# Learning to Shape Rewards Using a Game of Two Partners

**David Mguni[†,1], Taher Jafferjee[1], Jianhong Wang[2], Nicolas Perez-Nieves[3], Wenbin Song[6], Feifei Tong[1], Matthew E. Taylor[4,5], Tianpei Yang[4,5], Zipeng Dai[1], Hui Chen[7], Jiangcheng Zhu[1], Kun Shao[1], Jun Wang[7], Yaodong Yang[†,8]**

[1]Huawei R&D
[2]University of Manchester, UK
[3]Imperial College London, UK
[4]University of Alberta, Edmonton, Canada
[5]Alberta Machine Intelligence Institute, Edmonton, Canada
[6]Shanghai Tech University, China
[7] University College London, UK
[8] Peking University, Beijing, China

## Abstract

Reward shaping (RS) is a powerful method in reinforcement learning (RL) for overcoming the problem of sparse or uninformative rewards. However, RS typically relies on manually engineered shaping-reward functions whose construction is time-consuming and error-prone. It also requires domain knowledge which runs contrary to the goal of autonomous learning. We introduce Reinforcement Learning Optimising Shaping Algorithm (ROSA), an automated reward shaping framework in which the shaping-reward function is constructed in a Markov game between two agents. A reward-shaping agent (Shaper) uses *switching controls* to determine which states to add shaping rewards for more efficient learning while the other agent (Controller) learns the optimal policy for the task using these shaped rewards. We prove that ROSA, which adopts existing RL algorithms, learns to construct a shaping-reward function that is beneficial to the task thus ensuring efficient convergence to high-performance policies. We demonstrate ROSA's properties in three didactic experiments and show its superior performance against state-of-the-art RS algorithms in challenging sparse reward environments.

## Introduction

Despite the notable success of RL in a variety of domains, enabling RL algorithms to learn successfully in numerous real-world tasks remains a challenge (Wang et al. 2021). A key obstacle to the success of RL algorithms is that sparse reward signals can hinder agent learning (Charlesworth and Montana 2020). In many settings of interest such as physical tasks and video games, rich informative signals of the agent's performance are not readily available (Hosu and Rebedea 2016). For example, in the video game Super Mario (Shao et al. 2019), the agent must perform sequences of hundreds of actions while receiving no rewards for it to successfully complete its task. In this setting, the infrequent feedback of the agent's performance leads to RL algorithms requiring large numbers of samples (and high expense) for solving problems (Hosu and Rebedea 2016). Therefore, there is a need for RL techniques to solve such problems efficiently.

Reward shaping (RS) is a tool to introduce additional rewards, known as *shaping rewards*, to supplement the environmental reward. These rewards can encourage exploration and insert structural knowledge in the absence of informative environment rewards thereby significantly improving learning outcomes (Devlin, Kudenko, and Grześ 2011). RS algorithms often assume hand-crafted and domain-specific shaping functions, constructed by subject matter experts, which runs contrary to the aim of autonomous learning. Moreover, poor choices of shaping rewards can *worsen* the agent's performance (Devlin and Kudenko 2011). To resolve these issues, a useful shaping reward must be obtained *autonomously*.

We develop a framework that autonomously constructs shaping rewards during learning. ROSA introduces an additional RL agent, the Shaper, that *adaptively* learns to construct shaping rewards by observing Controller , while Controller learns to solve its task. This *generates tailored shaping rewards without the need for domain knowledge or manual engineering*. These shaping rewards successfully promote effective learning, addressing this key challenge.

The resulting framework is a two-player, nonzero-sum Markov game (MG) (Shoham and Leyton-Brown 2008) — an extension of Markov decision process (MDP) that involves *two* independent learners with distinct objectives. In our framework, the two agents have distinct learning agendas but cooperate to achieve the Controller's objective. An integral component of ROSA is a novel combination of RL and *switching controls* (Mguni et al. 2022; Bayraktar and Egami 2010; Mguni 2018). This enables Shaper to quickly determine useful states to learn to add in shaping rewards (i.e., states where adding shaping rewards improves the Controller's performance) but disregard other states. In contrast Controller must learn actions for every state. This leads to the Shaper quickly finding shaping rewards that guide the Controller's learning process toward optimal trajectories (and away from suboptimal trajectories, as in Experiment 1).

This approach tackles multiple obstacles. First, a new agent (Shaper) learns while the Controller is training while avoid-

[†]Corresponding authors <davidmguni@hotmail.com> <yaodong.yang@pku.edu.cn>.

ing convergence issues. Second, unlike standard RL, the Shaper's learning process uses *switching controls*. We show successful empirical results and prove that ROSA converges.

## Related Work

**Reward Shaping** Reward Shaping (RS) adds a *shaping function* $F$ to supplement the agent's reward to boost learning. RS however has some critical limitations. First, RS does not offer a means of finding $F$. Second, poor choices of $F$ can *worsen* the agent's performance (Devlin and Kudenko 2011). Last, adding shaping rewards can change the underlying problem, therefore, generating policies that are completely irrelevant to the task (Mannion et al. 2017). In Ng et al. 1999 it was established that potential-based reward shaping (PBRS) which adds a shaping function of the form $F(s_{t+1}, s_t) = \gamma\phi(s_{t+1}) - \phi(s_t)$ preserves the optimal policy of the problem. Recent variants of PBRS include potential-based advice which defines $F$ over the state-action space (Harutyunyan et al. 2015) and approaches that include time-varying shaping functions (Devlin and Kudenko 2012). Although the last issue can be addressed using potential-based reward shaping (PBRS) (Ng, Harada, and Russell 1999), the first two issues remain (Behboudian et al. 2021).

To avoid manual engineering of $F$, useful shaping rewards must be obtained autonomously. Towards this Zou et al. 2019 introduce an RS method that adds a shaping-reward function prior which fits a distribution from data obtained over many tasks. Recently, Hu et al. 2020 use a bilevel technique to learn a scalar coefficient for an already-given shaping-reward function. Nevertheless, constructing $F$ *while training* can produce convergence issues since the reward function now changes during training (Igl et al. 2020). Moreover, while $F$ is being learned the reward can be corrupted by inappropriate signals that hinder learning.

**Curiosity-Based Reward Shaping** Curiosity-Based Reward Shaping aims to encourage the agent to explore states by rewarding the agent for novel state visitations using exploration heuristics. One approach is to use state visitation counts (Ostrovski et al. 2017). More elaborate approaches such as Burda et al.(Burda et al. 2018) introduce a measure of *state novelty* using the prediction error of features of the visited states from a random network. Pathak et al. 2017 use the prediction error of the next state from a learned dynamics model and Houthooft et al. 2016 maximise the information gain about the agent's belief of the system dynamics. In general, these methods provide no performance guarantees nor do they ensure the optimal policy of the underlying MDP is preserved. Moreover, they naively reward exploration without consideration of the environment reward. This can lead to spurious objectives being maximised (see Experiment 3 in Sec. ).

Within these two categories, closest to our work are bilevel approaches for learning the shaping function (Hu et al. 2020; Stadie, Zhang, and Ba 2020). Unlike Hu et al. 2020 whose method requires a useful shaping reward to begin with, ROSA constructs a shaping reward function from scratch leading to a fully autonomous method. Moreover, in Hu et al. 2020 and Stadie et al.2020, the agent's policy and shaping rewards are learned with *consecutive* updates. In contrast, ROSA performs these operations *concurrently* leading to a faster, more efficient procedure. Also in contrast to Hu et al. 2020 and Stadie et al. 2020, ROSA learns shaping rewards only at relevant states, this confers high computational efficiency (see Experiment 2, Sec. )). As we describe, ROSA, which successfully *learns the shaping-reward function* $F$, uses a similar form as PBRS. However, in ROSA, $F$ is augmented to include the actions of another RL agent to learn the shaping rewards online. Lastly, unlike curiosity-based methods e.g., Burda et al. 2018 and Pathak et al. 2017, our method preserves the agent's optimal policy for the task (see Experiment 3, Sec. ) and introduces intrinsic rewards that promote complex learning behaviour (see Experiment 1, Sec. ) .

## Preliminaries & Notations

The RL problem is typically formalised as a Markov decision process $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the discrete set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition probability function describing the system's dynamics, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function measuring the agent's performance, and $\gamma \in (0, 1]$ specifies the degree to which the agent's rewards are discounted (Sutton and Barto 2018). At time $t$ the system is in state $s_t \in \mathcal{S}$ and the agent must choose an action $a_t \in \mathcal{A}$ which transitions the system to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and produces a reward $R(s_t, a_t)$. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a distribution over state-action pairs where $\pi(a|s)$ is the probability of selecting action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. The agent's goal is to find a policy $\pi^\star \in \Pi$ that maximises its expected returns given by: $v^\pi(s) = \mathbb{E}[\sum_{t=0}^\infty \gamma^t R(s_t, a_t)|a_t \sim \pi(\cdot|s_t)]$ where $\Pi$ is the agent's policy set. We denote this MDP by $\mathfrak{M}$.

**Two-player Markov games** A two-player Markov game (MG) is an augmented MDP involving two agent that simultaneously take actions over many rounds (Shoham and Leyton-Brown 2008). In the classical MG framework, each agent's rewards and the system dynamics are now influenced by the actions of *both* agents. Therefore, each agent $i \in \{1, 2\}$ has its reward function $R_i : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{R}$ and action set $\mathcal{A}_i$ and its goal is to maximise its *own* expected returns. The system dynamics, now influenced by both agents, are described by a transition probability $P : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S} \to [0, 1]$. As we discuss in the next section, ROSA induces a specific MG in which the dynamics are influenced by *only* Controller.

**Reward shaping** Reward shaping (RS) seeks to promote more efficient learning by inserting a (state-dependent) shaping reward function $F$. Denote by $\tilde{v}$ the objective function that contains a shaping reward function $F$ and by $\tilde{\pi} \in \tilde{\Pi}$ the corresponding policy i.e., $v^{\tilde{\pi}}(s) = \mathbb{E}[\sum_{t=0}^\infty \gamma^t (R(s_t, a_t) + F(\cdot))|a_t \sim \tilde{\pi}(\cdot|s_t)]$. Let us also denote by $v^{\pi_k}$ the expected return after $k$ learning steps, the goal for RS can be stated as inserting a shaping reward function $F$ for any state $s \in \mathcal{S}$:

**C.1.** $v^{\tilde{\pi}_m}(s) \geq v^{\pi_m}(s)$ for any $m \geq N$,

**C.2.** $\arg\max_{\pi\in\Pi} \tilde{v}^\pi(s) \equiv \arg\max_{\pi\in\Pi} v^\pi(s)$,

where $N$ is some finite integer.

Condition **C.1** ensures that RS produces a performance improvement (weakly) during the learning process i.e., RS induces more efficient learning and does not degrade performance (note that both value functions measure the expected return from the environment only). Lastly, Condition **C.2** ensures that RS preserves the optimal policy.[1]

Poor choices of $F$ *hinder* learning (Devlin and Kudenko 2011) in violation of (ii), and therefore RS methods generally rely on hand-crafted shaping-reward functions that are constructed using domain knowledge (whenever available). In the absence of a useful shaping-reward function $F$, the challenge is to *learn* a shaping-reward function that leads to more efficient learning while preserving the optimal policy. The problem therefore can be stated as finding a function $F$ such that (i) - (iii) hold. Determining this function is a significant challenge; poor choices can hinder the learning process, moreover attempting to learn the shaping-function while learning the RL agent's policy presents convergence issues given the two concurrent learning processes (Zinkevich, Greenwald, and Littman 2006). Another issue is that using a hyperparameter optimisation procedure to find $F$ directly does not make use of information generated by intermediate state-action-reward tuples of the RL problem which can help to guide the optimisation.

## Our Framework

We now describe the problem setting, details of our framework, and how it learns the shaping-reward function. We then describe Controller's and Shaper's objectives. We also describe the switching control mechanism used by the Shaper and the learning process for both agents.

The Shaper's goal is to construct shaping rewards to guide the Controller towards quickly learning $\pi^\star$. To do this, the Shaper learns how to choose the values of a shaping-reward at each state. Simultaneously, Controller performs actions to maximise its rewards using its own policy. Crucially, the two agents tackle distinct but complementary set problems. The problem for Controller is to learn to solve the task by finding its optimal policy, the problem for the Shaper is to learn how to add shaping rewards to aid the Controller. The objective for the Controller is given by:

$$\tilde{v}^{\pi,\pi^2}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t)+\hat{F}(a_t^2,a_{t-1}^2)\right)\Big|s=s_0\right],$$

where $a_t \sim \pi$ is the Controller's action, $\hat{F}$ is the shaping-reward function which is given by $\hat{F}(a_t^2,a_{t-1}^2) \equiv a_t^2 - \gamma^{-1}a_{t-1}^2$, $a_t^2$ : is chosen by the Shaper (and $a_t^2 \equiv 0, \forall t < 0$) using the policy $\pi^2 : \mathcal{S}\times\mathcal{A}_2 \to [0,1]$ where $\mathcal{A}_2$ is the action set for the Shaper. Note that the shaping reward is state dependent since the Shaper's policy is contingent on the state. The set $\mathcal{A}_2$ is a subset of $\mathbb{R}^p$ and can therefore be for example a set of integers $\{1,\ldots,K\}$ for some $K \geq 1$. With this, the Shaper constructs a shaping-reward based on the agent's environment interaction, therefore the shaping reward is tailored for the specific setting. The transition probability $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ takes the state and *only* the Controller's actions as inputs. Formally, the MG is defined by a tuple $\mathcal{G} = \langle \mathcal{N},\mathcal{S},\mathcal{A},\mathcal{A}_2,P,\hat{R}_1,\hat{R}_2,\gamma\rangle$ where the new elements are $\mathcal{N} = \{1,2\}$ which is the set of agents, $\hat{R}_1 := R + \hat{F}$ is the new Controller reward function which now contains a shaping reward $\hat{F}$, the function $\hat{R}_2 : \mathcal{S}\times\mathcal{A}\times\mathcal{A}_2 \to \mathbb{R}$ is the one-step reward for the Shaper (we give the details of this function later).

As the Controller's policy can be learned using any RL method, ROSA easily adopts any existing RL algorithm for the Controller. Note that unlike reward-shaping methods e.g. (Ng, Harada, and Russell 1999), our shaping reward function $F$ consists of actions $a^2$ which are chosen by the Shaper which enables a shaping-reward function to be learned online. We later prove a policy invariance result (Prop. 1) analogous to that in (Ng, Harada, and Russell 1999) and show ROSA preserves the optimal policy of the agent's underlying MDP.

### Switching Controls

So far the Shaper's problem involves learning to construct shaping rewards at *every* state including those that are irrelevant for guiding Controller. To increase the (computational) efficiency of the Shaper's learning process, we now introduce a form of policies known as *switching controls*. Switching controls enable Shaper to decide at which states to learn the value of shaping rewards it would like to add. Therefore, now Shaper is tasked with learning how to shape Controller's rewards *only* at states that are important for guiding Controller to its optimal policy. This enables Shaper to quickly determine its optimal policy[2] $\pi^2$ for only the relevant states unlike Controller whose policy must learned for all states. Now at each state Shaper first makes a *binary decision* to decide to *switch on* its shaping reward $F$ for the Controller. This leads to an MG in which, unlike classical MGs, the Shaper now uses *switching controls* to perform its actions.

We now describe how at each state both the decision to activate a shaping reward and their magnitudes are determined. Recall that $a_t^2 \sim \pi^2$ determines the shaping reward through $F$. At any $s_t$, the decision to turn on the shaping reward function $F$ is decided by a (categorical) policy $\mathfrak{g}_2 : \mathcal{S} \to \{0,1\}$. Therefore, $\mathfrak{g}_2$ determines whether the Shaper policy $\pi^2$ should be used to introduce a shaping reward $F(a_t^2,a_{t-1}^2), a_t^2 \sim \pi^2$. We denote by $\{\tau_k\}$ the times that a switch takes place, for example, if the switch is first turned on at state $s_5$ then turned off at $s_7$, then $\tau_1 = 5$ and $\tau_2 = 7$. Recalling the role of $\mathfrak{g}_2$, the switching times obey the expression $\tau_k = \inf\{t > \tau_{k-1}|s_t \in \mathcal{S}, \mathfrak{g}_2(s_t) = 1\}$ and are therefore ***rules** that depend on the state.*. The termination times $\{\tau_{2k-1}\}$ occur according to some external (probabilistic) rule i.e., if at state $s_t$ the shaping reward is active, then the shaping reward terminates at state $s_{t+1}$ with probability $p \in ]0,1]$. Hence, by learning an optimal $\mathfrak{g}_2$, the Shaper learns the best states to activate $F$.

[1]For sufficiently complex tasks, a key aim of an RS function is to enable the agent to acquire rewards more quickly provided the agent must learn an improvement on its initial policy that is to say $v^{\tilde{\pi}_n}(s) > v^{\pi_n}(s)$ for all $n \geq N$; whenever $\max_{\pi\in\Pi} v^{\pi}(s) > v^{\pi_0}(s)$. However, such a condition cannot be guaranteed for all RL tasks since it is easy to construct a trivial example in which RS is not required and the condition would not hold.

[2]i.e., a policy that maximises its own objective.

We now describe the new Controller objective. To describe the presence of shaping rewards at times $\{\tau_{2k}\}_{k>0}$ for notational convenience, we introduce a switch $I_t$ for the shaping rewards which takes values 0 or 1 and obeys $I_{\tau_{k+1}} = 1 - I_{\tau_k}$ (note that the indices are the times $\{\tau_k\}$ not the time steps $t = 0, 1, \ldots$) and $I_t \equiv 0, \forall t \leq 0$. With this, the new Controller objective is:

$$\tilde{v}^{\pi,\pi^2}(s_0, I_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left\{R(s_t, a_t) + \hat{F}(a_t^2, a_{t-1}^2) I_t\right\}\right].$$

**Summary of events:**

At a time $t \in 0, 1 \ldots$

- Both agents make an observation of the state $s_t \in \mathcal{S}$.
- Controller takes an action $a_t$ sampled from its policy $\pi$.
- Shaper decides whether or not to activate the shaping reward using $\mathfrak{g}_2 : \mathcal{S} \rightarrow \{0, 1\}$.
- If $\mathfrak{g}_2(s_t) = 0$:
  - The switch is not activated ($I_t = 0$). Controller receives a reward $r \sim R(s_t, a_t)$ and the system transitions to the next state $s_{t+1}$.
- If $\mathfrak{g}_2(s_t) = 1$:
  - Shaper takes an action $a_t^2$ sampled from its policy $\pi^2$.
  - The switch is activated ($I_t = 1$), Controller receives a reward $R(s_t, a_t) + \hat{F}(a_t^2, a_{t-1}^2) \times 1$ and the system transitions to the next state $s_{t+1}$.

We set $\tau_k \equiv 0 \forall k \leq 0$ and $a_k^2 \equiv 0 \quad \forall k \leq 0$ and lastly $a_{\tau_k}^2 \equiv 0, \forall k \in \mathbb{N}$ ($a_{\tau_k+1}^2, \ldots, a_{\tau_{k+1}-1}^2$ remain non-zero). The first two conditions ensure the objective is well-defined while the last condition which can be easily ensured, is used in the proof of Prop. 1 which guarantees that the optimal policy of the MDP $\mathcal{M}$ is preserved. Lastly, in what follows we use the shorthand $I(t) \equiv I_t$.

## The Shaper's Objective

The goal of the Shaper is to guide Controller to efficiently learn to maximize its own objective. The shaping reward $F$ is activated by switches controlled by the Shaper. To induce Shaper to selectively choose when to switch on the shaping reward, each switch activation incurs a fixed cost for the Shaper. This ensures that the gain for the Shaper for encouraging Controller to visit a given set of states is sufficiently high to merit learning optimal shaping reward magnitudes. Given these remarks the Shaper's objective is

$$v_2^{\pi,\pi^2}(s_0, I_0) = \mathbb{E}_{\pi,\pi^2}\left[\sum_{t=0}^{\infty} \gamma^t (\hat{R}_1 - \sum_{k\geq 1}^{\infty} \delta_{\tau_{2k-1}}^t + L(s_t))\right]$$

where $\delta_{\tau_{2k-1}}^t$ is the Kronecker-delta function which introduces a cost for each switch, is 1 whenever $t = \tau_{2k-1}$ and 0 otherwise (this restricts the costs to only the points at which the shaping reward is activated). The term $L$ is a Shaper *bonus reward* for when the Controller visits an infrequently visited state and tends to 0 as the state is revisited.

The objective encodes the Shaper's agenda, namely to maximise the expected return.[3] Therefore, using its shaping rewards, the Shaper seeks to guide Controller towards optimal trajectories (potentially away from suboptimal trajectories, c.f. Experiment 1) and enable Controller to learn faster (c.f. Cartpole experiment in Sec. ). With this, the Shaper constructs a shaping-reward function that supports the Controller's learning which is tailored for the specific setting. This avoids inserting hand-designed exploration heuristics into the Controller's objective as in curiosity-based methods (Burda et al. 2018; Pathak et al. 2017) and classical reward shaping (Ng, Harada, and Russell 1999). We later prove that with this objective, the Shaper's optimal policy maximises Controller's (extrinsic) return (Prop. 1). Additionally, we show that the framework preserves the optimal policy of $\mathfrak{M}$.

## Discussion on Shaper Bonus Term $L$

For this there are various possibilities e.g. model prediction error (Stadie, Levine, and Abbeel 2015), count-based exploration bonus (Strehl and Littman 2008). We later show our method performs well regardless of the choice of bonus rewards and outperforms RL methods in which these bonuses are added to the agent's objective directly (see Sec. ).

## Discussion on Computational Aspect

The switching control mechanism results in a framework in which the problem facing the Shaper has a markedly reduced decision space in comparison to the Controller's problem (though both share the same experiences). Crucially, the Shaper must compute optimal shaping rewards at only a subset of states which are chosen by $\mathfrak{g}_2$. Moreover, the decision space for the switching policy $\mathfrak{g}_2$ is $\mathcal{S} \times \{0, 1\}$ i.e at each state it makes a binary decision. Consequently, the learning process for $\mathfrak{g}_2$ is much quicker than the Controller's policy which must optimise over a decision space which is $|\mathcal{S}||\mathcal{A}|$ (choosing an action from its action space at every state). This results in the Shaper rapidly learning its optimal policies (relative to the Controller) in turn, enabling the Shaper to guide the Controller towards its optimal policy during its learning phase. Additionally, in our experiments, we chose the size of the action set for the Shaper, $\mathcal{A}_2$ to be a singleton resulting in a decision space of size $|\mathcal{S}| \times \{0, 1\}$ for the entire problem facing the Shaper. We later show that this choice leads to improved performance while removing the free choice of the dimensionality of the Shaper's action set. Lastly, we later prove that the optimal policy for the Shaper maximises the Controller's objective (Prop. 1).

## The Overall Learning Procedure

The game $\mathcal{G}$ is solved using our multi-agent RL algorithm (ROSA). In the next section, we show the convergence properties of ROSA. The full code is in Sec. **??** of the Appendix. The ROSA algorithm consists of two independent procedures: Controller learns its own policy while Shaper learns which states to perform a switch and the shaping reward magnitudes. In our implementation, we used proximal policy optimization

[3]Hence $\hat{R}_2(s_t, a_t, a_t^2) \equiv R(s_t, a_t) + \hat{F}(a_t^2, a_{t-1}^2) \cdot I_t - \sum_{k\geq 1}^{\infty} \delta_{\tau_{2k-1}}^t + L(s_t)$.

(PPO) (Schulman et al. 2017) as the learning algorithm for all policies: Controller's policy, switching control policy, and the reward magnitude policy. We demonstrated ROSA with various Shaper $L$ terms, the first is RND (Burda et al. 2018) in which $L$ takes the form $L(s_t) := \|\hat{h}(s_t) - h(s_t)\|_2^2$ where $h$ is a randomly initialised, fixed target network while $\hat{h}$ is the predictor network that seeks to approximate the target network. Secondly, to demonstrate the flexibility of ROSA to perform well with even a rudimentary bonus term, we use a simple count-based term for $L$, which counts the number of times a state has been visited (see Sec. ). The action set of the Shaper is thus $\mathcal{A}_2 := \{0, 1, ..., m\}$ where each element is an element of $\mathbb{N}$, and $\pi_2$ is a MLP $\pi_2 : \mathbb{R}^d \mapsto \mathbb{R}^m$. Precise details are in the Supplementary Material, Section 8.

## Convergence and Optimality of ROSA

The ROSA framework enables the Shaper to learn a shaping-reward function to assist the Controller when learning a (near-)optimal policy. The interaction between the two RL agents induces two concurrent learning processes, potentially raising convergence issues (Zinkevich, Greenwald, and Littman 2006). We now show that ROSA converges and that the performance of the resulting policy is similar to solving $\mathfrak{M}$ directly. To achieve this, we first study the stable point solutions of $\mathcal{G}$. Unlike MDPs, the existence of stable point solutions in Markov policies is not guaranteed for MGs (Blackwell and Ferguson 1968) and is rarely computable.[4] MGs also often have multiple stable points that can be inefficient (Mguni et al. 2019); in $\mathcal{G}$, the outcome of such stable point profiles may be a poor performing Controller policy. To ensure the framework is useful, we must verify that the solution of $\mathcal{G}$ corresponds to $\mathfrak{M}$. We address the following challenges:

**1.** ROSA preserves the optimal policy of $\mathfrak{M}$.

**2.** A stable point of the game $\mathcal{G}$ in Markov policies exists.

**3.** ROSA converges to the stable point solution of $\mathcal{G}$.

**4.** The convergence point of ROSA yields a payoff that is (weakly) greater than that from solving $\mathfrak{M}$ directly.

In proving 1–4 we deduce the following:

**Theorem 1.** *ROSA ensures conditions C.1 and C.2.*

Proofs are deferred to the Appendix.

We now give our first result that shows the solution to $\mathfrak{M}$ is preserved under the influence of the Shaper:

**Proposition 1.** *The following statements hold $\forall s \in \mathcal{S}$:*

*i)* $\arg\max_{\pi \in \Pi} \tilde{v}^{\pi,\pi^2}(s) = \arg\max_{\pi \in \Pi} v^{\pi}(s), \forall \pi^2 \in \Pi^2$,

*ii) The Shaper's optimal policy maximises $v^{\pi}(s)$.*

Recall, $v^{\pi}$ denotes the Controller's expected return without the influence of the Shaper. Result (i) therefore says that the Controller's problem is preserved under the influence of the Shaper. Moreover the (expected) total return received by the Controller is that from the environment. Result (ii) establishes that the Shaper's optimal policy induces Shaper to maximise Controller's extrinsic total return.

[4]Exceptions are *team* and *zero-sum* MGs (Yang and Wang 2020).

The result comes from a careful adaptation of the policy invariance result (Ng, Harada, and Russell 1999) to our multi-agent switching control framework, where the shaping reward is no longer added at all states. Building on Prop. 1, we find:

**Corollary 1.** *ROSA preserves the MDP $\mathfrak{M}$. In particular, let $(\hat{\pi}^1, \hat{\pi}^2)$ be a stable point policy profile[5] of the MG induced by ROSA $\mathcal{G}$. Then, $\hat{\pi}^1$ is a solution to the MDP, $\mathfrak{M}$.*

Hence, introducing the Shaper does not alter the solution.

We next show that the solution of $\mathcal{G}$ can be computed as a limit point of a sequence of Bellman operations. We use this to show the convergence of ROSA. We define a *projection* $\mathcal{P}$ on a function $\Lambda$ by: $\mathcal{P}\Lambda := \arg\min_{\bar{\Lambda} \in \{\Psi r | r \in \mathbb{R}^p\}} \left\|\bar{\Lambda} - \Lambda\right\|$:

**Theorem 2.** ***i)*** *Let $V : \mathcal{S} \times \mathbb{N} \to \mathbb{R}$ then the game $\mathcal{G}$ has a stable point which is a given by $\lim_{k \to \infty} T^k V^{\boldsymbol{\pi}} = \sup_{\hat{\boldsymbol{\pi}} \in \boldsymbol{\Pi}} V^{\hat{\boldsymbol{\pi}}} = V^{\boldsymbol{\pi}^\star}$, where $\hat{\boldsymbol{\pi}}$ is a stable policy profile for the MG, $\mathcal{G}$ and $T$ is the Bellman operator of $\mathcal{G}$.*

***ii)*** *ROSA converges to the stable point of $\mathcal{G}$. Moreover, given a set of linearly independent basis functions $\Psi = \{\psi_1, \ldots, \psi_p\}$ with $\psi_k \in L_2, \forall k$, ROSA converges to a limit point $r^\star \in \mathbb{R}^p$ that is the unique solution to $\mathcal{P}\mathfrak{F}(\Psi r^\star) = \Psi r^\star$, where $\mathfrak{F}$ is defined by: $\mathfrak{F}\Lambda := \hat{R}_1 + \gamma P \max\{\mathcal{M}\Lambda, \Lambda\}$ where $\mathcal{M}$ is defined by $\mathcal{M}^{\pi,\pi^2} v(s, \cdot) := \hat{R}_1 - 1 + \gamma \sum_{s' \in \mathcal{S}} P(s'; a_{\tau_k}, s) v(s', \cdot) | a_{\tau_k} \sim \pi^2$ and $r^\star$ satisfies: $\|\Psi r^\star - Q^\star\| \leq (1 - \gamma^2)^{-1/2} \|\mathcal{P}Q^\star - Q^\star\|$.*

Part i) of the theorem proves the system in which the Shaper and Controller jointly learn has a stable point and is the limit of a dynamic programming procedure. Crucially (by Corollary 1), the limit point corresponds to the solution of the MDP $\mathcal{M}$. This is proven by showing that $\mathcal{G}$ has a dual representation as an MDP whose solution corresponds to the stable point of the MG. This then enables a distributed Q-learning method (Bertsekas 2012) to tractably solve $\mathcal{G}$.

Part ii) establishes the solution to $\mathcal{G}$ can be computed using ROSA. This means that the Shaper converges to a shaping-reward function and (by Prop. 1) the Controller learns the optimal value function for $\mathcal{M}$. The result also establishes the convergence of ROSA to the solution using (linear) function approximators and bounds the approximation error by the smallest error achievable (given the basis functions).

Introducing poor shaping rewards can potentially worsen overall performance. We now prove ROSA introduces shaping rewards that yield higher total environment returns for the Controller, as compared to solving $\mathfrak{M}$ directly.

**Proposition 2.** *There exists some finite integer $N$ such that $v^{\tilde{\pi}_m}(s) \geq v^{\pi_m}(s), \ \forall s \in \mathcal{S}$ for any $m \geq N$, where $\tilde{\pi}_m$ and $\pi_m$ are the respective Controller policies after the $m^{th}$ learning iteration with and without the Shaper's influence.*

Note that Prop. 2 implies $v^{\tilde{\pi}}(s) \geq v^{\pi}(s), \ \forall s \in \mathcal{S}$. Prop. 2 shows that the Shaper improves outcomes for the Controller. Additionally, unlike reward shaping methods in general, the shaping rewards generated by the Shaper *never* lead to a reduction to the total (environmental) return for Controller (compared to the total return without $F$).

[5]I.e. a Markov perfect equilibrium (Fudenberg and Tirole 1991).

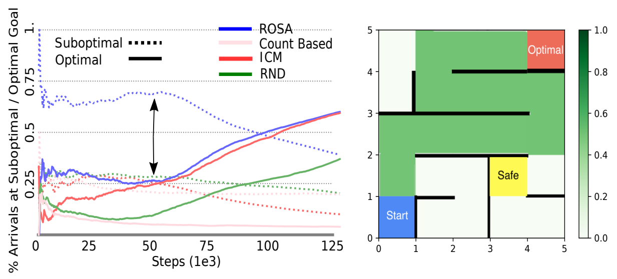

Figure 1: *Left.* Proportion of optimal and suboptimal goal arrivals. ROSA has a marked inflection (arrow) where arrivals at the sub-optimal goal decrease and arrivals at the optimal goal increase. Shaper has learned to guide Controller to forgo the suboptimal goal in favour of the optimal one. *Right.* Heatmap showing where ROSA adds rewards.

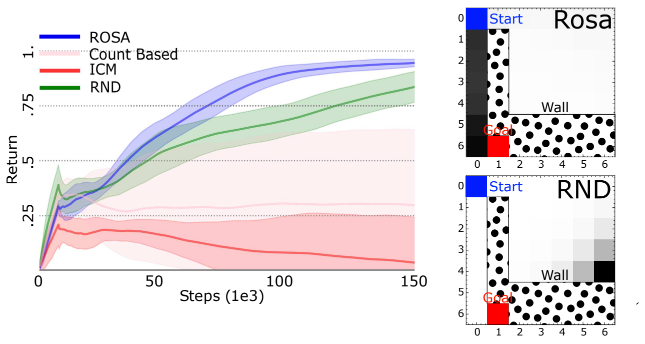

Figure 3: Red-Herring Maze. Ignoring non-beneficial shaping reward. *Left.* Learning curves. *Right.* Heatmap of added shaping rewards. ROSA ignores the RHS of the maze, while RND incorrectly adds unuseful shaping rewards there.

**Note:** *Prop. 2 compares the environmental (extrinsic) rewards accrued by the Controller. Prop. 2 therefore shows Shaper induces a Controller policy that leads to a (weakly) higher expected return from the environment.*

## Experiments

We performed a series of experiments to test if ROSA **(1)** learns beneficial shaping-reward functions **(2)** decomposes complex tasks, and **(3)** tailors shaping rewards to encourage the Controller to capture environment rewards (as opposed to merely pursuing novelty). We compared ROSA's performance to RND (Burda et al. 2018), ICM (Pathak et al. 2017), LIRPG (Zheng, Oh, and Singh 2018), BiPaRS-IMGL (Hu et al. 2020)[6] and vanilla PPO (Schulman et al. 2017). We then compared performances on performance benchmarks including Sparse Cartpole, Gravitar, Solaris, and Super Mario.

### Didactical Examples

**Beneficial shaping reward.** ROSA is able to learn a shaping-reward function that leads to improved Controller performance. In particular, it is able to learn to shape rewards that

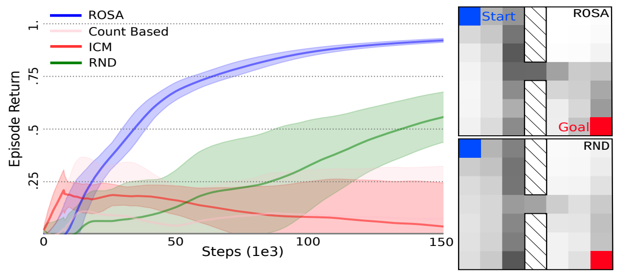

Figure 2: Discovering subgoals on Subgoal Maze. *Left.* Learning curves. *Right.* Heatmap of shaping rewards.

encourage the RL agent to avoid suboptimal — but easy to learn — policies in favour of policies that attain the maximal return. To demonstrate this, we designed a Maze environment with two terminal states: a suboptimal goal state that yields a reward of 0.5 and an optimal goal state which yields a reward of 1. In this maze design, the sub-optimal goal is more easily reached. A good shaping-reward function discourages the agent from visiting the sub-optimal goal. As shown in Fig. 1[7] ROSA achieves this by learning to place shaping rewards (dark green) on the path that leads to the optimal goal.

### Subgoal Discovery

We used the Subgoal Maze introduced in (McGovern and Barto 2001) to test if ROSA can discover subgoals. The environment has two rooms separated by a gateway. To solve this, the agent must discover the subgoal (reaching the gateway before it can reach the goal. Rewards are $-0.01$ everywhere except at the goal state where the reward is 1. As shown in Fig. 2, ROSA successfully solves this environment whereas other methods fail. ROSA assigns importance to reaching the gateway, depicted by the heatmap of added shaped rewards.

**Ignoring non-beneficial shaping reward.** Switching control gives ROSA the power to learn when to attend to shaping rewards and when to ignore them. This allows us to learn to ignore "red-herrings", i.e., unexplored parts of the state space where there is no real environment reward, but where surprise or novelty metrics would place high shaping rewards. To verify this claim, we use a modified Maze environment called Red-Herring Maze in which a large part of the state space that has no environment reward, but with the goal and environment reward elsewhere. Ideally, we expect that the reward shaping method can learn to quickly ignore the large part of the state space. Fig. 3 shows ROSA outperforms all other baselines. Moreover, the heatmap shows that while RND is easily dragged to reward exploring novel but non-rewarding states, ROSA learns to ignore them.

### Learning Performance

We compared ROSA with the baselines in four challenging sparse rewards environments: Cartpole, Gravitar, Solaris, and

[6]BiPaRS-IMGL requires a manually crafted shaping-reward (only available in Cartpole).

[7]The sum of curves for each method may be less than 1 if the agent fails to arrive at either goal.

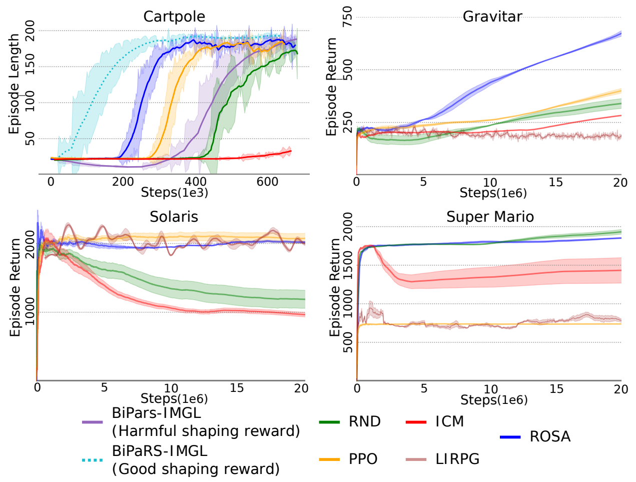

Figure 4: Benchmark performance.

Super Mario. These environments vary in state representation, transition dynamics and reward sparsity. In Cartpole, a penalty of $-1$ is received only when the pole collapses; in Super Mario Brothers the agent can go for 100s of steps without encountering a reward. Fig. 4 shows learning curves. ROSA either markedly outperforms the best-competing baseline (Cartpole and Gravitar) or is on par with them (Solaris and Super Mario) showing that it is robust to the nature of the environment and underlying sparse reward. Moreover, ROSA does not exhibit the failure modes where after good initial performance it deteriorates. E.g., in Solaris both ICM and RND have good initial performance but deteriorate sharply while ROSA's performance remains satisfactory.

## Ablation Studies

To understand how ROSA's performance is affected by components of the algorithm or hyper-parameter settings, we ran a series of ablation experiments. All experiments in this section were run on a simple 25x25 Gridworld in Fig. 5(a).

**ROSA is an effective plug & play framework.** Fig. 5 shows the performance of vanilla PPO and vanilla TRPO versus their ROSA-enhanced counterparts. Particularly notable is ROSA's enhancement to TRPO. Both vanilla TRPO and TRPO+ROSA perform equally well in the early stages of learning, but while vanilla TRPO seems to get stuck with a suboptimal policy, TRPO+ROSA consistently improves performance through learning until reaching convergence.

**ROSA delivers performance boost despite severe impairments to the method.** A core component of ROSA is the exploration bonus term in the Shaper's objective. We ran experiments to check if ROSA can still deliver a performance boost when this important component of the algorithm is either weakened or ablated out entirely. Fig. 6 shows the performance of various versions of ROSA: one with RND providing the exploration bonus, one with a simple count-based measure $L(s) = \frac{1}{\text{Count(s)}+1}$ providing the exploration bonus, and one where the exploration bonus is ablated out

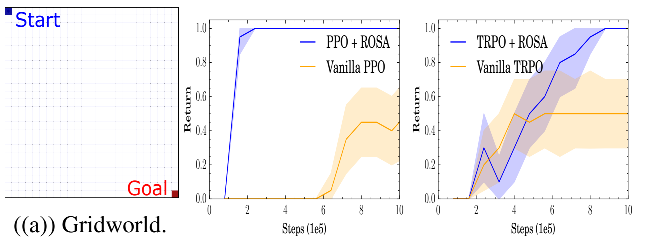

Figure 5: ROSA is an effective plug & play framework. Enhancing both PPO (left) and TRPO (right) with ROSA results in marked performance gains.

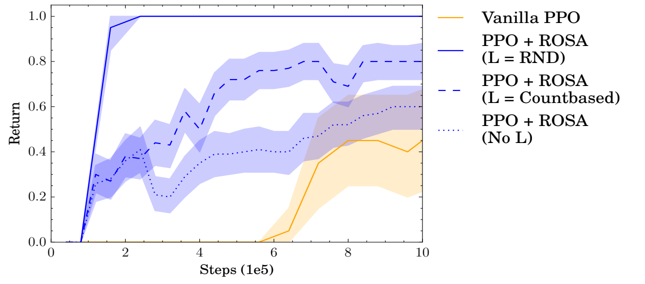

Figure 6: Ablation study on the exploration bonus.

entirely. Stronger exploration bonuses such as RND enable ROSA (PPO+ROSA ($L$=RND)) to provide more effective reward shaping over weaker exploration bonuses (PPO+ROSA ($L$=Count-based)), indicating this is an important aspect. Yet, ROSA can still usefully benefit learners even when the exploration bonus is ablated completely as shown by the fact that (PPO+ROSA (No. $L$)) outperforms Vanilla PPO.

## Conclusion

We presented a novel solution method to solve the problem of reward shaping. Our Markov game framework of a primary Controller and a secondary reward shaping agent is guaranteed to preserve the underlying learning task for the Controller whilst guiding Controller to higher performance policies. Moreover, ROSA is able to decompose complex learning tasks into subgoals and to adaptively guide Controller by selectively choosing the states to add shaping rewards. By presenting a theoretically sound and empirically robust approach to solving the reward shaping problem, ROSA opens up the applicability of RL to a range of real-world control problems. The most significant contribution is a novel approach that marries RL and multi-agent systems which we believe can be adapted to solve other open challenges in RL.

## Acknowledgements

Part of this work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC. Jianhong Wang is sponsored by UKRI Turing AI World-Leading Researcher Fellowship, EP/W002973/1.

## References

Bayraktar, E.; and Egami, M. 2010. On the one-dimensional optimal switching problem. *Mathematics of Operations Research*, 35(1): 140–159.

Behboudian, P.; Satsangi, Y.; Taylor, M. E.; Harutyunyan, A.; and Bowling, M. 2021. Policy invariant explicit shaping: an efficient alternative to reward shaping. *Neural Computing and Applications*, 1–14.

Bertsekas, D. P. 2012. *Approximate dynamic programming*. Athena scientific Belmont.

Blackwell, D.; and Ferguson, T. S. 1968. The big match. *The Annals of Mathematical Statistics*, 39(1): 159–163.

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2018. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Charlesworth, H.; and Montana, G. 2020. PlanGAN: Model-based Planning With Sparse Rewards and Multiple Goals. *arXiv preprint arXiv:2006.00900*.

Devlin, S.; and Kudenko, D. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 225–232. International Foundation for Autonomous Agents and Multiagent Systems.

Devlin, S.; Kudenko, D.; and Grześ, M. 2011. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14(02): 251–278.

Devlin, S. M.; and Kudenko, D. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 433–440. IFAAMAS.

Fudenberg, D.; and Tirole, J. 1991. Tirole: Game Theory. *MIT Press*, 726: 764.

Harutyunyan, A.; Devlin, S.; Vrancx, P.; and Nowé, A. 2015. Expressing arbitrary reward functions as potential-based advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Hosu, I.-A.; and Rebedea, T. 2016. Playing atari games with deep reinforcement learning and human checkpoint replay. *arXiv preprint arXiv:1607.05077*.

Houthooft, R.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; and Abbeel, P. 2016. Vime: Variational information maximizing exploration. *arXiv preprint arXiv:1605.09674*.

Hu, Y.; Wang, W.; Jia, H.; Wang, Y.; Chen, Y.; Hao, J.; Wu, F.; and Fan, C. 2020. Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping. *Advances in Neural Information Processing Systems*, 33.

Igl, M.; Farquhar, G.; Luketina, J.; Boehmer, W.; and Whiteson, S. 2020. The Impact of Non-stationarity on Generalisation in Deep Reinforcement Learning. *arXiv preprint arXiv:2006.05826*.

Mannion, P.; Devlin, S.; Mason, K.; Duggan, J.; and Howley, E. 2017. Policy invariance under reward transformations for multi-objective reinforcement learning. *Neurocomputing*, 263: 60–73.

McGovern, A.; and Barto, A. G. 2001. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 361–368. Morgan Kaufmann.

Mguni, D. 2018. A Viscosity Approach to Stochastic Differential Games of Control and Stopping Involving Impulsive Control. *arXiv preprint arXiv:1803.11432*.

Mguni, D.; Jennings, J.; Macua, S. V.; Sison, E.; Ceppi, S.; and de Cote, E. M. 2019. Coordinating the crowd: Inducing desirable equilibria in non-cooperative systems. *arXiv preprint arXiv:1901.10923*.

Mguni, D.; Sootla, A.; Ziomek, J.; Slumbers, O.; Dai, Z.; Shao, K.; and Wang, J. 2022. Timing is Everything: Learning to Act Selectively with Costly Actions and Budgetary Constraints. *arXiv preprint arXiv:2205.15953*.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Ostrovski, G.; Bellemare, M. G.; Oord, A. v. d.; and Munos, R. 2017. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *International Conference on Machine Learning (ICML)*, 2778–2787.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Shao, K.; Tang, Z.; Zhu, Y.; Li, N.; and Zhao, D. 2019. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*.

Shoham, Y.; and Leyton-Brown, K. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Stadie, B.; Zhang, L.; and Ba, J. 2020. Learning Intrinsic Rewards as a Bi-Level Optimization Problem. In *Conference on Uncertainty in Artificial Intelligence*, 111–120. PMLR.

Stadie, B. C.; Levine, S.; and Abbeel, P. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Strehl, A. L.; and Littman, M. L. 2008. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8): 1309–1331.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Wang, J.; Xu, W.; Gu, Y.; Song, W.; and Green, T. 2021. Multi-Agent Reinforcement Learning for Active Voltage Control on Power Distribution Networks. *Advances in Neural Information Processing Systems*, 34.

Yang, Y.; and Wang, J. 2020. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. *arXiv preprint arXiv:2011.00583*.

Zheng, Z.; Oh, J.; and Singh, S. 2018. On Learning Intrinsic Rewards for Policy Gradient Methods. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zinkevich, M.; Greenwald, A.; and Littman, M. 2006. Cyclic equilibria in Markov games. *Advances in Neural Information Processing Systems*, 18: 1641.

Zou, H.; Ren, T.; Yan, D.; Su, H.; and Zhu, J. 2019. Reward shaping via meta-learning. *arXiv preprint arXiv:1901.09330*.