expected, as the Bayes reward is an upper bound on the minimax reward (Berger, 2013).

Figure 4 plots the least-favorable prior obtained by our minimax policy on the point navigation task. For each unknown zone, we plot the probability that the least favorable prior places a negative reward bonus on that zone. We observe that the least favorable prior places large mass on the corner zones and center zone. Intuitively, the least favorable prior is setting up a barricade which prevents the policy from deviating into states where any reward function would return negative reward.

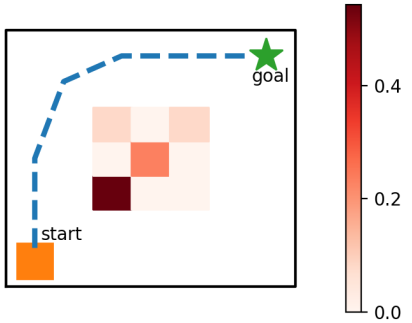


Figure 4: Least Favorable Prior for a 2D navigation task.

5 Discussion

This paper defines the unknown reward setting, derives the Bayes-optimal and minimax policies for this setting, and presents efficient algorithms for approximating them. While much prior work has looked at related problems, we defer to Appendix A for a thorough discussion. Two directions are of particular relevance and merit discussion here.

First, our work suggests that prior work on adversarial RL is actually obtaining minimax estimators with respect to unknown goals (Sukhbaatar et al., 2017) or dynamics (Pinto et al., 2017). In future work, we aim to formalize these connections to develop principled, convergent algorithms for robust RL. Second, our algorithm provides a tool to safely handle uncertainty in reward function design (Hadfield-Menell et al., 2017). For example, when a distribution of reward functions place both positive and negative reward on side effects (Krakovna et al., 2018), the minimax policy will attempt to avoid these side effects while completing the task. While AI safety is often hard to define, our work hints that concrete definitions and practical algorithms may be found in minimax analysis.

REFERENCES

Mehran Asadi and Manfred Huber. Effective control knowledge transfer through learning skill and representation hierarchies. In *IJCAI*, volume 7, pp. 2054–2059, 2007.

James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

George W Brown. Some notes on computation of games solutions. Technical report, RAND CORP SANTA MONICA CA, 1949.

Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. Sparse multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 819–827, 2014.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

Jaedeug Choi and Kee-Eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, pp. 305–313, 2012.

Jack Clark and Dario Amodei. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faultyreward-functions*, 2016.

Constantinos Daskalakis. 6.853 topics in algorithmic game theory, lecture 4. 2011.

Constantinos Daskalakis and Qinxuan Pan. A counter-example to karlin's strong conjecture for fictitious play. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 11–20. IEEE, 2014.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 720–727. ACM, 2006.

David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49 (2-3):325–346, 2002.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies. *arXiv preprint arXiv:1811.07819*, 2018.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in neural information processing systems*, pp. 6765–6774, 2017.

Victoria Krakovna, Laurent Orseau, Miljan Martic, and Shane Legg. Measuring and avoiding side effects using relative reachability. *arXiv preprint arXiv:1806.01186*, 2018.

Alessandro Lazaric. *Knowledge transfer in reinforcement learning*. PhD thesis, PhD thesis, Politecnico di Milano, 2008.

Neville Mehta, Sriraam Natarajan, Prasad Tadepalli, and Alan Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289, 2008.

John Nash. Non-cooperative games. *Annals of mathematics*, pp. 286–295, 1951.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Julia Robinson. An iterative method of solving a game. *Annals of mathematics*, pp. 296–301, 1951.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Christian R Shelton. Balancing multiple sources of reward in reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1082–1088, 2001.

Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3-4):323–339, 1992.

Matthijs Snel and Shimon Whiteson. Learning potential functions and their representations for multi-task reinforcement learning. *Autonomous agents and multi-agent systems*, 28(4):637–681, 2014.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE, 2017.

Thomas J Walsh, Lihong Li, and Michael L Littman. Transferring state abstractions between mdps. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022. ACM, 2007.

A Related Work

Krakovna et al. (2018) introduce a general definition of *side effects* based on relative state reachability, and argue that some efforts of reducing side effects can make matters worse. Shelton (2001) characterizes some of the issues in dealing with traditional scalar reward functions vs composite reward functions, and then proposes methods for balancing preferences between differing reward functions. Hadfield-Menell et al. (2017) introduces *inverse reward design* (IRD) as the problem of inferring the true objective based on the designed reward function, as a way of mitigating risk from misspecified objectives. They also provide methods for approximating the true objective. This method can be used to avoid side effects.

Schaul et al. (2015) introduce *universal value function approximators* (UVFAs) that generalize over both states and goals, and can therefore be used in any environment with any RL algorithm. They then propose a method for learning UVFAs with supervised learning or directly from experience. Sukhbaatar et al. (2017) propose *asymmetric self-play*, which is a method of allowing an agent to play a game with itself to better learn an environment. This is an unsupervised process, and allows for the agent to be familiar with an environment, before being tasked, therefore reducing the number of episodes needed to learn a new task in that environment.

Taylor & Stone (2009) provides a survey of transfer learning techniques in reinforcement learning, and here we will focus on the tasks that allow variation in the reward function. Singh (1992) and Foster & Dayan (2002) learn multiple tasks by assuming that each goal (or composite) task is composed of several elemental tasks, and then learning a set of elemental tasks that can be composed to solve each task of interest.

Solving multiple MDPs has also been approached from the representation perspective, specifically with the goal of developing a shared representation that can then be used to solve all tasks. The approach proposed by Asadi & Huber (2007) focuses on learning a more efficient state-space representation of the problem that will transfer between multiple tasks, and then learning options on the new representation. Walsh et al. (2006) use a similar approach, but rely on the learned state abstraction techniques to transfer between tasks. Another approach similar to state abstractions is to compare observations ($\langle s, a, r, s' \rangle$) tuples from previous tasks to new tasks, then select the best action from the most similar previously experienced observation. Lazaric (2008) uses this approach in an attempt to generalize experiences from learned to novel tasks, and then Calandriello et al. (2014) extend this approach to include sparse representations.

Furthermore, reward function restriction, state-space augmentation, and multiple policies have been used to ease the burden of dealing with multiple goals or tasks. If the class of considered reward functions is restricted to be linear, Mehta et al. (2008) show that a value function for the corresponding class of MDPs can be found by combining the learned value function for a finite number of MDPs with a specific reward function drawn from the restricted class. Snel & Whiteson (2014) augment a task's reward function with artificial rewards to capture important task and domain relevant knowledge. Fernández & Veloso (2006) consider when the distribution of reward functions is the set of reward functions in which only the goal state differs. A library of policies is kept and updated if a sufficiently different policy is learned. At each time step in novel tasks, the agent chooses exploit an existing policy, the current best policy, or randomly explore.

One approach to solving our problem formulation is to simply solve all MDPs, and place a constraint on the model to ensure the final policy accounts for each reward function. Several prior works have analyzed this approach. Wilson et al. (2007) consider learning in a hierarchical Bayesian RL setting, in which the prior is transferred from previously learned tasks. Choi & Kim (2012) present a nonparametric Bayesian approach to inverse reinforcement learning (IRL) with multiple reward functions by incorporating Dirichlet process mixture model into Bayesian IRL, along with a sampling algorithm to estimate the underlying reward functions. Another method of adapting a model to new tasks is that of Teh et al. (2017), where they solve all problems independently, but with the constraint that all individual policies stay close to a central shared policy. These approaches require learning against many MDPs, versus our method of redefining a new reward function and learning a single MDP (in the Bayesian case).

The game theory approaches mentioned in Section 3 have also been explored. Pinto et al. (2017) create a two-player zero-sum game by training an agent and a protagonist in conjunction, and then

framed the problem as finding the minimax equilibrium. The protagonist adversarially impedes an agent through external forces, through action, or through environmental changes, leading to better generalization. Pattanaik et al. (2018) take a similar approach, but directly optimize for the adversarial attack instead of forming a two-player game. These are similar to our setup, except that the protagonist is used to find the least favorable prior over reward functions in our case.

Adversarial training has also been well explored. Rajeswaran et al. (2016) propose a method for learning robust policies that can handle domain shift by adversarially training the policy among a range of domains. Pinto et al. (2017) use adversarial forces to disrupt systems during training, and as a result produce more robust policies that both generalize and transfer better than baselines. Fu et al. (2017) combine adversarial learning with inverse reinforcement learning and present a method that simultaneously learns the value function as well as a reward function that is invariant to the environments dynamics.

Similar to Adversarial training, another relevant and well explored domain relating to reward functions is safe RL. Garcıa & Fernández (2015) provides a survey on safe RL techniques. The *worst-case* RL considerst the minimax setting we propose, but is more general to allow for uncertainty in the system, estimators, or the policy. *Risk-Sensitive* RL focuses on controlling risk, where often times risk is defined as the variance of the return.