Universiteit Leiden

Master Computer Science

The XAI-Economist: Explainable Reinforcement Learning in RL-Assisted Policy Generation

Name:	Koen Ponse
Student ID:	s1861581
Date:	12/10/2023
Specialisation:	Artificial Intelligence
1st supervisor:	T.M. Moerland
2nd supervisor:	N. van Stein

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Reinforcement learning (RL) has come a long way in recent years and has recently shown the ability to assist in the creation of governmental policies by being able to both create policies and simulate their potential outcomes. However, if RL-assisted governmental policy design is to succeed, we need scalable RL solutions for complex future environments. Furthermore, we require the wider population to trust the black-box algorithms that will directly affect their lives. Explainability in reinforcement learning, however, is a young field of research with few generally applicable solutions. In this thesis, we recreate a recent application of RL-assisted governmental policy generation in a simpler environment and show how our environment demonstrates the same real-life economic behavior while being a much more scalable solution. We further propose an explainability pipeline for deep RL consisting of novel and existing explainability tools for finding state importance values and feature importance values. We then validate these methods in our new economic environment. While the state importance values may already explain what RL agents find important, we can further process them by reweighting feature importances by the state importance values. We show how this process allows for a clearer and more fair representation of the most important input features of an RL agent.

Contents

1 Introduction **1**

2 Preliminaries **2**
2.1 Definitions . . . 2
2.2 Reinforcement Learning . . . 2
2.3 Foundation . . . 4
2.4 Explainable Deep Reinforcement Learning . . . 5

3 Related work **8**
3.1 AI governmental policy generation . . . 8
3.2 Explainability in Deep Reinforcement Learning . . . 9

4 RL-assisted Economic Policy Generation **11**
4.1 Environment . . . 11
4.2 Methodology . . . 14
4.3 Results . . . 15

5 Explainable Deep Reinforcement Learning **20**
5.1 Explainability in RL-assisted policy generation . . . 20
5.2 State importances in Deep RL . . . 22
5.3 Feature Importances in Deep RL . . . 24
5.4 Results . . . 26

6 Limitations & Discussion **31**

7 Conclusion **33**

1 Introduction

Every day, more and more governmental policies are created that affect the lives of an increasing number of people. It is not surprising then that the task laid out to policymakers is becoming increasingly difficult. As such, historical data and (economic) models are of great importance in deciding on a good policy. However, the accuracy of historical data for predicting future events may be questionable in our ever-evolving world, and it is subject to the Lucas-critique [1]. Furthermore, traditional economic models may not capture the complexity of the world, and we may lack models in new, unexplored situations. Recent advances in AI technology allowed us to create better models, but these still assume the availability of a lot of data.

Reinforcement learning (RL) has come a long way in recent years, even playing a major role in the very well-known large language models [2]. As reinforcement learning may allow us to simulate unseen scenarios with a minimal amount of data, recent work has been published to assist policy-making through the use of RL [3, 4, 5]. These created simulations may not yet be accurate representations of the real world, but they could be the initial steps in assisting policy creation through AI. As the simulations become more sophisticated, these techniques can be used to both create policies and predict their potential outcomes without the need to manually program specific behavior.

In this thesis, we build on the work of the AI economist [4], a two-level multi-agent reinforcement learning algorithm designed to find optimal tax policies. In the AI economist, a small economy is simulated in which RL agents have to maximize their utility, while an RL policy-maker sets tax rates such that equality and productivity are both maximized. This economic framework is then used to generate new taxation rules that are compared with real-world taxes, showing an improvement in equality while productivity stays relatively high.

Although the AI economist is a good first step in AI-assisted policy design, the created environment is computationally demanding, simulates only a small population (4 or 10 agents), and the spatial setting it creates causes unrealistic biased behavior of the agents. As such, we propose a similar simulated economy, but in a simpler 1D environment. We show that our environment is capable of producing the same real-world economic behavior as observed in the original AI economist paper, while agents can learn more than 10 times faster in a much more scalable implementation.

When designing governmental policies, it is key that we use methods that can be trusted by the general population. While current black-box algorithms, such as neural networks, may be required for solving complex tasks, they do not provide the required trust, as they are not transparent by nature. As such, if AI-assisted policy design is to succeed through the use of deep reinforcement learning, we need strong explainability methods for DLR. However, general explainability for black-box algorithms such as deep learning, and especially deep reinforcement learning, is still a young and poorly explored field of research [6, 7, 8].

With this in mind, we propose novel methods for finding state importance values for explainability in reinforcement learning, applicable to any current policy- or value-based deep reinforcement learning method. We experiment with these methods in our new economic simulation. We further use the existing black-box explainers SHAP [9] and LIME [10] for local feature importance, and propose a novel method whereby we reweight the feature importance scores by state importances. This results in a clearer visualization of important features and, perhaps more importantly, provides us with a fairer representation of important input features.

The remainder of this thesis is structured as follows. The next section, Section 2, gives an overview of reinforcement learning in general and about *Foundation*, the framework on which we build our environment. Section 3 discusses earlier work in the field of AI-assisted policy generation and explainability in deep reinforcement learning. Next, Section 4 introduces our environment and methodology and compares our environment with that of the original work [4]. Section 5 discusses our method of explainability by providing insight into the model's state importance and also discusses existing methods for local feature importance explanations. These two methods are combined to provide weighted global feature importance explanations, and the section concludes with experimentation on trained agents of section 4. Finally, Sections 6 and 7 conclude our work with our limitations and pointers for future work.

2 Preliminaries

This section will provide background information on key concepts that are used throughout our work. First, we will define important terminology used throughout the rest of this thesis. Next, we will provide a mathematical and intuitive explanation of reinforcement learning, the backbone of our work. We will further describe *Foundation* and the *Gather-Trade-Build* environment, both designed as part of the AI economist [4]. Finally, this section contains background information and a taxonomy of explainable AI in deep reinforcement learning. This taxonomy is used to group related works and ours.

2.1 Definitions

Throughout this thesis, we refer to *governmental or economic policies*. This is done to avoid confusion between a policy and a *reinforcement learning policy*, which is used in reinforcement learning to describe how an agent decides on its actions. Although we only mention governmental policies, we should note that the general idea of AI-assisted policy design applies to any (to be) enacted policy by governments and businesses.

2.2 Reinforcement Learning

Mathematically, reinforcement learning (RL) [11] environments can be defined as a Markov decision process (MDP). An MDP consists of a 5-tuple (S, A, R, T, γ) where

S is the set of all states (state space), A the set of all possible actions (action space), R is the reward function: $S \times A \times S \mapsto \mathbb{R}$, T is the transition function that maps states and actions into a probability distribution of the next states: $S \times A \times S \mapsto [0, 1]$, and γ is the discount factor, which governs the importance of future states. in an MDP, a state $s \in S$ is a valid Markovian state if the probability of a next state, given an action, is independent of previous states. In reinforcement learning, an RL agent is given $s \in S$ and produces $a \in A$ according to its policy π, where π is trained to maximize the outcome of R, while R and T are unknown. Formally, the policy π gives the probability of action a in state s, given some (trainable) parameters θ:

$$\pi(a|s, \theta) = \mathsf{Pr}(a_t \in A, s_t \in S, \theta_t) \tag{1}$$

The mathematical formulation provides us with a machine learning paradigm, different from (un)supervised learning. In reinforcement learning, we are provided with an environment in which our learner is able to act using a set of pre-defined actions. The actions themselves are not labeled, as might be the case in supervised learning, but rather the actions lead to a new state of the environment along with a scalar reward. The environment state is what the learner observes and uses to decide on its actions, and it is the reward that the learner uses to update its predictions. However, rewards are not directly linked to any action and may even not be obtained after each action. Rather, the learner is tasked with finding specific rewards in a, possibly vast, state-action space, simply by performing actions. Through all this, reinforcement learning promises to be applicable to a broader range of applications without the requirement for human-labeled data, since we can theoretically simulate any environment and only need to define a reward function to optimize for.

Traditionally, *tabular reinforcement learning* focused on environments small enough so that we could fit the entire state action space into memory. Learners were typically assigned the task of updating a $\mathbf{Q}$-table, where each table entry, known as a $\mathbf{Q}$-value, represents the expected future rewards in a state, given an action. A policy could then use these $\mathbf{Q}$-values to select actions. For example, the greedy policy would always select the action in a state with the highest $\mathbf{Q}$-value.

As stated above, tabular reinforcement learning requires relatively small environments. As environments become more complex and contain larger or continuous state/action spaces, tabular reinforcement learning becomes infeasible. To combat this problem, new methods fall into the domain of *deep reinforcement learning*. Deep reinforcement learning methods use function approximation methods (often a neural network) to approximate the state so that not every state-action value needs to be stored in memory. Deep RL methods may follow the same *value-based* approach as traditional tabular RL and use the network to output $\mathbf{Q}$-values for a given state and use a predefined policy π to take actions [12]. However, deep RL methods may also be *policy-based* and output a policy π directly, for example, by providing an action distribution over the available actions. This eliminates the need for a predefined policy π and allows for continuous actions.

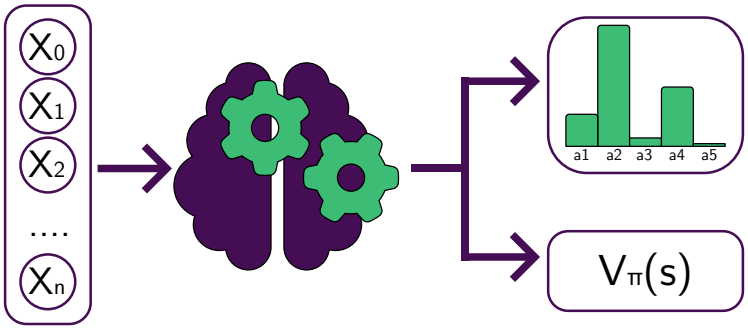

Figure 1: Abstraction of the actor-critic paradigm. A neural network takes in a vector (or matrix) of state features and outputs both a probability distribution over the actions and a scalar value V that is an estimate of the future rewards given the current state. Both the actor and the critic may be the same network with two output heads, be two completely different networks or share parts of the network.

However, pure policy-based methods produce high variance results due to Monte-Carlo estimates for the cumulative rewards. To combat this, some methods adopt the actor-critic paradigm [13] shown in Figure 1. These methods output both a policy and a state value V, where V is the expected future reward from the current state. In this thesis, we use PPO, an actor-critic algorithm explained in more detail in Section 4.2.

2.3 Foundation

Our work builds on *Foundation*, a multi-agent reinforcement learning framework for economic simulations, first introduced in the AI economist [4]. *Foundation* is built to simulate a general population and a government-like policy maker. The worker agents (general population) can take actions in an environment and are all trained to optimize the same, individual, goal. This goal may, for example, be to optimize one's own utility, defined as a function of consumption and labor. Meanwhile, the planner agent is tasked with setting governmental policies that affect the population. The policymaker has its own goal, for example, to optimize the equality and productivity of the population. The Foundation framework can be broken down as follows:

The scenario sets the number and type of agents active in the population, along with their optimization goal.
The world is the 2D grid world in which the population can move and act.
The components define the actions and reactions of the environment that all actors can perform. Creating new components is the primary method of expanding on your environment and is what makes Foundation a friendly method for building environments.

Foundation is built with the Gym-style API [14], providing the familiar `step()` and `reset()` functions, integrating with other libraries that rely on this structure. At each environment step, Foundation steps through each defined component and performs the

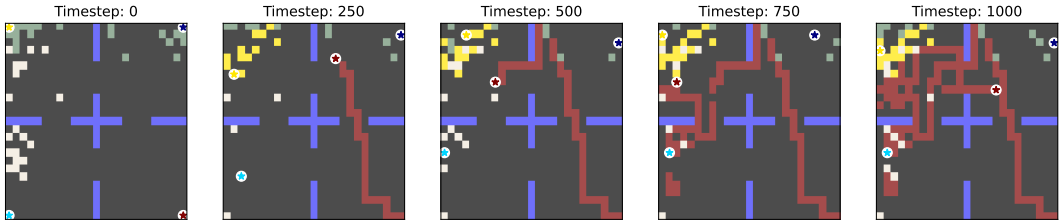

Figure 2: The original Gather-Trade-Build 2D environment [4] at different timesteps of an episode. This version with four agents contains four different quadrants. The colored stars indicate the different agents, with the same-colored pixels indicating houses built by this agent. The blue pixels represent water, which agents cannot move through. White and light green pixels indicate stone and wood respectively. In each episode, agents are spawned in a corner of each quadrant.

agent actions in random order if the agents decide to act in this component. The scenario may also define events that may occur at each step, such as updating the world with newly spawned resources.

With Foundation, the authors of the AI economist [4] have created the Gather-Trade-Build environment, shown in Figure 2. In Gather-Trade-Build, a number of agents are spawned in a grid world with two resources; wood and stone. At each time step, agents can move, build, and trade. All these actions cost a predefined amount of labor. When moving, if an agent lands on a resource, it is picked up and placed in the agent's inventory. Building is the process of combining one of each resource to build a house at the current location. In exchange for building a house, the agent receives coins. The agents are tasked with maximizing their utility, a function of (logarithmically decreasing) coin and labor. In addition, each agent has a different skill level for collecting resources and building houses. These skill levels influence the amount of coins and resources gained when performing the respective action. In Gather-Trade-Build, an additional *planner* agent collects taxes each 100th-time step and sets tax rates for the next 100 time steps. After collecting the taxes, the planner agent redistributes all the collected taxes uniformly among all agents. The planner is tasked with maximizing both the equality between agents and the total productivity. In Section 4 we experiment with the Gather-Trade-Build environment and explain, in more detail, the alterations we have made for our own environment.

2.4 Explainable Deep Reinforcement Learning

To define explainability in our work, we follow the definition of the work of Vourous in which the authors define explainability as *"the ability to provide surface representations of interpretations"* [6]. By this definition, explainable AI is not only the practice of providing good surface representations, but also creating methods that create good interpretations of a model or simply resort to using more interpretable models. However, these improved interpretations should still result in good explanations. A decision tree can be considered

a highly interpretable model, as it provides expressive explanations of its actions through a decision tree chart. However, the explainability of a decision tree may still be low if the tree becomes very large, as we are no longer able to properly provide a good surface representation of this large tree.

In this work, we focus on the *post hoc* methods. Post hoc methods create interpretations of (generally black-box) model's behavior after predictions are made. Although works such as that of Rudin et al. [15, 16] advocate against the use of black-box models in favor of intrinsically interpretable models [17, 18], we cannot ignore the recent successes of deep learning models, particularly when applied to deep reinforcement learning [19, 2, 20, 12]. Yet, it is widely understood that explainability in AI is the key to gaining trust in a system [21, 22, 15], and as such, we should attempt to improve the current post hoc explainability methods.

Generally, we can subdivide post hoc explainability methods in *mimicking* and *distillation* methods [6]. Mimicking methods attempt to mimic a trained black-box model with a more interpretable model, such as a decision tree. Through mimicking, we may gain all the benefits of the more interpretable model, but we should be aware that this method only explains the mimicked model and not the original model. This is generally not an issue if we can use the mimicked model in production (only leveraging the training capabilities of the original black-box model), but this may not always be feasible. Furthermore, as reinforcement learning methods improve and we are able to solve more complex problems through RL, we can expect environments and models to grow in complexity as well. This raises doubts about the scalability of these mimicking methods.

Distillation methods attempt to dissect a model's decision or reasoning via some part of the system, such as the inputs, outputs, or parts thereof. Distillation approaches include methods that attempt to explain an output through importance measurements of the input features. These methods, as well as the discussed mimicking approaches, are largely based on the literature of explainable AI (XAI) of supervised learning. Because reinforcement learning deals with recurrent problems and is more or less modeled after how humans behave in an environment, we can probe our reinforcement learning with different explainability questions. That is, we should be able to probe the model for explanations in a form similar to the way a human would provide explanations. Humans can explain their actions or feelings in various ways, which we list below.

Historical explanations use historical data to provide reasoning on the current action. Someone taking a longer route to work may explain this behavior because in the past it was observed (or heard) that the shortest route is under construction.
Current data (feature importance) explanations simply use data that we can currently observe. These we have described earlier and provide explanations by highlighting important input features.
(Sub-)goals may also be used as an explanation. Going to work can be explained by the objective of buying a house in the future. This kind of explanation attempts to estimate a future state and may be particularly interesting if an agent has multiple objectives, as

it may allow us to visualize what the agent is currently attempting to maximize.
What-if state explanations also provide explanations by estimating a future state. As an explanation for paying taxes, we may provide a state in which we are fined after a tax agency performs an audit. This example of an explanation is in its counterfactual form (*what-if we do not*) and these counterfactual forms are not limited to *what-if* explanations.
Intuitive explanations generally explain an action with a feeling or score about the situation. These explanations may be satisfactory because we can assume that these are a good summary of someone's past experiences. However, when information is highly critical, often when we attempt to visualize one critical action or state, we may want to understand the underlying explanation of this intuition. Yet, because intuitive explanations provide a representation of someone's inner beliefs, they may be highly relevant when attempting to generate trust in the system. In reinforcement learning, intuitive explanations imply the use of a model's output directly to generate explanations. We know that, through training, the outputs are a reasonable summary of the past experiences of the models.

Although the above list is not created as a comprehensive list, it functions as a good, and extendable, taxonomy for explainability methods for reinforcement learning methods. We also note that we can often explain situations or actions with multiple types of explanations. We can, for example, explain the fact that we would like to eat because we have not eaten in a long time (historical data) or because we feel hungry (current data). Both explanations are valid, but different types of explanations may be favorable in certain situations, depending on the target audience.

Local versus global explanations So far, we have discussed the "how" of explainability methods. Important is that we should also categorize "what" is explained. These categorizations are also made in the works of Vourous and Puiutta & Veith [6, 23]. To explain "what" is explained, we have mainly discussed *local explanations*. Local explanations are used to directly explain a specific action. It is important to note that we can also provide *(semi) global explanations* that explain the general behavior of an agent or at least the behavior over a longer period of time. These global explanations can be provided as a summary of local explanations in a single visualization. Furthermore, in addition to explaining actions, we may sometimes want to probe for someone's feelings about a particular situation or state to better understand its behavior. For example, we expect self-driving cars to find states with crossroads to be very important. We also expect models to be surprised when we know we have placed them in a highly nontypical state. If these *feelings* are not what we anticipate them to be, we are less likely to trust the system.

3 Related work

This section outlines earlier work on both subjects of this thesis. First, we discuss related work in AI-assisted governmental policy generation, both in the domain of reinforcement learning and non-RL works. Next, we describe previous work in the domain of explainability in deep reinforcement learning, not particularly applied to AI-assisted governmental policy generation.

3.1 AI governmental policy generation

Since modeling real-world behavior is a complex problem that is likely to require large amounts of computational power, designing and modeling real-world policies through artificial intelligence is a relatively new field. Previous work has been conducted in the space of governmental policy validation. Garrido and Mittone have trained agents using real-world data and use these agents to validate various tax audit strategies to optimize against tax evasion [24]. A similar paper by Bloomquist studied optimal strategies to combat tax evasion by small businesses [25]. Both studies show that it is possible to optimize tax audit strategies in their simulated world, highlighting the capabilities of AI-assisted governmental policy generation. However, in these works, the validated policies are still human-crafted, and the training of the agents required real-world data.

With recent advances in reinforcement learning, the AI economist [26, 27, 4] has attempted to make progress in the field by eliminating the need for real-world data and allowing AI-created policies. They introduce a simulated economy in a 2D world in which multiple RL agents with different skill levels can move, gather resources, build houses for income, and trade with each other. Simultaneously, an RL planner agent sets new tax rates every 100 time steps with the goal of maximizing equality times productivity. They compare tax policies created by their planner with existing tax policies and a free market setting. They find that, in their simulated environment, the RL-generated tax policies outperform existing policies when optimizing for productivity and equality.

Trott et al. [5], build on the same framework introduced by the AI economist, and experiment with COVID-19 policies in the USA. Here, the agents are the states that may enact more strict policies, possibly reducing COVID-19 deaths but increasing unemployment. The government (planner) is responsible for providing subsidies for policies it thinks need to be enacted to uphold social health and productivity.

Koster et al. [28] develop *Democratic AI*, in which reinforcement learning is used to design a policy that a majority of real humans prefer. This is validated in an online game where humans have to decide whether to keep or share a sum of money. An RL agent and humans both designed a mechanism for redistributing the money persuading humans to share. The authors found that the RL agent successfully outperformed the human mechanism by often winning the majority vote.

Recently, a policy design competition was held [3], in which competitors had to design AI

agents that negotiate with each other in a simulated world. Measured is then how these negotiated deals affect the global economy and the global climate crisis. The results have not yet been made public, but this competition highlights an increase in the popularity of AI-assisted policy design.

3.2 Explainability in Deep Reinforcement Learning

As discussed in Section 2.4, this work focuses on post hoc distillation methods for black-box reinforcement learning methods. In this area, various works exist to provide local explanations for a model's action through feature importance explanations. Obtaining approximated global feature importances in these methods is often trivially done by taking an average over a range of local explanations. Here, various methods focus on visual-based agents, often highlighting the visual element of a state to which attention is paid [29, 30, 31, 32, 33]. Other methods in this category include SHAP and LIME [9, 10]. Although not developed for reinforcement learning in particular, both methods are model-agnostic tools for finding feature importances given a single input (or observation). SHAP does this by removing features from the input one at a time and observing the effect on the output. SHAP has been applied in various reinforcement learning works to explain the outcomes of a model [34, 35, 36, 37]. LIME is similar to SHAP, but instead of removing features, it slightly shifts features around its mean. In Section 5, we will provide more detailed explanations of both SHAP and LIME and will use both methods in our pipeline to explain RL agent behavior.

As we have previously discussed in Section 2.4, reinforcement learning provides opportunities for various forms of explanations. Most notably, we are able to use the extra time dimension for explanations. Reinforcement learning explainability methods can, and likely should, take advantage of this. Amir and Amir were among the first to have done so by introducing the HIGHLIGHTS algorithm [38]. HIGHLIGHTS aims to provide the end-user with a summary of important states so that the user can make an educated assessment of the model's performance. To estimate importance, as measured by the model, HIGHLIGHTS assumes access to the model's $\mathbf{Q}$-values and calculates state importance by comparing the best action in a state with the worst action:

$$I(s) = \max_{a \in A} \mathbf{Q}^{\pi}_{(s,a)} - \min_{a \in A} \mathbf{Q}^{\pi}_{(s,a)} \tag{2}$$

HIGHLIGHTS is validated by showing summaries to humans of three agents with different skill levels. The humans are then tasked with selecting the best-performing agent. Results are compared against random summaries, and summaries only showing the first few states. The results show that the HIGHLIGHT-generated summaries outperform both baselines when it comes to correctness and confidence. HIGHLIGHTS attempts to explain a model by means of semi-global historical explanations, which are built by summarizing local intuitive explanations. HIGHLIGHTS success over the baselines shows that explanations built through a model's $\mathbf{Q}$-value estimate can be a valid explanation method, demonstrating the strength of intuitive explanations. However, a weakness of

HIGHLIGHTS comes from its evaluation. The agents compared are trained for various lengths. This means that the weaker agents may not yet have had the time to generate appropriate $\mathbf{Q}$-values and, in turn, they will not be able to demonstrate their true behavior well.

Probably developed simultaneously, Huang et al. [39] publish a work similar to HIGHLIGHTS in which they seek to automatically generate a set of *critical* states, as determined by the model through intuitive explanations. Huang et al. describe two methods of identifying critical states. The first method is value-based and is similar to the HIGHLIGHTS method. We again assume knowledge of a model's $\mathbf{Q}$-estimate and then calculate the state importance via the following formula:

$$I(s) = \max_{a \in A} \mathbf{Q}^{\pi}_{(s,a)} - \frac{1}{|A|} \sum_{a \in A} \mathbf{Q}^{\pi}_{(s,a)} \tag{3}$$

Huang et al. then determine whether a state is critical by comparing $I(s)$ to a user-defined threshold t:

$$\mathcal{C}(s) = I(s) > t \tag{4}$$

Compared to Equation 2, Huang et al. assign an importance score based on how much better the best-performing action is compared to all other actions, rather than the worst action. In reinforcement learning, this value is often referred to as the advantage ($\mathcal{A}$) of a state-action pair[1]. Equation 3 is likely to give a better estimate of the importance of a state, due to a lower variance.

The second method of Huang et al. of measuring state importance is policy-based. If the model outputs a policy directly in the form of a distribution over the actions, Huang et al. measure state importance by calculating the entropy $\mathcal{H}$ of the action distribution:

$$I(s) = \mathcal{H}(\pi(s|\cdot)) \tag{5}$$

The idea here is that in critical states, the necessary action would be very obvious and highly probable, resulting in a low entropy. As such, Huang et al. marks a state as critical if its entropy is below threshold t:

$$\mathcal{C}(s) = I(s) < t \tag{6}$$

Huang et al. found that computing importance scores through the value-based approach is more reliable, however, the policy-based approach is directly applicable to continuous action spaces without the need for discretization. In Section 5.2, we build on both approaches of Huang et al. to estimate state importances in multi-agent environments where agents have access only to a state's value (V).

[1] Here, the advantage $\mathcal{A}$ is calculated over a random policy. However, more generally in RL, the advantage is weighted by a (trainable) policy.

Sequira et al. further produced multiple works that extend the HIGHLIGHTS algorithm with additional interestingness factors beyond state importance [40, 41, 42]. Their most recent work [42] proposes an extensive toolkit for the explainability of reinforcement learning. In this paper, the authors go beyond providing intuitive explanations for the various interestingness factors and also provide SHAP [9] explanations to explain the intuitive explanations by means of feature importances. The proposed work is a first step in solving the lack of an easy-to-use and generally applicable explainability method for reinforcement learning. However, the toolkit has as of yet not been released and is not designed for multi-agent environments.

The explainability method of Bogges et al. [43] is two-fold. First, they propose to provide a user with a summary of an agent's policy, which is provided in the form of historical data. This summary is based on states that satisfy the inclusion of one or more predefined predicate features. This set is further refined by including only likely states. To provide a more local explanation of the agents, Bogges et al. also propose to provide a query-based explanation in the form of NLP answers to the following questions: *"when do agents do actions"*, *"why don't agents do actions in states"* and *"what do agents do in predicates"*. The first two questions provide feature importance explanations, with the latter doing so in counterfactual form. The final question uses intuitive explanations by providing action distributions over states that contain predicate features. The strength of Bogges et al. is that through the predicate set, the pipeline provides NLP explanations that can immediately be understood by a human. However, this implies that an expert has to create predicate sets beforehand, and larger predicate sets may adversely affect the explainability. Furthermore, the method is not tested on problems with large state-action spaces and, as such, may not be applicable to real-world simulations.

4 RL-assisted Economic Policy Generation

This section will first describe our environment, built as an adaptation of *Foundation* and the *Gather-Trade-Build* environment (Section 2.3). Our experimental setup is largely based on that of the AI economist [4], but adapted to our environment and lacking comparison with the existing US and Saez tax system [44]. We highlight the changes between our work and the AI economist, reproduce their work, and compare it with our environment. We show that our simplified and more scalable setup demonstrates the same real-life economic behavior in agents and, as such, is an improved tool for AI-assisted governmental policy design.

4.1 Environment

The base version of Foundation and the Gather-Trade-Build scenario of the AI economist uses a 2D map in which agents move around, gather resources, and build buildings to generate income. This spatial setting, with partial observability, creates the need for a complex network that is created in the form of a recurrent network with convolutional

layers to process the spatial map input. This complexity slows down training and requires extensive tuning. In turn, users are less able to prototype different scenario settings and/or scale to more complex scenarios or larger populations. Furthermore, as shown in the AI economist, a spatial setting may cause unrealistic behavior, such as agents being blocked by buildings and resources.

To solve the described problems, we propose a new environment, built with an adaptation of the *Foundation* framework that allows 1D vector observations. Similarly to the *Gather-Build-Trade* scenario used by the AI economist, our scenario allows agents to mine for resources, convert these resources into coins, and trade with other agents.

Mining allows an agent to choose one of the resources in the scenario and gain one or more of these resources from the available world resources in exchange for performed labor. The amount of resources mined by an agent is determined by the following formula:

$$M_r = min(W_r, max(1, \lfloor S_m + \rho \rfloor)) \tag{7}$$

where W_r is the amount of currently available resources in the world (infinity in our experiments), S_m is the skill of an agent and ρ is a random number between 0 and 1. Agent skills are randomly assigned to agents according to a Gaussian distribution with μ 1 and σ 0.16[2], similar to the distribution of human IQ levels [45].

Building allows agents to convert a number of each resource into coins. Coins directly influence the reward signal. In contrast to the AI economist, agents are not required to stand on an empty plot of land and can perform this action at any time they have the required resources in their inventory. When building, an agent will lose the required resources and will gain a number of coins multiplied by their building skill (S_b). Similarly to the mining skill, the building skills are randomly distributed according to a Gaussian distribution with μ 1 and σ 0.16[1].

Trading allows agents to place a bid or sell order for one of the resources, exchanging a single resource for coin. Orders are active for a predetermined number of time steps, and any agent can place a maximum number of orders simultaneously. After each component step (after placing all orders), whenever a *bid price* is higher than or equal to an *ask price* of the same resource, a match is found and the resources and coins are exchanged between the two agents. When a match can be made between multiple existing orders, priority is given to the lowest ask and highest bid. Ties are first resolved by the lifetime of the order and then at random. When a match is made between two orders with different prices, the most recent order (the order that triggered the match) sets the price. This is in contrast to the original AI economist implementation, where the oldest order sets the price. We found that the original caused agents to bid (ask) at high (low) prices so that their order would be evaluated earlier without punishment.

[2]in order to amplify differences in skill levels in small populations, we raise each final skill level to the power of 8

At each time step, agents observe their inventory and labor, the available resources in the world, the state of the market, and the current tax brackets if applicable. A comprehensive image of the observation space is shown in Figure 3. Agents can then perform a single action out of the described components (mine, build, trade). A reward is given each time step as the delta in an agent's isoelastic utility minus their obtained labor:

$$U_{a,t} = \frac{c_{a,t}^{1-\eta} - 1}{1-\eta} - l_{a,t} \tag{8}$$

$$R_{a,t} = U_{a,t} - U_{a,t-1} \tag{9}$$

Where $c_{a,t}$ and $l_{a,t}$ are an agent's number of coin and labor respectively at time step t. η is a constant ≥ 0. Higher η values will create a more concave function, where marginal utility will decrease faster with each additional coin earned.

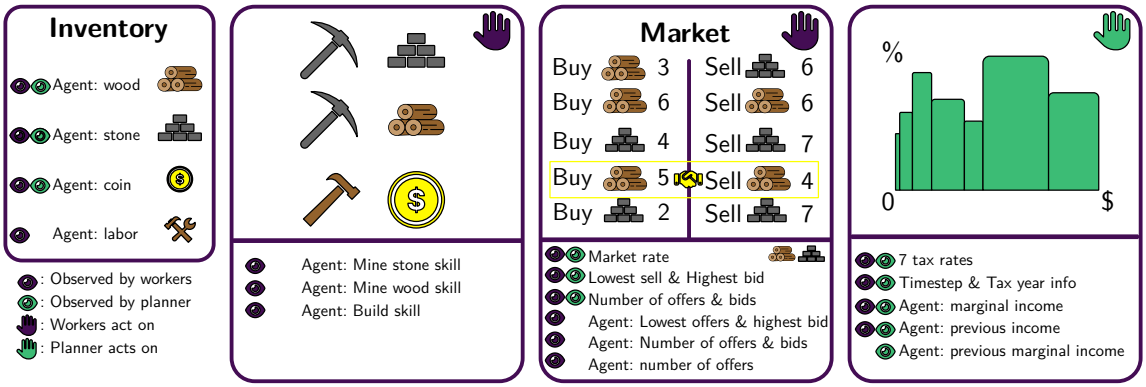

Figure 3: An overview of the components of our 1D vector environment, modeled after the 2D Gather-Trade-Build environment [4]. Worker agents keep an inventory and can mine for stone and wood, convert these resources into coins by building, and place orders on the market. The planner agent sets tax rates for the next 100 time steps. After each 100 time steps, taxes are collected according to these tax rates, and the collected coin is uniformly redistributed. An overview of the observation space is given through the eye icons. Whenever an observation property is prefixed with *'Agent:'*, this property is related to each individual worker agent and is not observable to other worker agents. For instance, *Agent: stone* is the amount of stone kept by a single agent, which is not observable by other agents. In contrast, *Market rate* refers to a global market rate, whereby all agents observe the same value. Whenever the planner observes an *Agent* observation, the planner does so for each agent individually. Market observations are separate for each of the resources (wood and stone).

The final component in our environment is only acted upon by the planner agent. As is the case in the AI economist, the planner agent sets tax rates in predefined brackets every 100 time steps. Before setting new tax rates, the planner will also collect coins from all agents according to the previously in-place tax brackets and will redistribute the collected coin uniformly across all agents. To set its taxes, the planner observes the

current and past incomes of all agents, as well as their inventory. Some information, such as an agent's labor, remains hidden. Again, a comprehensive overview of the observation space can be found in Figure 3. The planner is rewarded for increasing equality and productivity. Productivity is measured in the total coin among all agents and equality is measured as the Gini index [46]:

$$R_{p,t} = (E_t \cdot (\sum_{a=0}^{n} c_{a,t})) - (E_{t-1} \cdot (\sum_{a=0}^{n} c_{a,t-1})) \tag{10}$$

$$E_t = \text{Gini}_t = \frac{\sum_{a_i}^{n} \sum_{a_j}^{n} |c_{a_i} - c_{a_j}|}{2n^2 \bar{c}} \tag{11}$$

Where n is the total number of agents in the environment and $\bar{c}$ is the average amount of coin across all agents.

4.2 Methodology

We trained all agents using RLlib [47], allowing us to generate environment traces in parallel in 30 environments with different random seeds. Both the planner and the agents use a similar model of 2 fully connected hidden layers of 128 nodes with all worker agents sharing the same weights. Furthermore, all agents are updated using the RLlib implementation of proximal policy gradients (PPO) [48]. We have used the clipped PPO version, calculating the loss without a KL-penalty via the following formula:

$$L_{PPO} = \text{min}(\Phi_\theta(a|s) \cdot \mathcal{A}, \text{clip}(1-\epsilon, 1+\epsilon, \Phi_\theta(a|s)) \cdot \mathcal{A}) \tag{12}$$

$$\Phi_\theta(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} = \exp(\log \pi_\theta(a|s) - \pi_{\theta_k}(a|s)) \tag{13}$$

$\Phi_\theta(a|s)$, is the ratio between the probability of action a in state s in the old policy π and the new policy π_k. Due to the clipping in Equation 12, this ratio cannot stray further than the clipping parameter ϵ. The right-hand side of Equation 13 is generally how computing this ratio is implemented. Notably in Equation 12, is the Advantage $\mathcal{A}$, which we have discussed prior in Section 3.2. The advantage here is calculated by means of the critic model, part of PPO. The critic model is a regression model, updated with a mean squared error loss function. Lastly, to promote exploration, the complete loss also includes the entropy of the currently produced action distribution:

$$L_{\mathcal{H}} = \omega \cdot \mathcal{H}_\pi(s) \tag{14}$$

$$L_\pi = L_{PPO} - L_{\mathcal{H}} \tag{15}$$

The entropy coefficient ω scales the importance of the entropy in the loss. High ω values will promote action distributions with high entropy, rewarding more random action distributions and in turn increasing exploration. It is then common practice to anneal ω during

training, promoting exploration during the initial phases, and gradually decreasing this as the model learns how to properly act in the environment. This entropy regularization, together with stochastically sampling from the produced action distribution, is our main method of exploration in all our agents.

All hyperparameters, for both agents and planners, together with those of the environment, are listed in Table 1. Most notably in the environment parameters, the available resources is set to infinity. Although this eliminates scarcity, it removes an extra hyperparameter that is to be varied with the amount of agents. Additionally, time is still scarce, so agents still have to carefully manage their actions to maximize their reward.

Similarly to the curriculum learning approach used in the AI economist, we first train worker agents without any taxes in a *free market* setting, to ensure that agents start working. After training for 5 million time steps, we start training workers and planner simultaneously with taxes enabled. Agents are then tasked with maximizing their, now post-tax, utility. Lastly, in order to speed up training and prohibit agents from taking actions that are not possible, we apply action masking whenever an action cannot be performed. For example, a sell order cannot be placed on the market whenever an agent has no resources in their inventory to sell. Action masking is performed by setting the action probability of the masked actions to 0 after predictions are made.

4.3 Results

To reproduce the original AI economist results, we have used the code provided in the AI economist GitHub[3] and used the same two-phase training procedure to train the agents. First, agents were trained for 25 million steps in a no-tax environment. Afterwards, we enabled the planner and trained both agents and planner for an additional 150 million steps in both the environment with taxes enabled, and taxes disabled. The original experiments were conducted over 1 billion steps, far exceeding our 150 million steps. However, our results showed that rewards have (almost) converged at 150 million steps, whereas the original results showed training was still ongoing at over 800 million steps. This difference may be explained by the original graphs presenting results in their 10-agent environment, where we have used their 4-agent setup for our reproduction. Nevertheless, our reproduction results, shown in Figure 4, verify the original results and show how the AI economist is a capable method of finding new tax schedules that manage to increase equality while minimizing reduction in production.

However, our reproduction results highlight a key issue with the original AI economist setup, namely its computational requirements. Each of our runs was completed in approximately 8 to 9 days[4]. After the 150 million steps, most of our available RAM (1.5TB) was used up, limiting us from conducting longer experiments. As such, we conclude that the original AI economist setup has extreme computational demands for even a simple

[3] https://github.com/salesforce/ai-economist/tree/a84d5f3fdcabb207d9fde7754d34906903b3e184

[4] We used 16 Intel Xeon E5-2630v3 cores@ 2.40GHz (32 threads)

Worker agents	
Number of hidden layers	2
Number of hidden nodes	128
Discount factor (γ)	0.998
learning rate (α) start, end, steps	0.5, 0.025, 1000000
Entropy coefficient (ω) start, end, steps	0.5, 0.025, 1000000
Planner agent	
Number of hidden layers	2
Number of hidden nodes	128
Discount factor (γ)	0.998
learning rate (α) start, end, steps	0.003, 0.0001, 20000000
Entropy coefficient (ω) start, end, steps	2.0, 0.125, 20000000
Training paramaters (both planner and workers)	
PPO clip parameters	0.3
minibatch size	2000
num sgd iterations per update	5
train batch size	10000
Environment parameters	
Episode length	1000
Tax period length	100
Resources	2 (Wood, Stone)
Number of resources to build	3 per resource
Base build payment	20
Max multiplier build payment (B_s)	5
Build labor cost	2
Mine labor cost	2
Trade labor cost	0,02
Max bonus mining skill (B_m)	5
World resource (W_r) regeneration probability	∞ (no scarcity)
Agent starting coin	0
isoelastic eta (η)	0.27

Table 1: Model parameters for both our planner and worker agents and the environment parameters. All worker agents in the environments share a single network. Both the learning rate and the entropy coefficient are linearly decayed, from *start* to *end* over *steps*.

economic simulation with a population of 4. As real-world use cases will most likely require a more realistic environment, computational requirements will only increase. Creating a more realistic environment will make this problem worse, prohibiting any real use cases. We argue that AI-assisted policy generation should be more scalable and allow for relatively quick training, such that different scenarios can quickly be prototyped. For

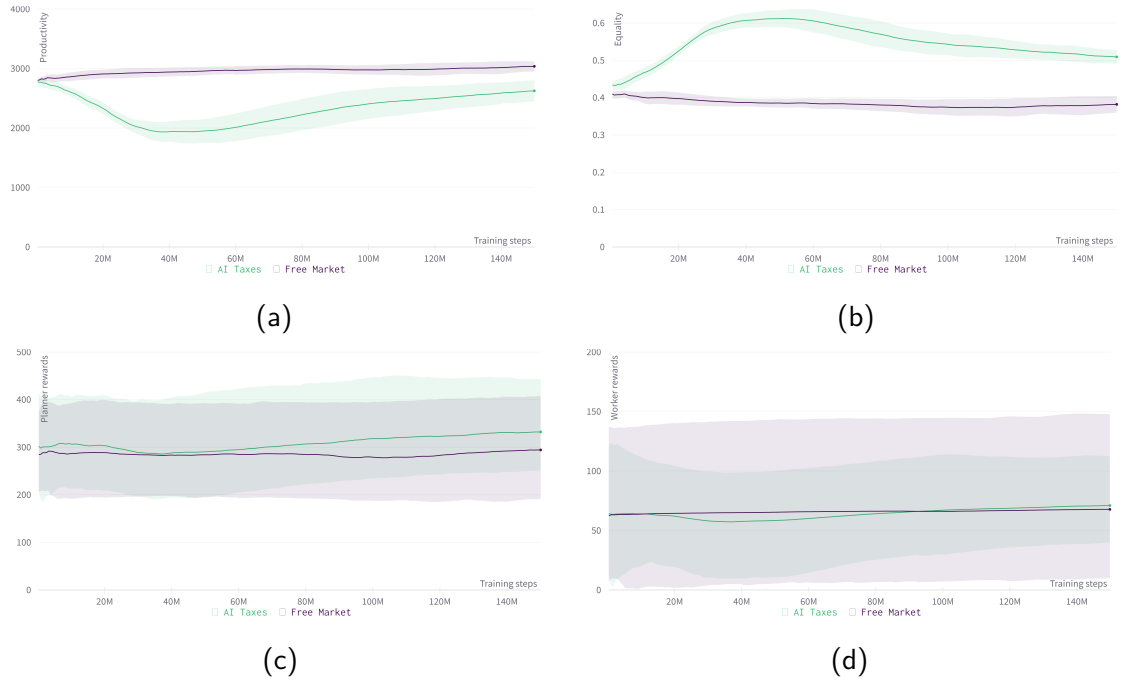

Figure 4: Original AI economist reproduction results. Results are averaged over 4 repetitions. Absolute numbers differ from the originally presented results [4] due to possible changes in the environment settings. Yet, we observe the expected behavior of the AI economist (green line) managing to raise equality (Figure 4b) through its taxes over the free market setting (purple line). Due to these taxes, we expect productivity to drop compared to the free market setting, which we observe in Figure 4a. The final planner rewards in Figure 4c show that the combined task of optimizing both equality and productivity is improved upon by the AI tax planner. Figure 4d shows us that the worker agents achieve, on average, marginally higher utility. However, the maximum and minimum values (indicated by the shaded area) are brought substantially closer by the taxes.

this reason, we recreated the original environment as outlined in Section 4.1.

The results of our vector environment are shown in Figure 5. We observe similar results in our simplified environment but converge after roughly just 40 million training steps, or 20 hours, in an environment of 15 agents (in contrast to 4 in the reproduction).

The AI economist's [4] Gather-Trade-Build environment demonstrated real-world economic behavior without preprogramming this behavior. This is noted as the key advantage of RL-assisted governmental policy generation, as environments can be simulated without the need to manually implement all behaviors (which may be incorrectly implemented). In order for our simplified environment to be an improvement over the original work, we should be able to observe similar behavior emerge. We demonstrate that our simplified environment recreates the same three economic phenomena.

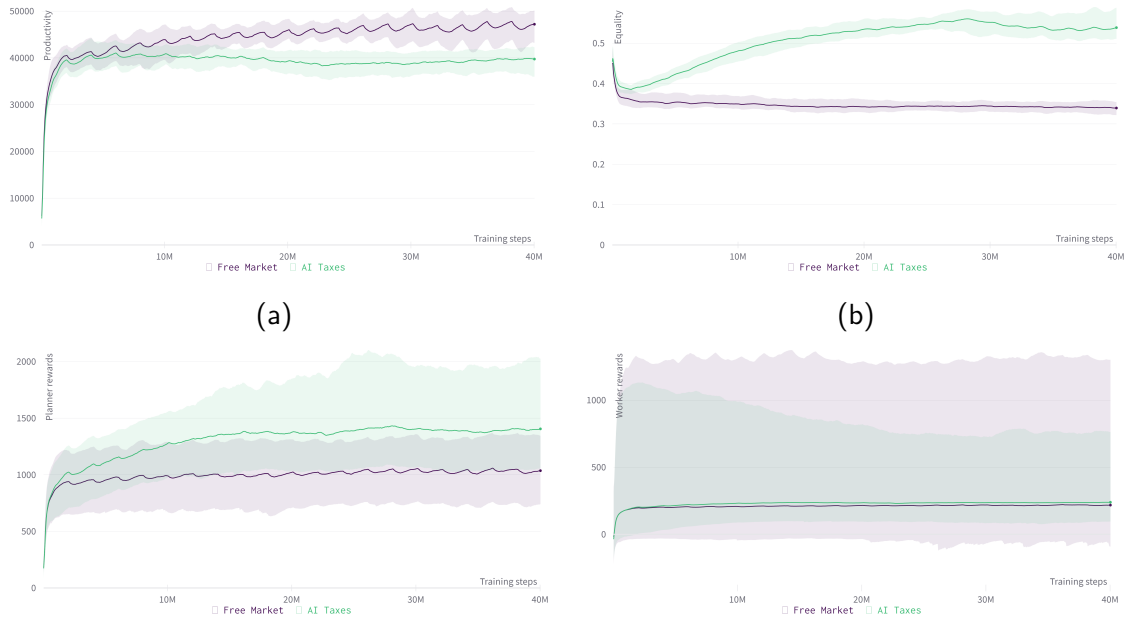

Figure 5: Results of our vector environment during 40 million training steps. The top two images show productivity and equality averages over the 30 simulated environments. The two bottom images represent the average rewards of the planner (5c) and workers (5d). Shaded areas indicate the minimum and maximum observed values across agents in the 30 environments. Figure 5d shows that tax policies slightly increase the average utility of all agents.

The equality-productivity tradeoff is the phenomenon observed in the real world, whereby people are not motivated to perform difficult labor if they are not adequately rewarded. Generally, a government has a great incentive to increase equality among its population, which may be achieved by raising taxes and redistributing wealth. However, high taxes can disincentivize working, as marginal incomes become smaller and work becomes less rewarding. This increase in equality may then have an adverse effect on total productivity, and this effect should be carefully managed with any tax policy. This effect is demonstrated in our environment in Figures 5a and 5b. The free market setting achieves higher overall productivity compared to the environment with taxes enabled, but this causes lower overall equality.

Specialization in our environment is demonstrated in Figure 6. We visualize the distributions of each agent and clearly show that each agent has a different approach to maximizing its post-tax utility. This behavior is not manually programmed, but rather just a result of their randomly distributed skill level for mining resources and building. We can see some agents opting to mine and sell, some agents buy their resources to build, while other agents may choose to generate income simply by buying low and selling high.

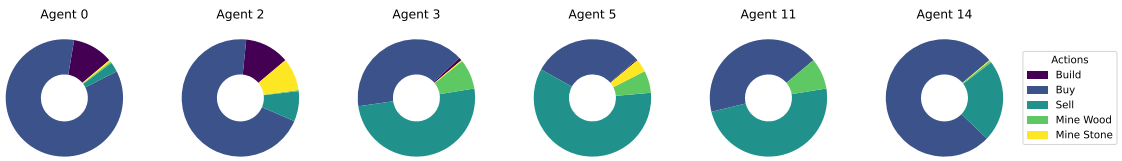

Figure 6: Action distribution over an entire episode in our environment for a subset of agents. Agents are not manually programmed to pursue certain actions but still specialize in a different strategy to optimize their post-tax utility. This emergent specialization is only due to the different randomly assigned skill levels for mining and building. While this data comes from a single episode, similar distributions are found in other episodes and training runs.

Finally, the authors of AI economists show that their agents also demonstrate *tax gaming* strategies. Agents move labor to low tax years in order to maximize their total post-tax income. We can demonstrate this very same behavior in our environment in a similar fashion by calculating the paid tax under the assumption that agents earned a fixed income each year (the average yearly income). Figure 7 shows this effect by plotting the actual taxes paid against the taxes to be paid under the described assumption. This effect may also be explained by the productivity-equality trade-off, as it shows that agents work more in low-tax years. Nonetheless, all described behaviors demonstrate real-world economic effects that emerge simply by training to optimize a real-life goal rather than manually modeling this behavior. This emphasizes the advantages of reinforcement learning for simulating behavior in economic simulations.

We finish this section with a discussion about the produced economic policies. We have plotted some of the RL-created tax schedules in Figure 8. We can observe that the created tax schedules are very unconventional. That is, whereas real-life tax schedules usually follow a progressive system, the produced tax schedules do not adhere to this. Naturally, real-world use cases should most likely constrain the policymaker in some way such that the produced policies would align better with the preferences of the population. Likely, the policymaker should also be penalized for producing erratic tax schedules that are very different from the previously in-place tax schedule. It is worth noting that not limiting the policymaker in such ways could be a major advantage of using RL to create governmental policies, as it could lead to the development of novel concepts. An illustration of this can perhaps already be seen in Figure 8, where taxes are typically higher for lower incomes than for middle-income brackets (and then taxes increase again for the highest incomes). This could be interpreted as the planner guiding agents into the middle-income brackets, encouraging them to work but then punishing them for becoming too wealthy. Highly rewarding agents for doing an *average* amount of work.

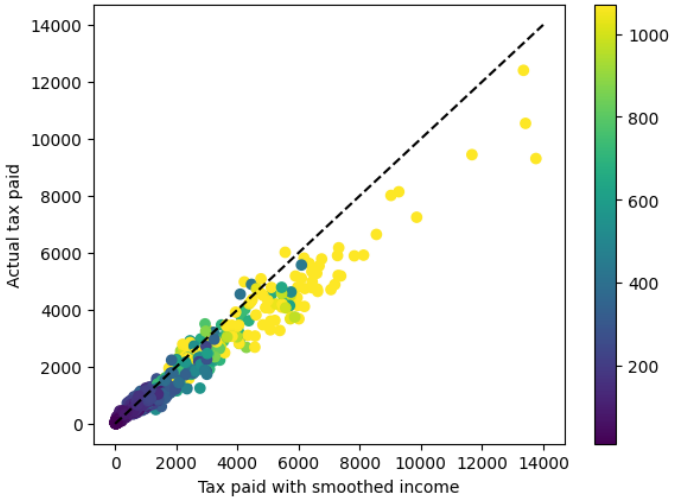

Figure 7: Tax gaming strategies visualized for all 15 agents with data from 100 different episodes. The colors indicate the average income earned by each agent. The actual paid tax is compared with the tax agents would pay if they consistently performed an equal amount of labor (earned 1/10th of the average total income) in each tax year. Agents often move labor into low tax years and are in turn able to decrease their overall taxes.

5 Explainable Deep Reinforcement Learning

This section delves into our method of explainability in multi-agent deep reinforcement learning and will validate our method on the models created in Section 4. Our pipeline consists of three methods for discovering state importance, inspired by previous work focusing on small tabular problems [38, 39, 40] and more recent work that includes the focus on deep RL methods [41, 42]. The proposed methods make use of the current typical deep RL model outputs (the value head and the policy head) and, as such, should be applicable to most, if not all, current deep RL algorithms.

5.1 Explainability in RL-assisted policy generation

A policymaker considering reinforcement learning to aid in its policy creation process has some important explainability issues that need to be addressed when it comes to the produced policies. First, the policymaker must be certain that the policies produced are not biased or discriminatory. Another issue is that a policymaker needs to understand the considerations that an RL planner has made. Finally, the policymaker must be able to verify that the simulated environment is indeed simulating real life. These final two issues are broadly defined, and a policymaker may have many questions to satisfy them. We explain them more thoroughly below.

The **discriminatory policy issue** is a prevalent issue in any AI system that has to make decisions about humans. In typical AI systems, the training data may contain inherent

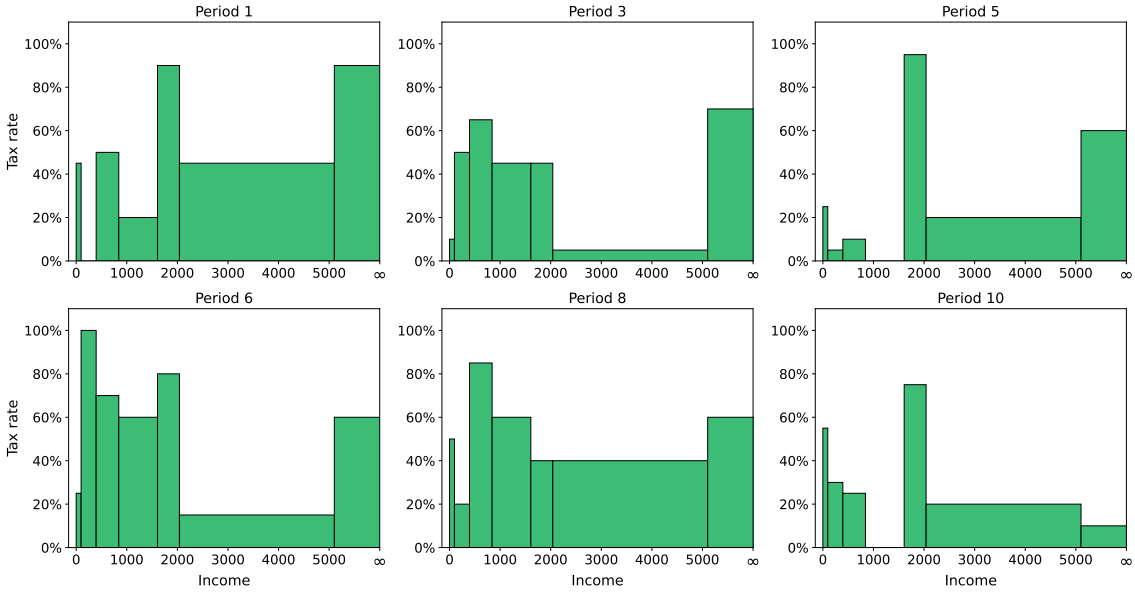

Figure 8: Different tax schedules created by the RL planner, after training, in a single episode. Similar, unconventional and erratic tax schedules are found in different episodes. The cutoffs of the different brackets are fixed, the planner is only able to adjust the tax rates within these fixed cutoffs with intervals of 5%. Real-world applications would likely require constraining the planner, but these unconventional outcomes may provide insights not thought of before. Interestingly, we often observe that taxes are higher in low-income brackets than middle-income brackets (after which taxes rise again). This could be seen as the planner guiding the worker agents into middle-income brackets, motivating them to do an average amount of work but punishing them for becoming too wealthy.

biases, and the model may be taught to replicate these biases. Because RL works without such training data, this problem is partly alleviated. However, RL systems may still learn unwanted biases that need to be made visual. We may address this problem by removing certain input features, yet we still run the risk that other input features inadvertently permit the same behavior [49]. Further still, removing all features that may lead to (negative) biased or discriminatory behavior may not always be feasible, especially in a policy creation task. For example, if we design traffic policies, we may want to include data on percentages of people with visual impairment in a region to create a good policy. However, we must then ensure that the reinforcement learning system is not using this data negatively. Because we cannot effectively eliminate all input features that lead to biases or discrimination, we have to inspect feature importances. In the next two sections, we go over our pipeline for feature importance in reinforcement learning, which involves utilizing the previously discussed SHAP and LIME methods [9, 10]. In this thesis, we apply this pipeline in the domain of RL governmental policy creation, but note that the proposed work should be applicable to other domains in model-free deep reinforcement learning.

The **issue of understanding** the produced policy is more domain-specific and likely requires a solution that is less generally applicable. Likely, a policymaker would benefit from a flow chart that indicates which actions are taken based on the composition of population characteristics. To further create trust in the system, a policymaker would also likely need to probe the system with various *what-if* questions. This work does not aim to solve these explainability questions. However, part of our pipeline, explained in the next section, does highlight important time steps and, as such, does give some insight into what the RL policymaker thinks about the state of the world. If this aligns well with the policymakers' thoughts, this will, we hope, increase their trust in the system.

Finally, the **environment validation issue** would generally require comparisons with real-world data. While this is not necessarily a traditional XAI solution, policymakers would likely only trust a simulation that demonstrates real-life behavior. For example, we may simulate existing tax rate schedules and observe whether the simulated equality and productivity match that of the real world. Even if we meet this requirement, we must still verify if the agents behave like real people to create new policies, since the agents are placed in a new unseen environment that we cannot validate with actual data. This work attempts to partly validate human-like behavior by investigating whether the agents pay attention to features that humans would and whether the agents find similar time steps as important.

5.2 State importances in Deep RL

In this section, we describe our methods for estimating state importance (as measured by the model). Visualizing the importance of states has proven to be an effective method for increasing trust in a system and quickly demonstrating a model's capabilities [38, 39]. Furthermore, state importance may partly aid us in solving the *environment validation issue*, by allowing us to compare a model's state importance values with those of a human. If these values align, this may increase trust in the simulation. Perhaps most importantly, state importance values allow us to improve global feature importance measurements in reinforcement learning systems by weighing local feature calculations by their importance. We argue that this improves measurements because important features in important states are more important than important features in unimportant states.

Similarly to the work of Huang et al. [39], we measure state importance through policy-based and value-based methods. Allowing compatibility with each of the types of outputs as shown in Figure 1. We have altered their methods and reformulated them so that they are applicable to multi-agent environments. Together, our two methods are applicable to value-based [12], policy-based [50], and actor-critic-based models [13]. Accommodating for most, if not all, current model-free deep RL models.

Value-based State Importance If the used RL model predicts state values V (the model's estimate of future rewards in the current state), this output can function as an excellent statistic for what the model considers important. However, this absolute

value needs to be put into context to provide useful information. As such, we define the importance of a state as the difference between two value estimates. That is, we compare the value of state s_{t+H}, where H is a predefined horizon of steps, with a value estimate of the same time step, when we have taken random steps in the time steps between t and $t+H$:

$$I_{t,t+(H-1)} = \frac{|V(s^{\pi}_{t+H}) - V(s^{\gamma}_{t+H})|}{H} \tag{16}$$

Here $V(s^{\pi}_{t+H})$ and $V(s^{\gamma}_{t+H})$ are the value estimates after acting on the actual policy π and a random policy γ respectively, for H steps. We average the final calculated importance over the entire horizon of timesteps. The entire process is schematically visualized in Figure 9a.

We take H steps before measuring I because V-estimates may attribute a sudden increase or decrease in V to only a single step. For example, a self-driving car closing in on a wall may only drop V on the first time step where it is too late to correct a mistake and hit the wall. However, due to an increased risk, we would typically also attribute importance to the steps leading up to the mistake. Granted, some methods may already incorporate risk in their V-estimates, especially in *on-policy* algorithms. In these methods, we may want to decrease H, or even set it to 1, to achieve an increased accuracy of I. However, this comes at the cost of computational requirements due to the need to copy the environment state at each time step. Furthermore, even on-policy methods likely never incorporate all the risk in their V estimate that our random agent would estimate.

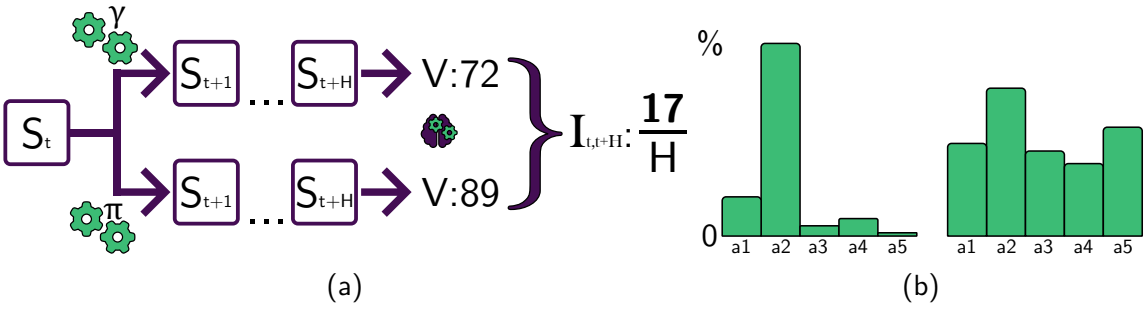

Figure 9: On the left (a), our value-based method for finding state importance I_t. At timestep t, we perform two rollouts of H steps. One rollout performs regular steps with policy π and the other rollout performs random steps with random policy γ. After H steps, the trained value head of the neural network evaluates both states and outputs V. The absolute difference between these values, divided by H, is then attributed as the state importance for the time steps t to $t+H$. On the right (b), there are two action distributions. The leftmost action distribution is of low entropy (decisive) and is attributed with high importance by the regular policy-based method. In contrast, the rightmost action distribution is of high entropy and indicates low importance.

Policy-based State Importance For measuring the importance of states through the policy output of a model, we may use the definition by Huang et al. [39] which measures the entropy of the output distribution (Figure 9b). In Huang et al., this entropy is compared to some threshold to indicate whether a state is critical. Because our work focuses on importance scores instead of a binary answer, the inverse of the normalized entropy could be used to indicate state importance:

$$I_t = 1 - \overline{\mathcal{H}(\pi(\cdot, s))} \tag{17}$$

Here, we normalize the entropy (indicated by the bar) over a single episode. However, the entropy of the output distribution only indicates the *decisiveness* of the agent. Although decisiveness may indeed indicate importance (a fireman will act decisive in case of a fire), we may also act decisive in trivial situations, and so can RL agents. Due to this, in addition to measuring the decisiveness of the agents, we also propose to measure importance as a change in entropy. We calculate this by summing over the current entropy minus H previous entropy values and averaging this final value:

$$I_t = \frac{\sum_{i=0}^{H} |\mathcal{H}(\pi(\cdot|s_t)) - \mathcal{H}(\pi(\cdot|s_{t-i}))|}{H} \tag{18}$$

By taking the absolute value, this method finds states where the agent suddenly has to act decisively (likely indicating an important state) or suddenly stops being decisive (indicating an end to importance or perhaps some form of surprise). Arguably, the latter of the two is not as significant and we may not even consider these importance values. Nevertheless, we believe that there is still information present in these circumstances that a human may be interested in and thus have chosen to include both in this study.

In general, our state importance methods all produce *historical data explanations* by summarizing local *intuitive explanations*. Although we discuss state importance in this work, our complete pipeline used in this thesis averages state importances over multiple episodes. As such, our final results are more akin to time step importance values and can in turn be calculated once and then used in future explanations. To get a reasonable average of the state importance per time step in the high-variance multi-agent RL environments, we average over 10 episodes. This allows the implementation to be easily parallelizable.

5.3 Feature Importances in Deep RL

This section details the SHAP and LIME methods [9, 10], two model-agnostic tools for finding local feature importances, and the manner in which they are used in our work. We will strengthen the outputs generated by both SHAP and LIME by the state importances we have calculated in the previous section.

Local Interpretable Model-agnostic Explanations (LIME) [10] assumes that all complex models are linear in their locality, similar to how regular functions are linear in their

locality. With this assumption, LIME works by fitting a simpler interpretable model to this local state and uses this interpretation as a local explanation. LIME aims to minimize the following formula:

$$\underset{g \in G}{\text{argmin}} \mathcal{L}(f, g, s) \tag{19}$$

Which is a measure of how well the simpler interpretable model g describes the complex model f in the local state s. Here, G is the set of all possible linear models. LIME requires a set of background data, which we can obtain by simply sampling the environment. From this data, we obtain the mean and standard deviations for each feature. Equation 19 is then minimized by obtaining a large set of perturbed data points, which (for our numerical features) are drawn from a normal distribution with mean and standard deviation according to our data. We then obtain black-box prediction labels for these new data points. LIME is then able to explain state s, by fitting a simple linear regression model to the perturbed data and black-box labels, while weighing the data points by their distance to s. By default, LIME uses Euclidean distance and Ridge regression [51] as the interpretable model. The coefficients produced by this linear regression method are used as the interpretation.

Shapley Additive explanations (SHAP) [9] is a method rooted in game theory to measure the importance of features by *removing* a feature in a state and measuring the difference in the outcome in this new altered state. Shapley values [52] measure the impact of players on the outcome of a game. In machine learning, the game is a prediction, and the players are the input features. Intuitively, Shapley values are calculated by continuously adding features and measuring the difference between adding feature x and the average gain of adding any feature. Because features are often not completely uncorrelated, the order in which these features are added for measurement matters. As such, with a high number of features, the calculation of exact Shapley values becomes extremely computationally demanding. Furthermore, Shapley values require us to add and remove features from the input data, while most models are not capable of changing the input shape after training.

Lundberg et al. [9] proposed different methods to approximate Shapley values more efficiently. In most of these methods, the problem of removing features is addressed by again retrieving a background dataset and for each *removed* feature, sample a random value of the same feature from this background dataset. One of the proposed methods, KernelSHAP is surprisingly similar to LIME. As discussed, instead of creating a new dataset of perturbed data points, KernelSHAP creates its new data by creating new data points where one or more of the data points are *removed*. As is the case in LIME, KernelSHAP then fits a linear regression model on this new data to explain a single state s. The main difference between LIME and KernelSHAP lies in the weighting of the altered samples. Because data points where most features (or close to no features) are removed, can provide us with the most information about a particular feature, these data points are weighted more heavily. This then allows for an additional optimization step by first creating data points with $F-1$ or $F-(F-1)$ features removed, where F is the number of features. Depending on the budget, we may then elect to create more samples that

will have a lower weight associated with them. Even with the added optimization steps in SHAP, approximating Shapley values still proves to be computationally demanding. However, Shapley values have a strong theoretical foundation which may prove to be a requirement if more AI regulations get enforced.

We use both LIME and SHAP to generate *current data explanations* in order to address the *discriminatory policy issue* and in part the *environment validation issue*. To strengthen these methods, we combine SHAP and LIME outputs with our previous method of state importance by weighing the outputs of each state s by the state importance I_{s_t}. This is done because SHAP and LIME both calculate feature importance scores for a single state while not considering whether a state is important or not. By combining the methods and reweighting the feature importance scores, we should obtain a better representation of feature importance, since important features are even more important when in important states. Furthermore, presenting feature importances in environments with many features will, by default, result in a large list not easily understood by humans. We hope that our method of combining feature importance with state importance eliminates some otherwise important features, reducing the list of *important* features.

5.4 Results

Figure 10 displays our results for our value-based and policy-based methods for finding state importance values I. Produced importance scores are first normalized between 0 and 1 and then averaged over 10 episodes. The horizon H is set to 5 for all agents and 1 for the planner, as the planner can only act every 100 time steps.

In the value-based approach, we observe how the state importance values tend to decrease over time. This is expected behavior in our episodic economic environment. Early states may be comparable to someone's early life, where all decisions may still have a long-term effect on utility. Near the end of someone's life, these effects are expected to be less long-term, and their importance will be valued less (given that the agent only considers its own utility). Although this highlights yet more real-life economic behavior in our agents, one may wish to correct for this trend in other problems. Notably in poorer agents, such as Agent 14, we tend to observe a slight increase in value-based state importance during the first few hundred time steps. These agents are primarily concerned with buying and selling goods and, as such, likely have many more opportunities after a few hundred time steps. This is because other agents will have plenty of goods to trade by then and the market is likely fully stocked, potentially benefiting traders.

We observe how the regular policy-based method has its flaws in Agent 14. Here, we see a large shift in action distribution entropy around the 200th time step. Around this time, trading agents tend to switch to a trading strategy that should be marked as an important state. Our policy-based difference method better estimates these situations by measuring when an agent has to respond to an (important) situation, rather than whether the agent acts decisively.

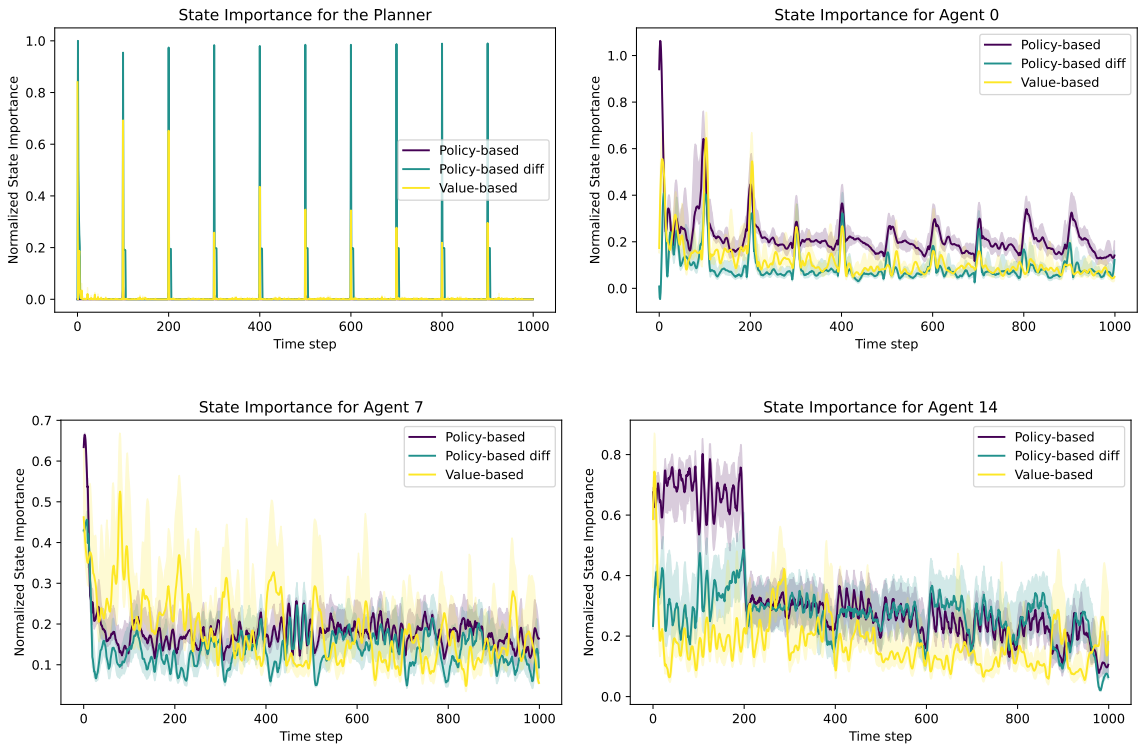

Figure 10: State importance for the planner and various worker agents as estimated by our methods. The importance scores are first normalized between 0 and 1 in their respective runs and then averaged over 10 episodes. The upper and lower standard deviations over the 10 episodes are highlighted. Horizon H is set to 1 for the planner agent, and to 5 for the other agents. The state importance values for the planner are as expected, with high importance only being attributed to states in which taxes can be set. For the agents we observe that these same states are often important as well, but to what extent differs per agent. The results of the worker agents are smoothed for an improved visual interpretation in these figures, but non-smoothed results are used for further processing.

When inspecting which time steps are deemed important, we can see that the results are as we would expect them for the planner agent. Each 100th-time step, when the planner is able to act, is marked as important. The policy-based methods make no distinction between each tax year, but the value-based method does demonstrate the previously described linearly decreasing trend over time. Again, this is the result of the planner agent valuing later time steps less because there is less chance of long-term utility effects and because of the lower V outputs, the changes in V will also be lower. Worker agents have a more diverse state importance output, but to spot a trend, we again have to look at the time step in which taxes are set and collected. Clearly, these are important time steps as they will have a significant effect on the worker's utility and their next actions.

This importance is most pronounced in Agent 0 (generally the richest agent / highest build skill) and to a lesser extent in Agent 14 (generally the poorest agent / lowest build skill). As previously described, the policy-based difference method also shows how trading agents tend to start trading around the 200th time step.

Feature importance scores Next, we measure the feature importance values of our planner and worker agents with both the SHAP and the LIME method [9, 10]. We then re-weight the feature importance values by their state importance.

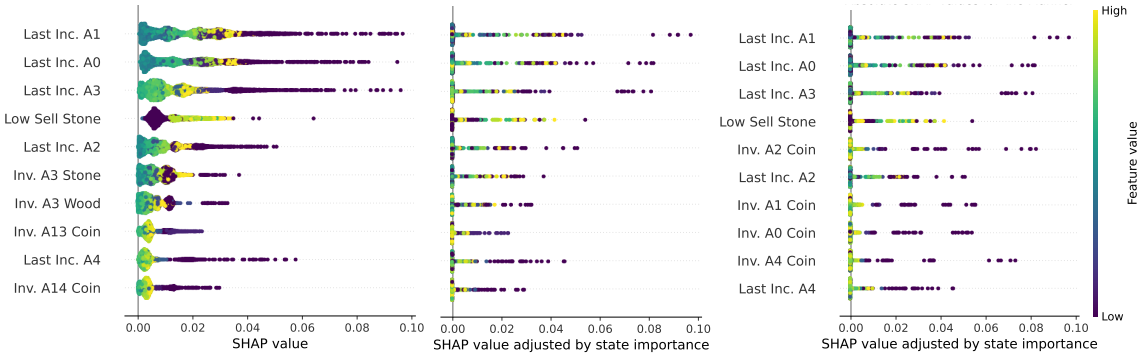

Figure 11: on the left, SHAP values of the planner agent for its 10 most important features (sorted by the SHAP values). In the middle, the same features with their SHAP values reweighted by state importance. The right image shows the new 10 most important features after reweighting the features. The SHAP values displayed are only calculated based on the actions that the planner has taken. Because the planner takes multiple actions simultaneously (one for each tax bracket), the absolute SHAP values for each action are used and then averaged for the final displayed result. The reweighting of the SHAP values significantly decreases the number of important instances, allowing for a clearer visualization. Generally, we observe the previous incomes (Prev. Inc.) and the inventory of coins (Inv. Coin) to be most important. This intuitively makes sense as they are directly linked to both productivity and equality. Nevertheless, we observe the SHAP values to be extremely low in general, indicating that the planner does not incorporate much information from any input features into its decision.

Figure 11 shows SHAP values for the planner agent, both before and after reweighting the SHAP values with state importance. As a final importance score, we have added all three importance scores together, weighing each of our three state importance methods equally. We observe how the reweighting of the SHAP values significantly decreases the number of important instances, resulting in a clearer visualization. Furthermore, while the original results may show how the planner attends to the poorer agents (such as A13 and A14), after reweighting we observe that most attention is actually paid to only the richest agents. However, we note that, according to the SHAP values, all features contribute very small amounts to the outputs for the planner. While we can conclude

that the model is likely not discriminatory against any of the agents, a policymaker would still likely not want to use this model as all input features are barely considered. Future work may look into different input features for the planner in order to hopefully improve on this issue. For example, instead of an input for each individual worker agent, we may generate general statistics about the population, such as the number of agents that belong in a specific tax cutoff bracket.

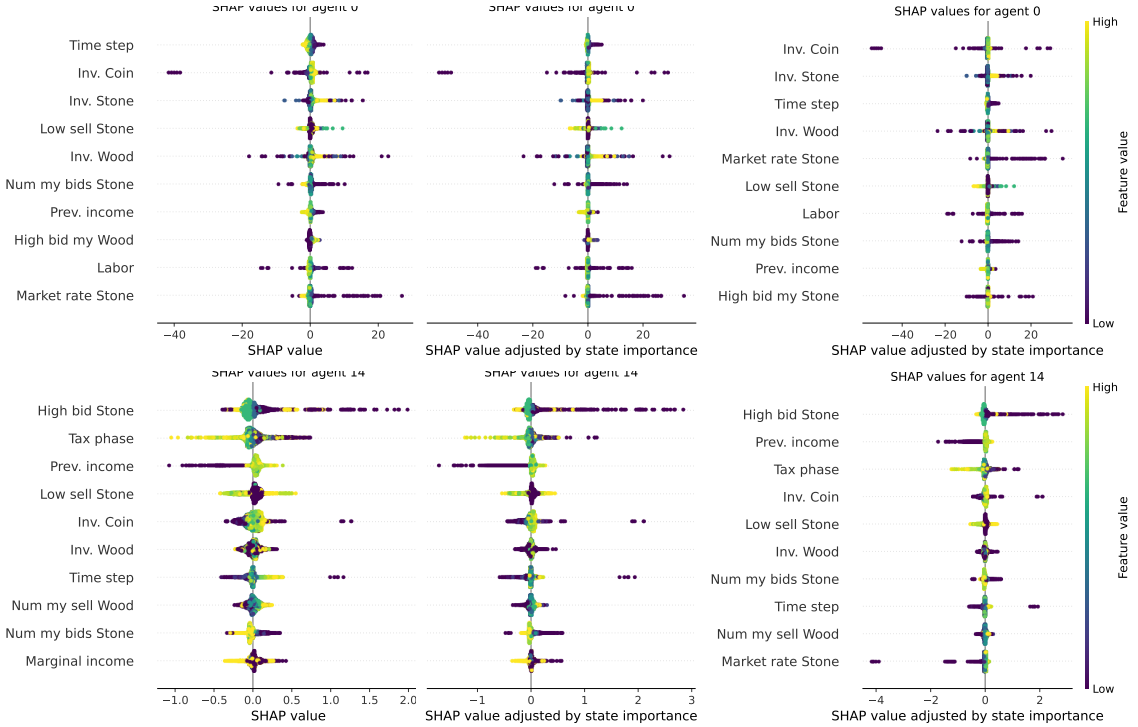

Figure 12: on the left, SHAP values of the richest and poorest agents (A0 and A14) for their 10 most important features (sorted by the SHAP values). In the middle, the same features with their SHAP values reweighted by state importance. The right image shows the new 10 most important features after reweighting the features. The SHAP values displayed are only calculated based on the actions that the agents took. The reweighting of the SHAP values decreases the number of important instances, allowing for a clearer visualization.

In Figure 12 we have displayed SHAP values for both the generally richest agents (highest build skill) and the poorest agent (lowest build skill). Again, after reweighting the SHAP values, our most important features are reordered according to the new values, resulting in a more fair representation. Whereas the current time step appeared to be the most important feature for Agent 0, reweighting for importance reveals that the number of coins instead has a larger impact on its actions. This better aligns with our human intuition of what the agent should consider when deciding on its actions, increasing trust in the simulation. The number of coins in inventory is less important to Agent 14, which

primarily trades and as such highly considers the current highest bid for stone on the market. In general, we observe that the SHAP values for the richer agents are much higher compared to the poorer agents. This intuitively makes sense as the richer agent's actions likely have larger consequences. However, the effect seems overblown, likely indicating an unrealistic utility function was used as the agent reward function.

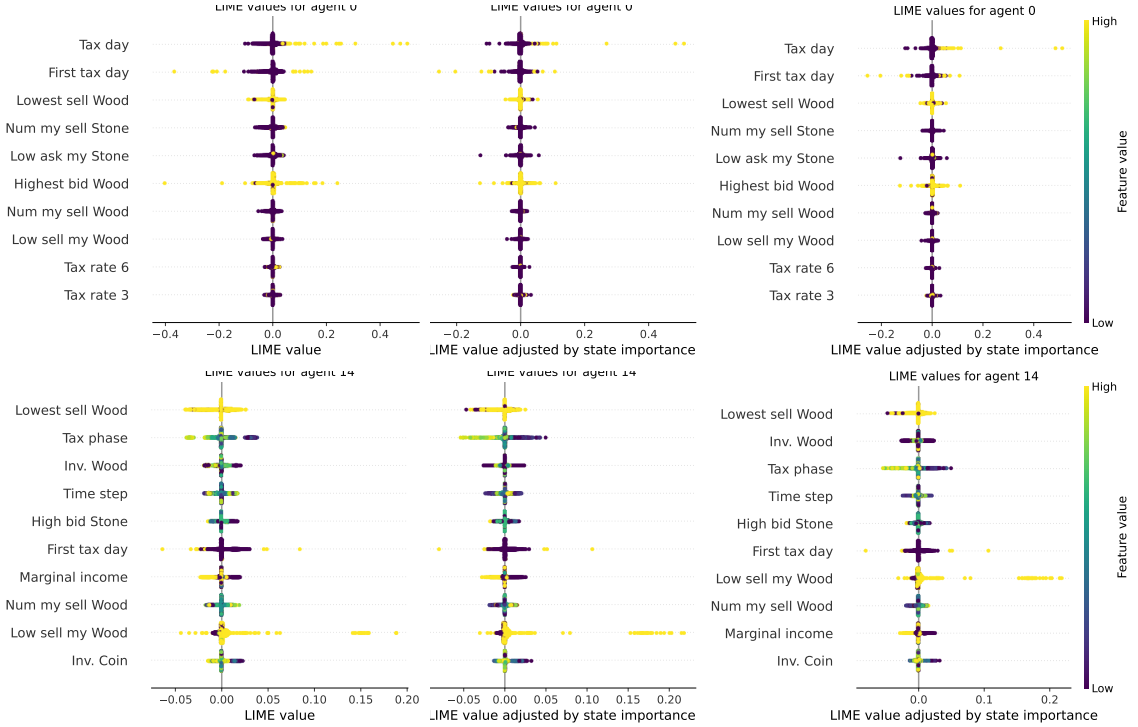

Figure 13: on the left, LIME values of the richest and poorest agents (A0 and A14) for their 10 most important features (sorted by the LIME values). In the middle, the same features with their LIME values reweighted by state importance. The LIME values displayed are only calculated on the basis of the actions that the agents took. The reweighting of the LIME values decreases the number of important instances, allowing for a clearer visualization, although the effect is less pronounced compared to the SHAP visualization in Figure 12.

Figure 13 shows the LIME values for the same two agents (A0 and A14). While not shown, the planner agent's LIME values indicate a zero or near-zero importance for all features, as was also the case in the SHAP results. For the agents, we can see a smaller effect on the LIME values after adjusting the values by state importance. This is due to LIME not attributing a value to each of the features for every local explanation. Instead, only the top few features (in each individual explanation) are considered and given a LIME score, resulting in a lot of 0 score instances. With many 0 feature scores already present, adjusting for state importance has a much smaller effect. Furthermore, when comparing the LIME scores to the SHAP scores, we observe a very different set of top

10 important features compared to our SHAP method. This is an unfortunate result, as the two methods currently do not validate each other, which would have boosted trust in the system.

6 Limitations & Discussion

RL-assisted policy generation has shown to be a promising field for being able to quickly generate and validate new governmental policies. However, before being truly implemented in real-world decision making, existing simulations require a greater amount of realism and more scalable implementations. Although we believe our work to be a step in the right direction when it comes to scalability, real-world applications would likely require simulating populations of thousands of agents. For these population sizes, our setup still requires weeks, if not months, of training due to the processing complexity of the step function. Yet, we do believe reinforcement learning may play a role in governmental policy creation and/or validation, possibly in the not-too-distant future. This is because creating and validating tax policies, as done in our work, requires a simulated national economy. Although simulating a true economy may not be possible in the near future, if ever, many policy creation settings may not have this demanding requirement. Furthermore, as the world becomes more connected and complex, we may see an increase in demand for such hyper-realistic environments, as traditional models will simply be inadequate.

Beyond creating a more realistic environment, by, for example, expanding the agent's action space to allow for a more diverse population, future works may investigate more possibilities for the planner agent. Currently, the planner creates unrealistic and erratic policies. Instead, future iterations should penalize the planner for large changes in policies and perhaps be forced to create only some kind of policies (for instance only progressive tax systems). However, as pointed out previously, providing the RL planner with more freedom may produce new original policies that would otherwise not have been considered. As was also apparent from our explainability experiments, the planner barely considered any features, and this leads to distrust in the system as we cannot verify whether the planner is merely creating semi-random policies. Future work should improve the planner, namely its input features. Currently, the planner observes various features of each agent individually, but these values may be better grouped into population statistics.

To better address the explainability questions posed to the real policymaker, new methods must be better able to address the issues of environment validation and of understanding the policy. Our work provides some tools to shed light on the behavior of the simulated agents. This, perhaps together with the proposed methods of Sequeira and Gervasio [42] allows for some environment validation, but we still end up with a lot of data that is not easily understood by humans. As such, further experimentation with different visual presentation methods is required, perhaps incorporating methods such as GroupShapley [53] to reduce the number of features in a single explainability visualization. In the

future, more complex environments likely require many more tools for better insights. With currently proposed methods, we do have access to methods to visualize which input features are important to the RL planner. This, together with the fact that RL does not incur an inherent bias through training data, provides strong tools to address the discriminatory or biased RL policy issue. However, such issues may still occur through a biased reward function, and, as such, these reward functions must always be transparent. Methods to truly understand the created policy have yet to be created in future work. These solutions are highly sought after because a policymaker requires reasoning about its proposed policy, perhaps even presented in natural language. It is possible that mimicking methods could be effective in this situation, as the governmental policy that is generated may be simple to describe, and only the means to arrive at such a policy are not, due to the moving objectives in a highly complex multi-agent environment during training.

Future work in our explainability pipeline in general should incorporate all methods in a comprehensive, easy-to-use toolkit that researchers can easily use to explain their own models. As pointed out in the work of Vourous, such a general toolkit for explainability in DLR is currently missing. We eagerly await the release of the toolkit created by Sequeira and Gervasio [42], to see if this toolkit is sufficient and can be expanded on. In contrast to their work and other similar previous works, our (value-based) method relies heavily on being able to take steps in a copy of the environment. Although in our experiments, finding state importances over averaged over 10 episodes did not take more than 20 minutes, relying on copying the environment likely does not scale well to more complex environments. As such, alternatives or more efficient approaches need to be investigated in future work.

To further improve our pipeline before creating a general toolkit, we should broaden its applicability. Currently, our state importance methods rely on the environment being episodic with a fixed time step length. Our value-based method also disregards any rewards obtained during the horizon H rollout. Because of this, when high rewards are obtained in the real rollout, and not obtained in the random rollout (but removed from V, because they can no longer be obtained), importance I is set to 0, because the V estimates do not differ. While this is not a problem in our environment (as long as H is reasonably small), due to the smooth reward function, future work should incorporate an extra importance method that also compares the obtained rewards during the rollout. Furthermore, while our work is applicable to most, if not all, current model-free deep RL methods, we currently do not support model-based algorithms [54]. Because these models create their own model of the environment, these algorithms may yet prove to play a key role in explainable RL.

We do believe our novel method of amplifying feature importance scores in important states presents a more truthfull representation of important features. Future work should focus on exploiting this further and validate the method on smaller and simpler environments that should, in contrast to our economic simulation, be more suitable for validation of the methods. In these simpler environments, we could then better compare the dif-

ferent state importance methods against baseline methods and determine which weights to use in specific situations as final importance scores. Such an improved validation is another requirement before working on a fully-fledged toolkit.

On a final note, we would like to address the ethical considerations when it comes to AI-assisted policy generation. Any production-grade product for creating and/or validating (governmental) policies may be misused, whether with intent or out of ignorance, and cause undesired consequences that are hidden behind an RL system. We do not believe that our work is a sufficient leap forward in AI-assisted policy generation, whereby it would benefit from a new ethical review, as already conducted in the AI economist paper [4]. Nevertheless, we hope that future research will take explainability considerations into account, reducing the risk of misuse or, at least, increasing transparency.

7 Conclusion

In this thesis, we have recreated the work of the AI economist [4], a framework for building and validating tax policies with reinforcement learning. We have altered the original framework and built a new environment as a vector-based environment, rather than a simulation in a 2D world. We have shown how this simplified vector-based environment demonstrated the same real-life economic behavior. While demonstrating the same behavior, training was much faster in our environment, indicating that this is an improved and more scalable direction for future work in RL-assisted policy generation.

We further combined novel and existing explainability methods in a pipeline to investigate feature importance and state importance in our RL agents. Feature importances were calculated using the existing SHAP and LIME [9, 10] methods and then adjusted by the importance of the state in which the feature importance values were calculated. This provided a clearer representation of the important features and should provide a fairer representation of the most important features. Unfortunately, the latter of the described benefits could not be properly evaluated because the two methods disagreed on important features or found all features to have an importance of near 0. This indicates that a policymaker would likely not trust the current system to aid in its (governmental) policy design. However, having this fact transparent is already a step forward in the path to RL-assisted policy generation that is both beneficial and explainable.

References

[1] R. E. Lucas Jr, "Econometric policy evaluation: A critique," *Carnegie-Rochester conference series on public policy*, vol. 1, pp. 19–46, 1976.

[2] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," Mar. 2022. arXiv:2203.02155 [cs].

[3] T. Zhang, A. Williams, S. Phade, S. Srinivasa, Y. Zhang, P. Gupta, Y. Bengio, and S. Zheng, "AI for Global Climate Cooperation: Modeling Global Climate Negotiations, Agreements, and Long-Term Cooperation in RICE-N," Aug. 2022. arXiv:2208.07004 [cs].

[4] S. Zheng, A. Trott, S. Srinivasa, D. C. Parkes, and R. Socher, "The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning," *Science Advances*, vol. 8, p. eabk2607, May 2022. Publisher: American Association for the Advancement of Science.

[5] A. Trott, S. Srinivasa, D. van der Wal, S. Haneuse, and S. Zheng, "Building a Foundation for Data-Driven, Interpretable, and Robust Policy Design using the AI Economist," Aug. 2021. arXiv:2108.02904 [cs, econ, q-fin].

[6] G. A. Vouros, "Explainable Deep Reinforcement Learning: State of the Art and Challenges," *ACM Computing Surveys*, vol. 55, pp. 1–39, May 2023.

[7] S. Milani, N. Topin, M. Veloso, and F. Fang, "A Survey of Explainable Reinforcement Learning," Feb. 2022. arXiv:2202.08434 [cs].

[8] A. Krajna, M. Brcic, T. Lipic, and J. Doncevic, "Explainability in reinforcement learning: perspective and position," Mar. 2022. arXiv:2203.11547 [cs].

[9] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 1135–1144, Association for Computing Machinery, Aug. 2016.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*. MIT Press, Nov. 2018. Google-Books-ID: uWV0DwAAQBAJ.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Dec. 2013. arXiv:1312.5602 [cs].

[13] V. Konda and J. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems*, vol. 12, MIT Press, 1999.

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," June 2016. arXiv:1606.01540 [cs].

[15] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Statistics Surveys*, vol. 16, pp. 1–85, Jan. 2022. Publisher: Amer. Statist. Assoc., the Bernoulli Soc., the Inst. Math. Statist., and the Statist. Soc. Canada.

[16] C. Rudin and K. L. Wagstaff, "Machine learning for science and society," *Machine Learning*, vol. 95, pp. 1–9, Apr. 2014.

[17] A. M. Roth, N. Topin, P. Jamshidi, and M. Veloso, "Conservative Q-Improvement: Reinforcement Learning for an Interpretable Decision-Tree Policy," July 2019. arXiv:1907.01180 [cs].

[18] A. Silva, M. Gombolay, T. Killian, I. Jimenez, and S.-H. Son, "Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pp. 1855–1865, PMLR, June 2020. ISSN: 2640-3498.

[19] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, pp. 1140–1144, Dec. 2018. Publisher: American Association for the Advancement of Science.

[20] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 982–987, Aug. 2023. Number: 7976 Publisher: Nature Publishing Group.

[21] S. Lo Piano, "Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward," *Humanities and Social Sciences Communications*, vol. 7, pp. 1–7, June 2020. Number: 1 Publisher: Palgrave.

[22] M. Ashoori and J. D. Weisz, "In AI We Trust? Factors That Influence Trustworthiness of AI-infused Decision-Making Processes," Dec. 2019. arXiv:1912.02675 [cs].

[23] E. Puiutta and E. M. Veith, "Explainable Reinforcement Learning: A Survey," May 2020. arXiv:2005.06247 [cs, stat].

[24] Nicolás Garrido, N. Garrido, and L. Mittone, "An agent based model for studying optimal tax collection policy using experimental data: The cases of Chile and Italy," *Journal of Socio-economics*, vol. 42, pp. 24–30, Feb. 2013. MAG ID: 2079252372.

[25] K. M. Bloomquist and Kim Bloomquist, "Tax Compliance as an Evolutionary Coordination Game: An Agent-Based Approach," *Public Finance Review*, vol. 39, pp. 25–49, Jan. 2011. MAG ID: 2059969213.

[26] S. Zheng, A. Trott, S. Srinivasa, N. Naik, M. Gruesbeck, D. C. Parkes, and R. Socher, "The AI Economist: Improving Equality and Productivity with AI-Driven Tax Policies," Apr. 2020. arXiv:2004.13332 [cs, econ, q-fin, stat].

[27] S. Zheng, A. Trott, S. Srinivasa, D. C. Parkes, and R. Socher, "The AI Economist: Optimal Economic Policy Design via Two-level Deep Reinforcement Learning," Aug. 2021. arXiv:2108.02755 [cs, econ, q-fin].

[28] R. Koster, J. Balaguer, A. Tacchetti, A. Weinstein, T. Zhu, O. Hauser, D. Williams, L. Campbell-Gillingham, P. Thacker, M. Botvinick, and C. Summerfield, "Human-centred mechanism design with Democratic AI," *Nature Human Behaviour*, vol. 6, pp. 1398–1407, Oct. 2022. Number: 10 Publisher: Nature Publishing Group.

[29] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh, "Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution," Apr. 2020. arXiv:1912.12191 [cs].

[30] W. Shi, Z. Wang, S. Song, and G. Huang, *Self-Supervised Discovering of Causal Features: Towards Interpretable Reinforcement Learning*. Mar. 2020.

[31] R. M. Annasamy and K. Sycara, "Towards Better Interpretability in Deep Q-Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4561–4569, July 2019.

[32] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez Rezende, "Towards Interpretable Reinforcement Learning Using Attention Augmented Agents," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[33] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara, "Transparency and Explanation in Deep Reinforcement Learning Neural Networks," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, (New Orleans LA USA), pp. 144–150, ACM, Dec. 2018.

[34] S. G. Rizzo, G. Vantini, and S. Chawla, "Reinforcement Learning with Explainability for Traffic Signal Control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3567–3572, Oct. 2019.

[35] K. Zhang, P. Xu, and J. Zhang, "Explainable AI in Deep Reinforcement Learning Models: A SHAP Method Applied in Power System Emergency Control," in

2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2), pp. 711–716, Oct. 2020.

[36] K. Zhang, J. Zhang, P.-D. Xu, T. Gao, and D. W. Gao, "Explainable AI in Deep Reinforcement Learning Models for Power System Emergency Control," *IEEE Transactions on Computational Social Systems*, vol. 9, pp. 419–427, Apr. 2022. Conference Name: IEEE Transactions on Computational Social Systems.

[37] D. Beechey, T. M. S. Smith, and . Şimşek, "Explaining Reinforcement Learning with Shapley Values," June 2023. arXiv:2306.05810 [cs].

[38] D. Amir and O. Amir, "HIGHLIGHTS: Summarizing Agent Behavior to People," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, (Richland, SC), pp. 1168–1176, International Foundation for Autonomous Agents and Multiagent Systems, July 2018.

[39] S. H. Huang, K. Bhatia, P. Abbeel, and A. D. Dragan, "Establishing Appropriate Trust via Critical States," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3929–3936, Oct. 2018. ISSN: 2153-0866.

[40] P. Sequeira and M. Gervasio, "Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations," *Artificial Intelligence*, vol. 288, p. 103367, Nov. 2020.

[41] P. Sequeira, J. Hostetler, and M. Gervasio, "Global and Local Analysis of Interestingness for Competency-Aware Deep Reinforcement Learning," Nov. 2022. arXiv:2211.06376 [cs].

[42] P. Sequeira and M. Gervasio, "IxDRL: A Novel Explainable Deep Reinforcement Learning Toolkit based on Analyses of Interestingness," July 2023. arXiv:2307.08933 [cs].

[43] K. Boggess, S. Kraus, and L. Feng, "Toward Policy Explanations for Multi-Agent Reinforcement Learning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, (Vienna, Austria), pp. 109–115, International Joint Conferences on Artificial Intelligence Organization, July 2022.

[44] E. Saez and S. Stantcheva, "Generalized Social Marginal Welfare Weights for Optimal Tax Theory," *The American Economic Review*, vol. 106, pp. 24–45, Jan. 2016. MAG ID: 1559593144.

[45] D. M. Johnson, "Applications of the standard-score IQ to social statistics.," *Journal of Social Psychology*, vol. 27, pp. 217–227, May 1948. Num Pages: 11 Place: Worcester, Mass., United States Publisher: Clark University Press.

[46] C. Gini, *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.[Fasc. I.]*. Tipogr. di P. Cuppini, 1912.

[47] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 3053–3062, PMLR, July 2018. ISSN: 2640-3498.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017. arXiv:1707.06347 [cs].

[49] A. E. R. Prince and D. Schwarcz, "Proxy Discrimination in the Age of Artificial Intelligence and Big Data," *Iowa Law Review*, vol. 105, p. 1257, 2019.

[50] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992.

[51] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970. Publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality].

[52] L. S. Shapley, "A Value for n-Person Games," in *17. A Value for n-Person Games*, pp. 307–318, Princeton University Press, 1953.

[53] M. Jullum, A. Redelmeier, and K. Aas, "groupShapley: Efficient prediction explanation with Shapley values for feature groups," June 2021. arXiv:2106.12228 [cs, stat].

[54] A. Plaat, W. Kosters, and M. Preuss, "Deep Model-Based Reinforcement Learning for High-Dimensional Problems, a Survey," Dec. 2020. arXiv:2008.05598 [cs].