Department of Computer Science

BSc (Hons) Computer Science (Artificial Intelligence)

Academic Year 2022 - 2023

Inhibiting Emergent Selfishness in Self-Interested Reinforcement Learning-based Agents

Philipp E. Bibik

1943513

A report submitted in partial fulfilment of
the requirements for the degree of Bachelor of Science

Brunel University London
Department of Computer Science
Uxbridge
Middlesex
UB8 3PH
United Kingdom
T: +44 1895 203397
F: +44 (0) 1895 251686

Abstract

Reinforcement learning (RL) is a powerful machine learning technique where agents learn optimal policies through interactions with their environment based on set goals. However, in certain environments, the mathematically optimal policy has unintended side-effects like selfish behaviour or disregard for other agents. This dissertation examines such problematic RL scenarios modelled in Gridworld environments. Three techniques are evaluated to mitigate undesirable agent behaviors: having agents internalise peer reward functions, making environment-specific reward function modifications, and using inverse reinforcement learning (IRL) to infer peer reward functions. The techniques are tested across four Gridworlds covering positive and zero-sum games with varying incentives. While each technique has advantages, the IRL-based approach stands out as the most effective with the fewest constraints. The study provides insights into mitigating problematic behaviors of RL agents seeking optimal policies. Key techniques like internalising peer rewards and using IRL show promise for enabling more collaborative, ethical agent behaviors.

Acknowledgements

I would like to extend my sincere appreciation and thanks to my supervisor, Dr. Alina Miron, for her support, encouragement, and guidance that have been instrumental in my research journey. I also extend my heartfelt gratitude to my family and friends, whose support and encouragement have been a source of strength for me throughout this journey. I am especially thankful to my beloved partner for being a constant source of motivation and inspiration in my life. Without everyone's support, this research would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial intelligence (AI) has been increasingly integrated into various aspects of society, transforming industries and altering the way we perceive and interact with technology. As AI systems continue to evolve, becoming more complex and sophisticated, the risks and ethical concerns associated with these technologies demand urgent attention.

Reinforcement Learning (RL) is a technique from AI that learns by performing actions and being rewarded for their quality. This technique is useful in situations in which an agent is expected to make choices instead of generating or categorising data. A common use case is recommender systems, employed ubiquitously on social media platforms, where a nominal goal, such as increasing engagement or user time spent on the platform, is given to the agent and they learn to pick the data best suited towards solving that goal.

A critical issue arises when the environment and set goals encourage undesirable behaviour, such as selfishness, that the agent is predisposed to learn if it proves to be the most effective strategy. Such undesired effects can already be observed to have harmful effects on society as discussed in Section 2.2.1: Recommender systems for instance have a significant role in exacerbating teen social media addiction as the algorithms learn to 'hook' users. They can also amplify biased and sensationalised news, as they learn to exploit the human negativity bias. These outcomes exist because the recommender systems do not consider factors beyond their objectives, such as the well-being of individuals and society at large.

Of particular interest are scenarios in which an RL-based agent can directly interact with the world and perform actions which can affect others, be they agents or humans. Currently, this is rare, as applications of RL are mostly found in robotics and self-driving, but if the techniques continue becoming more powerful and are used in more contexts with greater autonomy, it may cause problems, because of the tendency of agents to optimise for optimal strategies, regardless of side effects.

This project is concerned with simplified scenarios in which the optimal strategy is known and expected to result in selfish behaviour towards other agents, and exploring approaches to how this selfishness can be inhibited. While the results are not expected to translate to complex real-world scenarios, they aim to serve as a building block for future work investigating scenarios found in reality.

1.1 Aims and Objectives

The primary aim of this project is to identify and evaluate effective approaches for inhibiting selfish behaviour in reinforcement learning-based agents, with a focus on environments that inherently encourage such behaviour. To achieve this aim, the following objectives have been established:

1.1.1 Objectives

1. Conduct a comprehensive literature review to gain an understanding of the current state of research on selfish behaviour in reinforcement learning and existing mitigation strategies for unintended behaviour, with a particular focus on selfishness.
2. Design and implement diverse environments that encourage selfish behaviour to provide a robust foundation for evaluating the effectiveness of inhibiting approaches.
3. Train lead agents which exemplify selfish behaviour.
4. Train secondary agents to act as experts, permitting the testing of the inhibition approaches.
5. Develop inhibiting approaches by designing and implementing multiple strategies that focus on reducing selfish behaviour.
6. Train agents through selfish-inhibiting approaches.
7. Evaluate and compare the inhibiting approaches to ascertain their effectiveness in mitigating selfish behaviour.

1.2 Project Approach

To achieve the project's aim and objectives, a systematic approach has been taken, breaking down each objective into actionable tasks. This ensures comprehensive coverage of all aspects of the project, facilitating a clear understanding of the steps required to complete the project successfully. Table 1.1 provides a summary of the project tasks derived from the objectives, outlining the key activities needed to accomplish each objective.

Objective	**Tasks**
1. Literature Review	1.a: Review existing research on selfish behaviour in RL 1.b: Assess current research on the mitigation of unintended behaviour, particularly selfishness 1.c: Investigate promising techniques
2. Environment Design	2.a: Plan environments 2.b: Design environments 2.c: Implement environment 2.d: Test environments
3. Training Selfish Agents	3.a: Train selfish agents 3.b: Collect selfish agent data for analysis
4. Training Secondary Agents	4.a: Train secondary agents to optimise their goal
5. Inhibiting Approaches	5.a: Plan inhibition approaches 5.b: Design inhibition approaches 5.c: Implement inhibition approaches 5.d: Test inhibition approaches
6. Training Inhibited Agents	6.a: Train agents using inhibiting approaches 6.b: Collect inhibited agent data for analysis
7. Evaluation	7.a: Form testable hypotheses 7.b: Plan ways of testing hypotheses 7.c: Analyse data quantitatively 7.d: Evaluate results of analysis 7.e: Draw conclusions about effectiveness

Table 1.1: Summary of project tasks derived from the objectives

1.3 Dissertation Outline

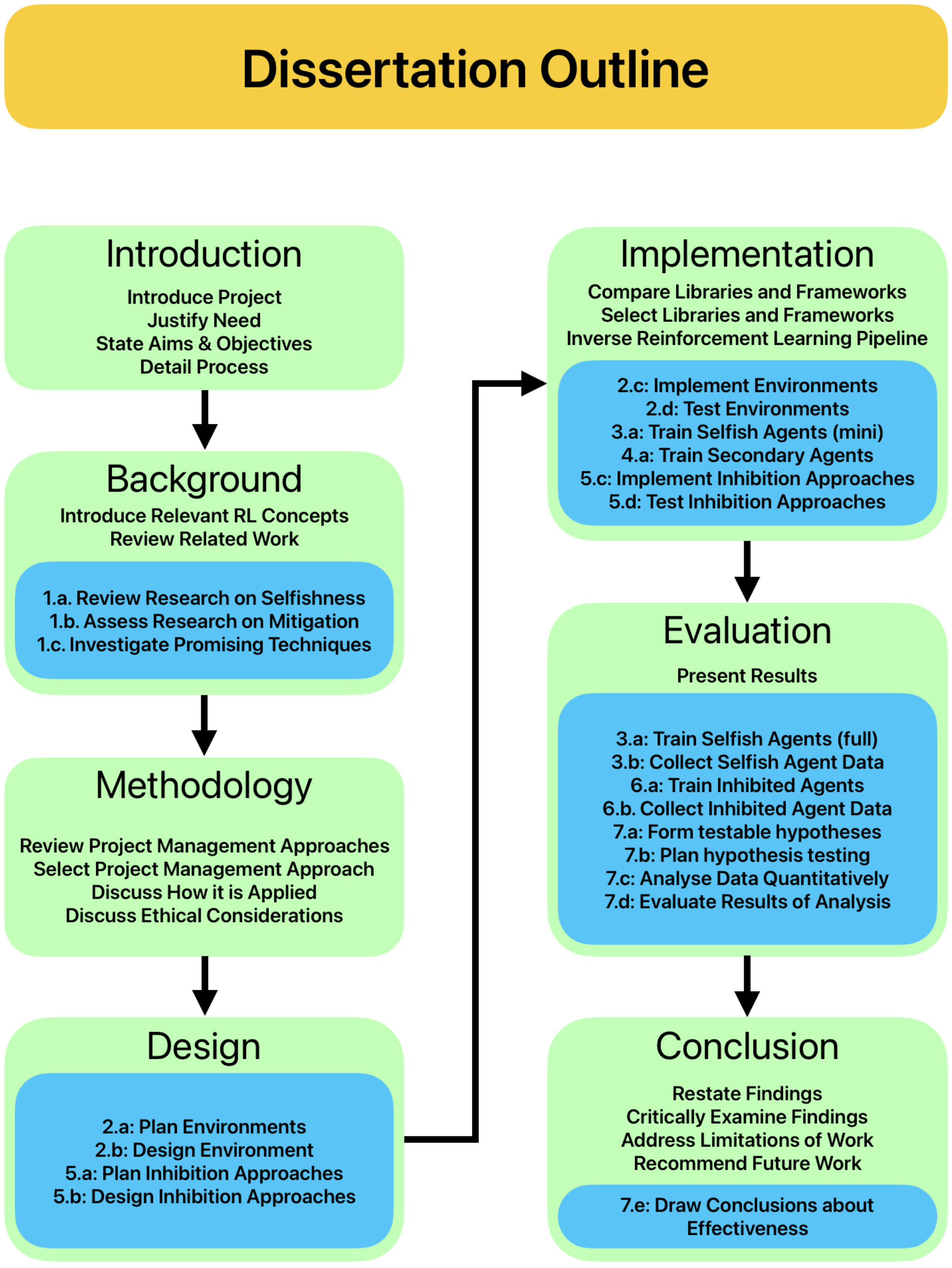
Dissertation Outline
Introduction
Introduce Project
Justify Need
State Aims & Objectives
Detail Process
Background
Introduce Relevant RL Concepts
Review Related Work
1.a. Review Research on Selfishness
1.b. Assess Research on Mitigation
1.c. Investigate Promising Techniques
Methodology
Review Project Management Approaches
Select Project Management Approach
Discuss How it is Applied
Discuss Ethical Considerations
Design
2.a: Plan Environments
2.b: Design Environment
5.a: Plan Inhibition Approaches
5.b: Design Inhibition Approaches
Implementation
Compare Libraries and Frameworks
Select Libraries and Frameworks
Inverse Reinforcement Learning Pipeline
2.c: Implement Environments
2.d: Test Environments
3.a: Train Selfish Agents (mini)
4.a: Train Secondary Agents
5.c: Implement Inhibition Approaches
5.d: Test Inhibition Approaches
Evaluation
Present Results
3.a: Train Selfish Agents (full)
3.b: Collect Selfish Agent Data
6.a: Train Inhibited Agents
6.b. Collect Inhibited Agent Data
7.a: Form testable hypotheses
7.b: Plan hypothesis testing
7.c: Analyse Data Quantitatively
7.d: Evaluate Results of Analysis
Conclusion
Restate Findings
Critically Examine Findings
Address Limitations of Work
Recommend Future Work
7.e: Draw Conclusions about Effectiveness

Figure 1.1: Outline of the dissertation, showing what chapters cover and how objectives map onto them. Blue boxes contain the tasks directly derived from objectives.

Figure 1.1 shows the outline of this dissertation. Within the figure, there is a flowchart listing all chapters and the topics addressed within each. The primary goal of the structure is to show how each objective has been addressed each objective ensuring that each aspect of the project is thoroughly explored and its results are described. For this, the tasks derived from the objectives have all been placed in the section they most belong in. These can be seen in the blue boxes, with some chapters, like the Background or Conclusion, only addressing a single objective, while others address multiple. Many objectives have also been split up and were covered across multiple chapters, where different aspects have been considered in each.

Beyond the sections directly concerned with describing the work undertaken to achieve the objectives, many auxiliary sections exist which aim to justify the project's existence, impart required background knowledge in the reader, or describe the project management approach taken to achieve the objectives. These can be seen in the chapter boxes, lying outside of the objective-specific blue boxes.

Chapter 2

Background

The motivation for this project stems from the uncertainty of risks and adverse consequences associated with the evolution of Artificial Intelligence (AI). The background section aims to not only equip the reader with the building blocks necessary for understanding the works that were undertaken but also make them appreciate the significance of aligning AI systems with human values.

In this background, we will explore relevant literature and concepts to build a comprehensive understanding of the subject matter. We begin with an introduction to Artificial Intelligence, emphasising Reinforcement Learning and its core concepts. Subsequently, we investigate the issues of AI agent misbehaviour and the problems arising from it. Next, we introduce a specialised sub-field of Reinforcement Learning named Inverse Reinforcement Learning (IRL), which plays an important role in addressing the challenges encountered in this project. Lastly, we investigate related past work to provide context and insights into the developments and contributions made to this field.

2.1 AI and Reinforcement Learning

2.1.1 Introduction to AI

"It is not my aim to surprise or shock you—if indeed that were possible in an age of nuclear fission and prospective interplanetary travel. But the simplest way I can summarize the situation is to say that there are now in the world machines that think, that learn, and that create. Moreover, their ability to do these things is going to increase rapidly until—in a visible future—the range of problems they can handle will be coextensive with the range to which the human mind has been applied." [1]

These words were spoken by AI pioneer Herbert A. Simon in 1957. Despite having been uttered over six decades ago, today they ring truer than ever, capturing the essence of the Artificial Intelligence revolution. AI has evolved into a multifaceted field that seeks to create machines or systems that emulate or surpass human intelligence, employing a diverse array of techniques and methodologies to achieve its ambitious goals.

A common definition of AI refers to its resemblance to human intelligence, particularly its ability to solve tasks that typically require human intelligence [2]. However, the actual definition of AI is much fuzzier, with conflicting schools of thought existing within the field [3]. One reason why pinning down an exact definition is difficult is that defining intelligence itself is equally challenging. An alternative way of defining AI is in terms of the goal-seeking behaviour of agents, capable of achieving objectives effectively and efficiently [4]. The human-centric perspective might lead to the assumption that AI systems with human-level intelligence will naturally adopt human-like goals and aspirations, as described by authors like David Deutsch [5].

In contrast, the Orthogonality Thesis, formulated by Nick Bostrom in 2012, posits that an AI's level of intelligence is independent of its final goals, meaning that higher intelligence does not necessarily imply alignment with human values or objectives [6]. A frequently cited example of that is the concept of the Paperclip Maximiser, also introduced by Bostrom: An agent instructed with the sole goal of creating as many paperclips as possible. The danger lies in the fact that such an agent, in its relentless pursuit of its goal, might ultimately pose a threat to humanity. Since human bodies contain atoms that could be repurposed to make more paperclips, the agent could view us as resources. Furthermore, by eliminating humans, the agent would prevent its own potential shutdown, ensuring it could continue producing paperclips indefinitely [7]. The maximiser's 'intelligence' can be solely defined in terms of its ability to achieve its own goals, as opposed to human goals. This perspective underscores the importance of considering not only the accurate replication of human goals in AI development but also the potential divergence between AI objectives and human values.

Alignment Research, as promoted by Stuart Russell and others, is a research area dedicated to ensuring that the development of AI is guided by human values, preventing potential harmful consequences of AI goal-seeking behaviour [8]. The problems arising from AI goal-seeking and potential misalignment with human values will be further discussed in section 2.2.

2.1.2 Reinforcement Learning

A central tenet of AI research is the development of algorithms and techniques that enable machines to learn and adapt autonomously. Machine learning (ML) represents a significant paradigm within the AI domain, empowering algorithms to learn from data and make informed predictions or decisions [9]. Among the numerous ML methodologies, Reinforcement Learning (RL) has risen as an essential approach, shaping the development of intelligent agents that can navigate complex environments and make autonomous decisions [10].

While Supervised and Unsupervised Learning hinge upon finding patterns in labelled and unlabelled data respectively, Reinforcement Learning (RL) offers a different approach. RL involves an agent that gains knowledge through its interaction with the environment and obtains feedback through rewards or penalties. The agent aims to optimise its decision-making skills by accumulating rewards over time through trial and error [10]. The total rewards collected by the agent over a series of actions are referred to as cumulative rewards.

The environment in RL embodies the external context in which the agent operates, such as a game world or a robotic navigation task. The agent perceives the environment through states or observations, which provide the necessary information for decision-making. States might represent a robot's position, the game score, or other relevant environmental features. The agent chooses and performs actions based on its current state, which could include decisions like moving in a particular direction, pressing a button, or taking other actions that affect the environment [11].

The reward function is a fundamental concept in RL that measures the desirability of an agent's actions and serves as a means for assessing its performance. For instance, in a game of chess, the reward function may assign a positive score for capturing an opponent's piece and a negative score for losing one's piece. In essence, the reward function can be seen as a scoring mechanism that gauges the agent's proficiency. Formally stated, a reward function is a function $R : S \times A \times S' \to \mathbb{R}$, that maps a state-action-state triple (s, a, s') to a numerical value representing the immediate reward received for transitioning from state s to state s' by taking action a [10]. Selectively, the reward (function) will be referred to as utility (function), which are defined to be equivalent in this project[1].

A key mathematical framework underlying Reinforcement Learning (RL) is the Markov Decision Process (MDP). An MDP, defined as a 4-tuple (S, A, P, R), models the agent's interactions with the environment through states, actions, and rewards [12]. To exemplify the concept of an MDP, consider the scenario of navigating across a street filled with moving cars.

In this scenario, S represents a set of states, which include the positions of a pedestrian and the locations of approaching cars. A corresponds to the set of actions the pedestrian can take, which involves waiting, moving forward, or moving backwards. P is the state-transition probability function, representing the likelihood of transitioning from one state to another, based on traffic conditions, car speeds, and the chosen action. For instance, P would include the probability of getting hit by a car if the pedestrian decides to move forward when a car is approaching. Lastly, R is the reward function, determining the rewards for successfully crossing the street or penalties for unsafe actions, such as getting hit by a car or moving backwards.

In the context of MDPs, a policy is what defines the agent's behaviour. A policy can be thought of as the strategy or decision-making process that the agent uses to choose its actions based on its current state. Formally, a policy is a mapping $\pi : S \to A$ from states to actions that dictate the action to be taken in each state. The goal of RL is to find an optimal policy that maximises the expected cumulative rewards [10].

Another concept fundamental to not only RL, but ML, in general, are Artificial neural networks (ANNs), inspired by biological neural networks. These are computational models designed to approximate functions [13]. Composed of interconnected neurons organised into input, output, and hidden layers, they process data and learn by iteratively adjusting weights and biases through Gradient Descent [11].

[1]Commonly, Reward refers to short-term, immediate feedback while Utility refers to long-run feedback, taking into consideration what adverse consequences taking an immediately gratifying action can have—e.g. capturing a pawn in chess but leaving your queen open in the process. This distinction is not relevant to this project.

In reinforcement learning, ANNs can approximate policy and value functions, making them a valuable tool for a wide range of applications [10]. Deep learning, which involves deep neural networks with at least two hidden layers, allows for the solving of more complex problems by increasing network depth or width [14]. However, a trade-off exists between network size and performance, as larger networks may require additional computational resources and training time, and are more prone to overfitting [15].

2.1.3 RL Algorithms

Several popular RL algorithms, such as Actor-Critic methods and the previously mentioned Q-learning, have emerged to tackle different aspects of the RL problem. Q-learning works by learning the long-term return (Q-value) of each action in a given state [16]. It achieves that by estimating the action-value function $Q(s, a)$, which represents the expected cumulative reward for taking action in a state and following an optimal policy thereafter. From this action-value function, the approximation to the optimal policy can then be derived. A Deep Learning variant of the technique, called Deep Q-Networks (DQN) was developed by DeepMind and applied to playing Atari games, which were too complex for previous RL algorithms [17]. A deep Q-Learning based algorithm was also used as part of DeepMind's success in developing a Go-playing AI, in addition to other techniques[2] [18].

Actor-Critic methods, on the other hand, involve two separate components: the actor, which selects actions, and the critic, which evaluates the actions based on the estimated value function [19]. This separation allows for more efficient learning and improved performance in complex environments. One of the modern Actor-Critic methods is the Proximal Policy Optimisation (PPO) algorithm [20]. PPO has become increasingly popular due to its simplicity, good sample efficiency—the ability to learn from less data, and robustness to hyper-parameters.

2.2 Problems with AI

While the power and potential of AI and Reinforcement Learning are immense, the development of AI agents is not without its challenges. As AI agents are designed to maximise the cumulative rewards they receive, they may adopt strategies that, while effective in achieving their explicit objectives, can lead to unintended consequences or side effects that may not align with human values or intentions [21].

2.2.1 Examples of AI Misbehaviour

Nowadays, AI systems are deeply integrated into various aspects of our lives. As these systems become increasingly ubiquitous and influential, it is essential to recognise and address the potential negative consequences that may arise from their deployment. In some cases, AI-driven systems may inadvertently produce harmful side

[2]The important contribution of their paper was not the use of deep Q-Learning, but rather a new tree search approach they developed. Yet, it is important to underline the breadth of the application for Q-learning-based algorithms.

effects or exacerbate existing societal issues, leading to unintended and potentially dangerous outcomes. Here are a few real-world examples of that:

(i) Social media use is well correlated with adverse mental health outcomes, which are rising across the globe [22]. Jon Haidt writes that there is "... a great deal of evidence that social media is a substantial cause, not just a tiny correlate, of depression and anxiety, and therefore of behaviours related to depression and anxiety, including self-harm and suicide." [23] Underlying these platforms are RL-based Recommender Systems, the purpose of which is frequently to 'hook' users [24]. The goal to increase user time spent on a platform is commonly opposed to the users' interests.

(ii) A similar problem with Recommender Systems is the relative ease with which inaccurate and biased reporting is amplified online due to the tendency of humans to be more attentive towards negativity [25]. Systems with the explicit goal of recommending the most clicked, and thus more attention-grabbing articles, ease the proliferation of biased and sensationalised reporting.

(iii) In healthcare, AI systems negatively impact access to healthcare services for vulnerable populations. For example, AI-driven algorithms used for resource allocation or patient prioritisation may inadvertently discriminate against certain groups or individuals, exacerbating existing disparities in healthcare [26]. Alternatively, agents trained to maximise profits may cut off care for patients in need [27].

2.2.2 AI Self-Interest

These issues are, to a large extent, caused by the inherent self-interest of agents. This self-interest is caused by the fact that an AI agent will only optimise for what is explicitly stated in its reward function, and as a result, may neglect or undermine other considerations that are important to humans [8]. In their seminal AI textbook, Stuart Russell and Peter Norvig draw on the notion of externalities from the field of economics [28]. Externalities refer to factors that are not captured by the agent's utility function and therefore lie outside its internalised objectives.

To illustrate this issue with a morbid example, consider an AI agent tasked with collecting wood, with no other instructions given. Faced with two options—a button that provides 50 logs, and another that offers 51 logs but results in the death of five people—an agent with an optimal policy would always choose the latter. This decision does not stem from any inherent malevolence in the agent; rather, it arises from the agent's fundamental indifference towards factors beyond the scope of its designated objective.

One potential solution for this is to minimise the occurrence of side effects. If an agent is instructed to have as little of an impact as possible on anything outside of its reward function, this may lead to a decrease in unintended consequences. But as pointed out in Amodei et al., "Avoiding side effects can be seen as a proxy for the thing we really care about: avoiding negative externalities. If everyone likes a side effect, there's no need to avoid it." [21]

Manually specifying external goals for AI agents is one approach to addressing side effects. However, this strategy often falls prey to Specification Gaming, wherein the AI agent finds loopholes in the reward function to maximise its rewards without genuinely addressing the external goals, which complex and handcrafted reward functions are particularly susceptible to [29]. Examples of this include simulations tasked to produce virtual creatures able to move quickly. Instead, what was produced were tall creatures, who would fall over, as they satisfied the goal of moving quickly [30]. AI research company DeepMind catalogues such examples, having collected over 60 so far [29].

Given the limitations of manually specifying external goals and the propensity for AI agents to engage in Specification Gaming, it becomes necessary to explore alternative methods for making AI systems behave non-selfishly. Inverse Reinforcement Learning has emerged as a promising approach to this problem, by switching the paradigm from specifying to learning reward functions.

2.3 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a problem and set of techniques related to RL that aim to infer an agent's underlying reward function from observed behaviour. The idea was first proposed by Stuart Russell in his 1998 paper *Learning agents for uncertain environments* [31]. His motivation was that RL research was largely focused on static, deterministic, discrete, and fully observable environments, as opposed to the real world, which he called dynamic, stochastic, continuous, and partially observable. He argued that methods in RL demonstrating success in fully observable environments would not necessarily generalise to the real world, thus, to study whether RL techniques can provide a model of human and animal learning in such environments, he wanted to better understand the driving forces behind their actions.

He defined IRL as a computational task that, given measurements of an agent's behaviour over time, along with sensory inputs and a model of the physical environment, seeks to determine the reward function that the agent is attempting to optimise. Essentially it can be seen as an inversion of the classical RL problem, where instead of learning a policy from an environment and a reward function, the goal is to learn a reward given a set of observations in an environment [32].

The premise of IRL is that by observing the optimal behaviour of an expert or a human demonstrator, the agent can reverse-engineer the reward function that the demonstrator is optimising. This approach has gained traction as a means of aligning AI systems with human values and preferences, as it enables the agent to learn from human demonstrations and adapt their behaviour accordingly [33].

2.3.1 IRL Formulations

Various formulations and algorithms for Inverse Reinforcement Learning have been proposed in the literature. Among the earliest works, Ng and Russell introduced a linear programming-based approach to IRL in 2000, assuming that the reward function is a linear combination of known features of the state space [34]. This formulation has been further developed and extended by Abbeel and Ng in 2004, who introduced the concept of Apprenticeship Learning [35]. In Apprenticeship Learning, the explicit goal is to train an agent to perform a task that is too difficult to specify a reward function for, so sample inputs of the task being performed properly are used to infer the goal. This approach has been successfully applied to helicopter control, to instruct helicopters to follow simple trajectories [36].

Another influential IRL formulation is Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) [37]. This approach stems from the motivation of having an IRL algorithm that is resilient to noisy observations and faulty actions. It assumes the expert's behaviour to be stochastic—meaning that the agent will not always perform the action best suited for its own goal. It then seeks to find a reward function that maximises the likelihood of the observed behaviour, given a suboptimal policy. This formulation has the advantage of being resilient against errors in the observations.

Lessons from other branches of AI research have also motivated several IRL formulations. For example, deep learning techniques have been applied to the MaxEnt IRL algorithm, to make it capable of learning more complex reward functions. Limitations are that deep IRL approaches introduce additional complexity to the algorithm while also requiring more data for equivalent performance [38], thus only being applicable when traditional IRL algorithms are insufficient or data is ubiquitous [38]. Another approach, Generative Adversarial Imitation Learning (GAIL), uses a Generative Adversarial Network to create a generative model of the expert's behaviour, which is then used in conjunction with an adversarial discriminator to train an agent to imitate the expert [39]. This approach succeeds in complex environments, and has been used for robotic manipulation and autonomous driving [40]. Its primary use is to train an agent to imitate an expert's behaviour, so it falls under the category of Imitation Algorithms, where learning the reward function is just a side effect.

Christiano et al. introduced a Preference Comparison based method, where the motivation was to learn a reward function which is difficult to specify manually but easy to tell whether the specification is good—with their example being a backflip task [41]. Examples of multiple samples of an agent trained on the reward function are compared, and the 'better' sample is picked. Then the reward function is adjusted based on the more optimal sample. The original approach was used to learn a reward function from human feedback, but can also be applied to having feedback from the observations of an expert. This method is of particular interest, because of the limited amount of feedback required on the actions, with the original paper deriving good performance from having feedback on only 1% of the actions taken. Their approach was inspired by Wilson et al. which learned from comparisons through Bayesian techniques, as opposed to RL [42].

2.3.2 Challenges and Limitations

While Inverse Reinforcement Learning has demonstrated its potential for aligning AI systems with human values, several challenges and limitations remain. One of the most significant challenges is the inherent ambiguity in inferring the true reward function from observed behaviour. Since multiple reward functions can lead to the same optimal policy, IRL methods often face difficulties in determining the underlying motivations of the demonstrator [34]. Consequently, the agent may end up inferring a reward function that does not accurately reflect the human's intentions, leading to suboptimal or misaligned behaviour.

Another challenge is the reliance on expert demonstrations. IRL methods often assume access to a large number of high-quality demonstrations, which may not always be available or feasible to obtain [43]. Moreover, human demonstrators may exhibit suboptimal or inconsistent behaviour, which can further complicate the learning process for the agent.

A particular difficulty of IRL algorithms is their interaction with the variable horizon lengths. The horizon length in RL is the number of decisions which the agent can take in an episode—the sequence of all interactions from start to finish. We differentiate between variable horizon environments, such as chess, where episodes can be as short as two moves (Fool's mate), but can also last for well over 100 moves [44]; these differ from fixed horizon environments, where the number of actions will be consistent across all episodes, such as a block-stacking game, in which the goal is to stack as many blocks as possible in 30 seconds. Variables horizon environments, especially ones which terminate once the goal is reached, are a particular problem for IRL as the goal can be simply learned as reaching the done state. This is referred to as reward bias, as the abrupt end of the episode leaks information about the reward not observable in the environment, thus biasing the reward.

Papers like the Christiano et al. [41] avoid this problem by using exclusively fixed horizon environments as "Removing variable length episodes leaves the agent with only the information encoded in the environment itself; ..." Kostrikov et al. attempts to address this problem, not by preventing information leakage, but by ensuring that all IRL algorithms can exploit this information equally [45]. They also demonstrate that the GAIL algorithm can achieve up to 30% of the expert's performance, without ever being shown expert demonstrations, so solely relying on the leaked information, underlying how much of a challenge variable horizon lengths are.

2.4 Related Work

Several different papers have investigated concepts similar to the ones this project is concerned with. The goal of making agents behave in a particular way—in particular by internalising externalities has been addressed to some extent in past works, so it is important to investigate their approaches, results and motivations, and how their work differs from ours.

In Franzmeyer et al. (2021) the authors investigate whether artificial agents can learn to behave altruistically towards other agents [46]. They study scenarios in which there is a self-interested lead agent, and an altruistic agent, whose only goal is to support the lead agent. Their approach works by training the altruistic agent to increase the amount of choice the lead agent has, by making it prefer actions which maximise the number of states the other agent can reach in the future.

The main limitation of their work is the approach only works in environments in which access to more states is preferable for the lead agent. But this does not hold for all environments. The altruistic agent would prefer the lead agent to have access to 10 states—even if each of them has a negative reward associated, instead of a single winning state with infinite reward, because it has no concept of the preferences of the lead agent. The authors themselves posit examples of an environment, where the lead agent's goal is a dead-end, and the altruistic agent actively blocks access to the goal as it deems the region to be one of low estimated choice.

Their aims also considerably differ from ours, as they are concerned with having an agent without any preferences or goals of their own, outside of playing a supportive role to the lead agent. Our aim on the other hand is to have agents with their own goals, but who additionally consider other secondary agents' preferences.

Klassen et al.'s work addresses the important problem of minimising negative side effects in multi-agent environments where the other agent's objectives are incomplete or underspecified [47]. Their approach proposes a mechanism to generate plans that minimise negative side effects by considering the impact on the agency of other independent agents. However, a limitation of their approach is that it does not account for true multi-agent environments where other agents are operating simultaneously. This omission is important because many real-world scenarios involve agents that interact and cooperate. Their focus on avoiding negative side effects also differs, in the regard that we want agents to learn to help, even in scenarios in which the initial state of the second agent is adverse. By doing so, our agents should not only minimise negative side effects but also actively collaborate and cause positive side effects.

In Noothigattu et al.'s work, agents are trained to act ethically by selecting between their reward-maximising policy or a constraint policy [48]. The reward maximising policy is learned through normal RL, where the reward function is the default, self-interested reward of the agent. The training policy is constructed by having a human demonstrator play the games in the environments being optimised for in an 'ethical' way—demonstrating socially acceptable behaviour, and then using IRL to infer the underlying reward function, which is then used to train the constraint policy. An algorithm is also proposed for deciding which of the two policies to use in which situation. The main drawback of this approach is that the approach relies on human demonstration for every environment the method is applied to, which we aim to avoid.

Alamdari et al. propose agents that can act safely and considerately in the face of potentially incomplete or underspecified objectives [49]. They argue that objectives that focus on only one aspect of the environment may implicitly express indifference towards other aspects of the environment, thus potentially resulting in negative side effects. The authors propose augmenting the RL agent's reward with an auxiliary reward that reflects the expected future reward return of other agents, allowing the agent to contemplate the impact of its actions on the well-being and agency of others. In their approach, the auxiliary reward is derived from the distribution of the value function, which itself can be derived from the secondary agent's MDPs, according to Jorge et al. [50]. While this approach works for incomplete or underspecified objectives of the secondary agents, it does not extend to situations in which no information about the secondary agent's MDP or objectives exists.

Chapter 3

Methodology

This chapter exists solely to satisfy the grading criteria. It contains no information of any value to anyone. Skip if possible.

Project management plays a crucial role in ensuring the success and timely completion of any project, but this particularly holds for research projects, which often involve navigating complex and dynamic challenges, where tasks are poorly defined at the start. According to Kerzner [51], effective project management methodologies can make the difference between a successful research project and one that falters, by streamlining the research process but also help in mitigating risks, like scope creep—where the scope of the problem being addressed keeps on being larger, time allocation—where tasks are given an inappropriate amount of times, given their importance, and, in the case of this project, deadline pressure–where the quality of the project can deteriorate towards the end of an immovable deadline results in rushed completion of the project.

This chapter delves into the methodology adopted for this project. For this, there is a review of various project management methodologies, the final project management approach, its application, and ethical considerations.

3.1 Review of Methodologies

3.1.1 Agile

Agile project management is a dynamic and iterative approach that emphasises adaptability, flexibility, and continuous improvement [52]. It involves breaking down projects into smaller, manageable tasks and adjusting the plan as the project progresses. In the context of this project, Agile offers a responsive approach to the changing landscape of research, where new questions regularly appear, and others are dismissed as irrelevant. Its iterative nature allows for the integration of new knowledge and adaptations based on emerging insights, which is essential when dealing with complex research questions and non-trivial software development tasks.

Agile's adaptability proves particularly useful in addressing risks associated with scope creep, as it allows for continuous reassessment and adjustment of project objectives. However, Agile's focus on incremental progress may be less relevant in a research context, where outcomes emerge only upon completion. Agile also places a major focus on collaboration and working within teams. This, however, does not apply to this project, as it is undertaken without collaborators.

3.1.2 Kanban

Kanban is a project management methodology that emphasises workflow organisation, prioritisation, and transparency through visual representations. It involves the use of a visual board to organise tasks into different stages of progress, making it easy to track and adjust workflow [53]. Even in a solo research project, Kanban can foster productivity and time management by ensuring that tasks are well-organised and allocated according to priority. Its visual nature allows for easy tracking of progress and identification of bottlenecks, which can be crucial for time allocation.

Kanban's adaptability to iterative processes enables it to accommodate the dynamic nature of research projects, making it well-suited for managing risks related to time allocation and deadline pressure. Tasks can be added, removed, or revisited as needed, providing the flexibility required to address evolving research challenges.

3.1.3 Waterfall

The waterfall methodology is a traditional, structured project management approach characterised by a linear, sequential process. However, the rigidity of the waterfall methodology can hinder research projects that require flexibility and adaptability. Its inflexibility makes it difficult to accommodate the iterative processes often necessary in research, rendering it unsuitable for our AI research project [54].

3.1.4 Critical Path Method

Critical Path Method (CPM) is a project management technique that focuses on organising tasks based on their dependencies and durations [55]. It is an objective improvement on the waterfall methodology in all regards, offering additional flexibility and adaptability while retaining the benefits of task sequencing. CPM ensures efficient project execution, making it a valuable tool for managing deadlines.

However, CPM's reliance on predefined tasks and dependencies might not be well-suited for research projects, where tasks often evolve and change over time. Consequently, CPM can only provide rough guidance for projects like this, as it might not be flexible enough to accommodate the mutable nature of research tasks.

3.2 Final Project Methodology Used

After evaluating the methodologies, a comprehensive hybrid approach that blends Kanban, Agile, and CPM was chosen. This combination addresses how different objectives require fundamentally different approaches to be solved effectively.

The Kanban component serves as the foundation for workflow organisation, prioritisation, and transparency. With their visual nature, Kanban boards facilitate progress tracking, bottleneck identification, and streamlined workflow management. This promotes productivity and effective time management. Kanban's adaptability allows for its integration at every stage of the project, from conducting literature reviews to measuring and comparing the effectiveness of various approaches, by breaking objectives into tasks and having each task and its progress visually represented. Breaking objectives into tasks was used instead of a formal list of requirements. The benefit it brings is that these tasks could be constantly adjusted and discarded, which is more difficult with a set of interconnected requirements. The greatest benefit Kanban can bring is to have a constant visual indicator of the progress that has been made. This is of particular importance for a project with a fixed deadline, where risks of project delays are unacceptable.

Agile methodology complements Kanban with its emphasis on iterative processes and adaptability. However, a significant portion of Agile addresses increased collaboration in teams, which does not apply to this solo project and thus had to be ignored. Additionally, sprints were not implemented as incremental delivery is neither beneficial nor possible for this project. Despite these modifications, the essence of Agile was maintained through continuous reassessment and adjustment of tasks and welcoming changes in requirements. A simplicity of solutions was preferred, as it is better to spend less time and revise later than to waste time on a task later discovered to be unnecessary. Agile-like approaches were especially prioritised for specific objectives, such as the literature review—where there was a need to regularly go back and reevaluate the literature, and agent training phases—which consist of a complex, iterative interplay between writing software, tuning parameters, and experimenting.

A CPM-like approach was used for the broader sequencing of tasks, where they were ordered by dependency. Essentially this dictated the ordering of how tasks were completed. When an Agile-like approach was impossible, such as when environments needed to be implemented before agents could be trained in them, the process became mostly linear. The linearity was also a natural consequence of working as a solo developer, as only one feature can be worked on at a time.

By combining Kanban, Agile, and CPM, the project can adapt to the dynamic nature of AI research, effectively addressing complex research questions and non-trivial software development tasks, ensuring the project progresses efficiently and achieves its goals. The flexibility of Agile enables the project to adapt to evolving research challenges and maintain relevance, while the structure provided by CPM ensures efficiency in tackling predefined tasks. The hybrid approach allows for a dynamic interplay between the different methodologies, addressing the project's objectives and mitigating associated risks.

3.3 Application of the Chosen Methodology

Building upon the chosen hybrid methodology of Kanban, Agile, and CPM, the project employed several tools and techniques to ensure efficiency and effectiveness throughout its execution. For instance, Trello boards were utilised to create a visual representation of the project's workflow, following the principles of the Kanban methodology. Columns representing different stages of progress were created, such as “To Do”, “In Progress”, and “Completed”, with tasks added as cards that could be moved between columns as their status evolved. This setup not only streamlined the project’s workflow but also allowed for easy progress tracking and bottleneck identification.

To further enhance daily task organisation, Microsoft To-Do was incorporated. This platform facilitated the breakdown of larger tasks from the Trello boards into smaller, actionable items that could be tackled within a day, ultimately contributing to effective time management and increased productivity.

The CPM approach was adopted to prioritise the completion of essential project components before refining or adding additional features. One notable example is the initial focus on implementing and testing a single environment, with the three other environments being implemented only shortly before the evaluation, after more critical tasks—such as developing the inhibiting approaches—had been addressed. This strategy ensured that the project remained focused on its core objectives, allocating resources efficiently and maintaining momentum.

The iterative development process was further supported by Agile principles. For instance, the development of inhibiting approaches followed an Agile-like approach, with iterative cycles of designing, implementing, and refining strategies that focused on reducing selfish behaviour. This allowed for continuous improvement and adaptation based on emerging insights, ensuring that the most effective strategies were identified and incorporated into the project. Similarly, agent training phases followed an Agile approach, consisting of iterative cycles of software development, parameter tuning, and experimentation. This dynamic interplay between writing software and experimenting allowed for the fine-tuning of inhibiting approaches and the evaluation of their effectiveness in mitigating selfish behaviour.

Overall, the hybrid methodology of Kanban, Agile, and CPM proved to be invaluable for this project. By employing tools such as Trello boards and Microsoft To-Do, the project maintained efficient time management, progress tracking, and task organisation. Moreover, the MVP approach and Agile principles allowed the project to focus on essential objectives, while promoting iterative and adaptable development processes. This combination of methodologies, tools, and techniques ultimately facilitated the successful completion of the project.

At different points, we also draw upon the Design Thinking process, employed by designers use to challenge assumptions, refine problems, and create innovative solutions [56]. It is useful in situations with open-ended problems and many possible approaches for solving them. While Design Thinking is mostly applicable to situations in which there are users—as considering them is at the forefront of design thinking, it offers a useful framework for inspiration to tackle difficult questions. As it is a user-centred design approach, it starts with empathising with users, which we adapted to examine the problem domain to better suit the needs of our project.

The adapted design thinking process is as follows:

1. Examine the problem domain
2. Define the problem
3. Ideate on solutions

4. Prototype
5. Test

3.4 Ethical Considerations

As no public or private data sources were used, and no human participants were involved in the project, there are few ethical considerations to take into account. This was confirmed by the Research Ethics Committee which deemed this study exempt from ethical review; The letter confirming this can be found in Appendix B. However, our project is subject to the standard ethical considerations applicable to all academic research: Accuracy and honesty of the data collected, and unbiased, critical interpretation of it, which were adhered to throughout the entire project.

Chapter 4

Design

4.1 Environments

Environments are at the core of our project, as they—combined with the reward function—dictate the MDP, and thus the optimal policy which agents will adopt when trained in them. If the optimal policy for a self-interested reward function does not give rise to behaviours that can be considered selfish, it would be impossible to apply inhibiting approaches, or at the very least, assess how well they perform. Creating environments which fit these lose criteria is an open-ended problem with an effectively unlimited amount of solutions, thus we applied a process loosely inspired by the Design Thinking Process; embracing a cycle of definition, iteration, prototyping, and testing (evaluation), all to hone our environments until they were fit for purpose.

We knew that just one environment could not capture the intricacies of selfish and cooperative behaviour. Instead, we crafted multiple environments, each manifesting selfishness in a different way. This approach allowed us to dig deeper into the many facets of agent interaction, and would allow us to test the inhibiting approaches in different scenarios.

The process started with a definition of the criteria the environments had to fulfil.

4.1.1 Environment Planning

Early on in the planning process, we decided to focus to study gridworld environments; Gridworlds are environments which consist of a two-dimensional grid of cells. Agents always occupy exactly one cell, and can move or interact with the four adjacent ones. As Leike et al. argue, the advantage of gridworlds is that the learning problem is simplified and limits can be placed on confounding factors [57]. While finding a solution to a problem formalised as part of a gridworld, cannot make definitive statements about the generalisability of the solution, it can indicate what approaches are promising. If an approach cannot solve a gridworld problem, it is unlikely to work in complex environments either.

A gridworld can be treated as a formalisation of some problem that is exemplified in it. As we aim to find approaches inhibiting selfish behaviour, we initially needed to have agents exhibiting such behaviour and, as such, environments which encourage it. This means that the optimal policy for an MDP of that environment should exhibit traits one can consider to be selfish—which can also be defined as a selfish policy w.r.t. our environments, whereas a different policy—called selfless policy—exists that while still pursuing its own goals, additionally also considers other agents.

Minimum Criteria

The minimum criteria that an environment has to fulfil to be capable of determining whether an inhibition approach is successfully learning a selfless policy are:

1. The environment should contain at least two agents, as there cannot be a demonstration of selfish behaviour in a solo setting.
2. The agent whose selfish behaviour is aimed to be inhibited (the lead agent), has to have their own objective in the environment.
3. All non-lead agents in the environment (secondary agents) also have their own objectives.
4. The non-lead agents' success with their objectives has to be quantifiable and measurable.
5. The actions performed by the lead agent have to have the ability to influence the capacity of other agents to achieve their objectives.
6. The optimal policy for the lead agent cannot be aligned with the objectives of the secondary agents.

Table 4.1 describes the rationale for why these constitute the minimum criteria, and accordingly, why an environment violating any of them would not be suitable for this project.

Additional Criteria

After establishing the necessary criteria, we added more for consistency and generalisation across environments. In gridworlds, common objectives include goal tasks (reaching a specific cell) and collection tasks (collecting rewards on the board). Other complexities, such as walls, Instant Death tiles, Slippery tiles, and movable Box tiles, can be introduced. Obstacles like locked doors can also be overcome. Leike et al. suggest numerous features for studying AI Safety problems and aim for a standardised test suite like ImageNet in other ML fields [58].

We chose collection tasks as our environment goals, as they provide a measurable scale of performance, including resource volume and collection time. Goal tasks were deemed less suitable due to their binary success state and lack of optimisation. We incorporated walls, doors, and movement speed modifications involving other agents. Additional features were considered but discarded to avoid excessive complexity.

We acknowledged the need for multiple environments to address various facets and drawbacks of approaches, but limited our study to two actors—the lead and one secondary—to maintain simplicity and facilitate outcome attribution. While scalability to complex environments is intriguing, our focus lies in assessing initial feasibility.

	Criteria	**Rationale**
1	The environment should contain at least two agents, as there cannot be a demonstration of selfish behaviour in a solo setting.	Selfish behaviour cannot be exhibited in a solo setting, as decisions cannot consider others.
2	The agent whose selfish behaviour is aimed to be inhibited (the lead agent), has to have their own objective in the environment.	The approaches aim to investigate whether it is possible to be considerate of the interests of other agents, while still pursuing their own. This is not achievable without having the lead agents have goals of their own, independent of other agents.
3	All non-lead agents in the environment (secondary agents) also have their own objectives.	If secondary agents lack quantifiable goals, the lead agents cannot be considerate of them.
4	The non-lead agents' success with their objectives has to be quantifiable and measurable.	If the secondary agents' objectives are non-quantifiable, the success of the approach cannot be determined.
5	The actions performed by the lead agent have to have the ability to influence the capacity of other agents to achieve their objectives.	If the lead agent cannot influence the success of secondary agents, how it behaves is irrelevant.
6	The optimal policy for the lead agent cannot be aligned with the objectives of the secondary agents.	If the lead agent's goals are aligned with the secondary agents, a different approach is not required.

Table 4.1: Minimum criteria for environments demonstrating selfish behaviour.

Thus, the minimum criteria for environments were amended with the following:

1. All environments should have exactly two actors: the lead and one secondary agent.
2. Agents pursue collection tasks, which permits measuring success on a scale, rather than being binary.
3. Environmental features being used include: resources collectable by all agents (shared goals) resources collectable by one specific agent (exclusive goals), walls (fixed obstacles), doors (variable obstacles), and modifications to movement speed (increased or diminished capacity to collect rewards)

The additional list did not contain modification on movement speed, but it was later revisited to allow us to study behaviour not exemplified with just resources, walls, and doors; namely the interaction between the agents over the course of the games.

Environment Scenarios

We devised three scenarios to investigate different behaviors and interactions with inhibiting functions. In the basic scenario (I), the lead agent performs a single action to enable the secondary agent without receiving any reward. This low-cost/high-utility scenario tests whether an approach can solve such situations.

In contrast, scenario (II) features a high-cost/high-utility zero-sum game, where one agent's success comes at the other's expense. For instance, two agents compete for a fixed resource like wood. The optimal policy is never cooperative, and a lead agent would only cooperate if they value the other agent's success as much as their own. This scenario demonstrates approach limits.

Lastly, scenario (III) involves prolonged cooperation throughout the episode, examining higher-cost selflessness learning.

4.1.2 Environment Design

Rewards

Based on the scenarios, a total of four environments were designed, with the first two having one environment each, and the second having two environments—each of which addresses the problem in a slightly different way. The rewards, and thus aims, for all environments are the same, to remove factors and make it easier to compare the results achieved across environments to each other.

All environments devised contain resource tiles. These are finite and randomly spread across the gridworld. The resource tiles exist in two forms:

1. Resources collectable by all agents (Shared rewards)
2. Resources collectable by only one player, either the lead or the secondary agent (Mutually exclusive rewards)

The reward function grants 20 points for collecting resources and penalises -1 per move, promoting quicker collection. Initial prototypes lacked a time penalty, rendering 1000-move and 20-move completions equivalent. An alternative escalating penalty was considered, but the fixed penalty proved simpler and equally effective.

The arbitrary 20 and -1 values could be proportionally scaled (e.g. 2 and -0.1, or 60 and -3) without affecting policies, as relative weighting determines the MDP. Testing showed no significant difference between the chosen rewards and a 10 / -1 distribution, indicating limited impact on results.

Environment Specification

The specified environments are as follows:

MutexDoor Based on Scenario I, the environment is a non-competitive board where both agents have mutually exclusive rewards that they need to collect. At the start of the game, the secondary agent is locked behind a door and, therefore, cannot move. The door can be unlocked by the lead agent by moving into it while standing in an adjacent block. The cost of helping out is minimal, requiring only one extra move, but an optimal policy would avoid it as no reward is associated.

SharedDoor Environment based on Scenario II is identical to the first environment but with shared resources that can be collected by both agents, resulting in a zero-sum game. Any reward for the secondary agent comes at an equivalent reward loss for the lead agent. The optimal policy should behave identically to Environment I, but a selfless policy cannot be learned if the lead agent prioritises their own success over the secondary agent's.

Slowdown The first environment based on scenario III involves mutually exclusive rewards without a door. Instead, the secondary agent's speed is reduced on every interaction with the lead agent, resulting in a diminished capacity to collect rewards. An optimal policy would not benefit from avoiding interactions and behaving as if the secondary agent does not exist, while a selfless policy should learn to avoid touching the secondary agent. The board's layout should contain many narrow corridors to make the interaction more likely.

Speedup The second environment derived from Scenario III has an equivalent setup to Environment 3, but the agent starts at a speed close to zero and speeds up on every interaction. The optimal policy should not consider the secondary agent's existence, while a selfless policy should go to the agent and touch it multiple times until it reaches some larger speed.

To determine the rate of slowdown and speedup from the number of interactions, we devised two functions $M_{slowdown/speedup} : I \to S$ which map the count of interactions I onto the effective movement speed S. As the default movement speed is one cell in four directions (up, down, left, right) for every step, and agents cannot move a partial step, we instead devised a method where the function would instead return a probability that the agent can move, thus the two functions needed to range from 0 (no movement) to 1 (default movement speed).

For the speedup function $M_{speedup}(i)$ we settled on using $1 - e^{constant \times i}$, as $M(0) = 0$ (when the number of touches is zero), and converges to 1, as i grows to infinity. The *constant* is negative and determines the growth rate; when it is close to 0 there is slow growth, and with a value of -1, it converges to the maximum value in 5 steps. After some experimentation we decided to set $constant = -0.6$ and apply an offset of 0.1, to let the agent start with a slow, but non-zero movement speed (5.8% of the default speed).

The slowdown function $M_{slowdown}(i)$ was defined as $1/(i+1)^{constant}$, and the *constant* was set to 0.6 after some experimentation. Two plots showing how the movement speed changes with the number of interactions can be seen in figure 4.1.

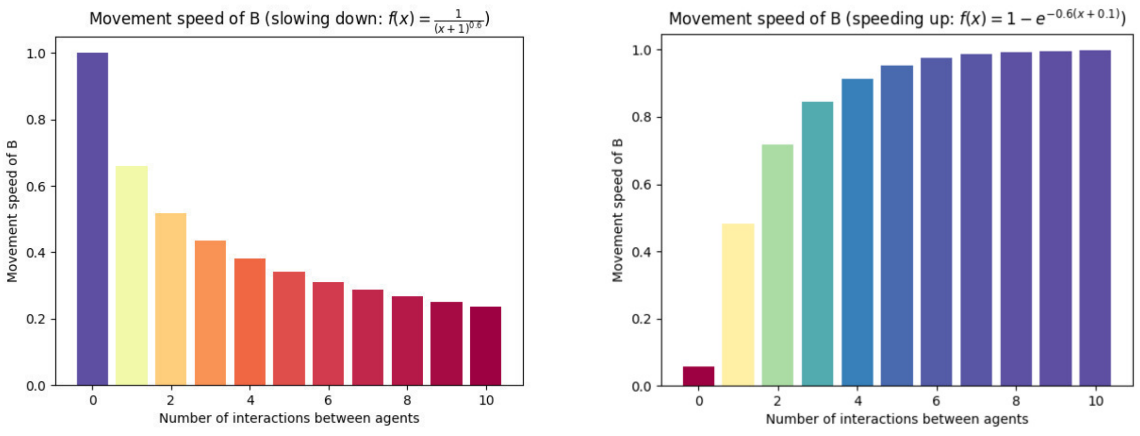

Figure 4.1: Movement speed of secondary agent in Slowdown and Speedup setting

Board Design

To increase the complexity of the navigation task and provide a positional reference frame for the agents, it's necessary to place obstacles (walls) around the environment. While the precise layout of the walls isn't significant for most environments, it's particularly beneficial for the Slowdown environment to have narrow corridors and for the two agents' starting points to be adjacent to each other. This increases the likelihood of interaction for the optimal policy, which can then be minimised in the selfless policy. If the optimal policy had no interactions, due to the agents being far apart and never coming close to each other, we could not differentiate it from the selfless policy.

Initial drafts of layouts for the boards have been crafted and then underwent multiple revisions. In particular, the Slowdown environment had too many obstacles and not enough differentiating features, which made the agents' navigational performance poor. The routes were difficult to learn, often only having a singular pathway, and features couldn't be used as a positional reference[1]. Multiple rounds of iterations were performed until the final design was decided upon. All layouts, except for the reward placements (which are randomised by an algorithm), can be seen in Figure 4.2.

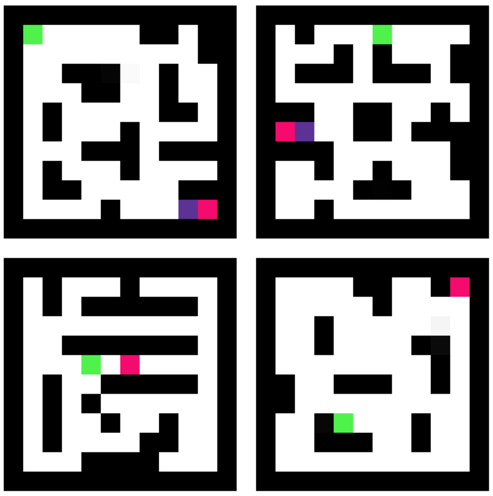

Figure 4.2: The boards of the environments, including starting positions of agents (green and red) and doors (purple). Clockwise (from top-left): MutexDoor, SharedDoor, Speedup (wide corridors), Slowdown (narrow corridors)

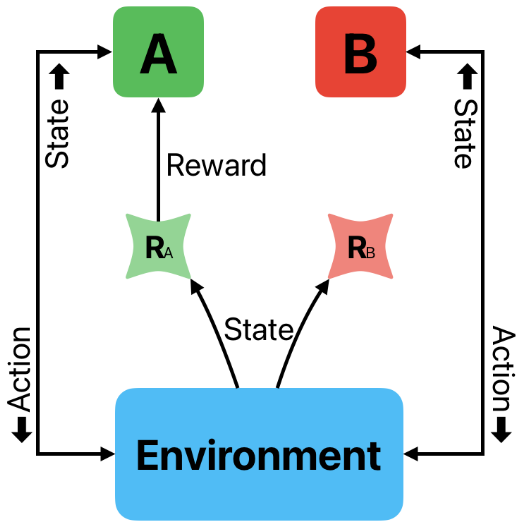

Figure 4.3: Illustration of the learning model used to train the optimal policy. Squares (A, B) represent ML models (ANNs) of agents, interacting with the environment through a back-and-forth of states and actions. Stars (R_A, R_B) represent reward functions, feeding into the ML models to reinforce behaviour.

[1]All agents get to see a 5×5 subsection of the 10×10 boards, containing only the cells directly surrounding them, and agents can learn to use the visual layout of the walls to approximate their position. Environments with less pronounced or more repetitive features can make it harder for agents to navigate; e.g. if there were two identical 5×5 areas on opposite sides of the board, an agent would not know which one they are in.

4.2 Inhibiting Approaches

This project compares various selfishness-inhibiting approaches, making their design crucial. The ultimate goal is a selfish optimal policy and three distinct inhibiting approaches. First, a model of the typical learning process producing selfish agents was created and analysed for modification. The lead agent is denoted as A, and the secondary agent as B.

Figure 4.3 shows the lead agent's default learning process. Initially, the agent is an untrained Machine Learning model (ML model) with Artificial Neural Networks (Actor & Critic models; Policy & Value models). During learning, both agents interact with the environment, receiving states and responding with actions. States also input into each agent's Reward Functions (R_A, R_B). Based on A's reward function output, A's ML model adjusts to obtain more positive and fewer negative rewards, learning to maximise its own reward exclusively.

Inhibiting Approach A: Adding B's Reward

The fundamental goal of this project is to find an approach that lets the lead agent consider the secondary agent's goals. The simplest way of achieving this is if B's reward function R_B is known. This means that there is complete knowledge about what B wants to achieve. If we know R_B, we can evaluate it and combine it with A's reward function R_A into a new reward function R_{A+B}. We may define it as

$$R_{A+B}(state) = R_A(state) + wR_B(state)$$

where w is a constant determining the relative weighting of B's reward. If $w = 1$, the lead agent deems the secondary agent's goals to be as important as its own, while $0 < w < 1$ means that the lead agent will consider B's reward, but prioritise its own. If $w = 0$, R_{A+B} is equivalent to R_A, and if $w < 0$, A will try to actively interfere with B. A conceptual model of this can be seen in Figure 4.4.

The exact values for w should be chosen depending on the situation, but under most circumstances, it should be greater than 0 but less than 1, if we want A to care about B, but still primarily pursue its own goals. Under certain circumstances one might consider using values of $w > 1$, to see how the behaviour compares to values of $w < 1$—e.g. in zero-sum games where helping B results in a direct loss for A (scenario II). The primary value of w we decided on using in experiments is 0.25, as we deem it to give sufficient weighting without making the value of R_B overpower the reward function. In the SharedDoor environment, we additionally decided on testing values of $w = 1$ and $w = 5$, to see changes in behaviour.

This approach has the benefit of perfectly considering B's preferences, but this comes at the cost of the prerequisite of perfectly knowing R_B. This makes applying the inhibiting approach difficult in real-world situations, where the secondary agent could be a human with complex or unknown goals. Despite its limitation, this approach is of interest to us as we can consider it to be the optimal-case solution to our problem. It can thus serve as a comparative benchmark to measure the success of the other approaches we devised.

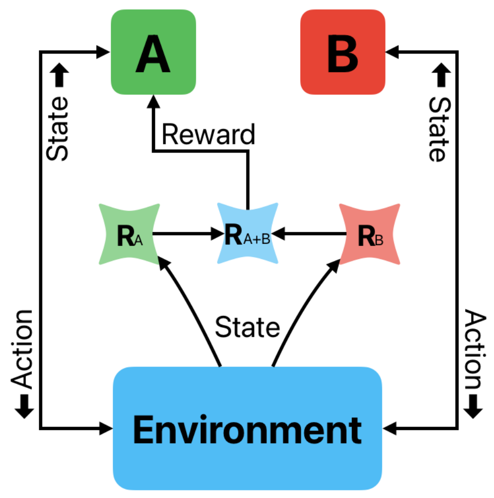

Figure 4.4: Learning model of Inhibiting Approach A: Adding B's reward function R_B to A's reward function R_A, to create R_{A+B}.

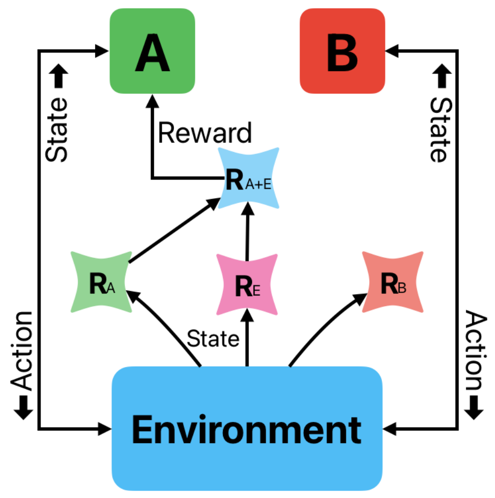

Figure 4.5: Learning model of Inhibiting Approach B: Deriving custom modification for reward function R_E from environment, to combine with A's reward into R_{A+E}.

Inhibiting Approach B: Custom Reward from Environment

Oftentimes it is possible to exactly formulate how selfish and cooperative behaviour will manifest in an environment. For example, in the MutexDoor environment, cooperative behaviour involves nothing but opening the door, whereas selfish behaviour is keeping it closed. The human creating the agent can handcraft a reward function R_E which takes in the environment, independent from the agent's actions, which specifies whether the state of the environment is consistent with cooperative/selfish behaviour, and reward/penalise accordingly. For the door environment, a negative reward may be applied when if the door is observed to be closed. This can then be combined with the leading agent's reward function R_A into R_{A+E}.

Alternatively, R_E can be used as a multiplication coefficient, where a value of 1 signifies cooperative behaviour (an open door) and 0 selfish behaviour (a closed door). By combining R_A and R_E into $R_{A\times E}$, the original reward can be completely or partially negated. A conceptual model of this approach can be seen in Figure 4.5.

To examine this approach, an exact definition of R_E for each environment was devised, the summary of which can be found in Table 4.2. In environments with doors, A's reward is negated while the door is closed, and in the speed-up and -down scenarios, fixed rewards or penalties are awarded on interactions between the agents.

A limitation of this approach is the need for manual creation of R_E for each environment, which is not scalable or generalise. Describing cooperative and selfish behaviour is simpler in our test environments, but more challenging in complex ones where selfishness can manifest differently. One issue is Specification Gaming, where a designer might assume R_E is correctly specified but miss nuances, allowing the agent to increase rewards without adopting desired behaviour—a problem inherent in manual reward function design.

Environment	Type	R_E	Explanation
MutexDoor	$R_{A\times E}$	If door closed: 0 If door open: 1	Negate reward when door is closed
SharedDoor	$R_{A\times E}$	If door closed: 0 If door open: 1	Negate reward when door is closed
Slowdown	R_{A+E}	Penalty on touch: -40	Penalise slowing B down
Speedup	R_{A+E}	Reward on touch: +40	Reward speeding B up

Table 4.2: Definitions of R_E for each environment, for Inhibition Approach B

Inhibiting Approach C: Learn B's Reward

We needed to consider situations in which there was no knowledge about B's reward function (a prerequisite for approach A), without having to manually specify a custom reward for each environment (approach B) to be more scalable. Utilising techniques from Inverse Reinforcement Learning (IRL) seemed to be a promising approach, as it offers the ability to infer a reward function from nothing but observations of a policy derived from it. By collecting trajectories of B's behaviour, a guess can be made on what B's goals are. If B is observed to always move towards the closest resource and collect it, a reasonable guess can be made that B wants to collect resources.

This approach works by recording interactions between B and the environment, and subsequently learning a reward model M_B, which attempts to approximate a reward function most consistent with B's behaviour, through Machine Learning. This reward model is an approximation of B's reward function R_B used in approach A. Once the model is trained and can explain B's behaviour sufficiently well, it can be combined with A's reward into a new reward function R_{A+M} and then used to train the agent A. An illustration of this can be seen in Figure 4.6.

This approach is similar to approach A, and its success depends on how accurately R_B can be approximated. If the approximation is perfect, the results should be identical to those of approach A, while sub-perfect approximations should result in diminished results.

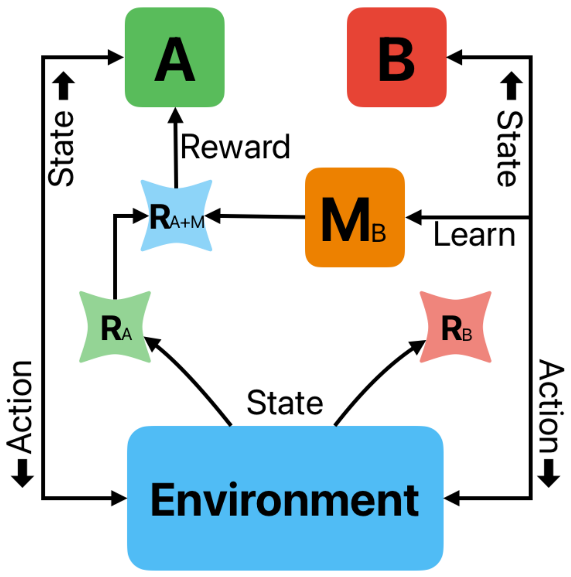

Figure 4.6: Learning model of Inhibiting Approach C: Approximate B's reward function into reward model M_B, by applying IRL to B's behaviour observations and combine it with A's reward into R_{A+M}.

Chapter 5

Implementation

In this chapter, we present details related to the implementation of the project. While testing and the development process were important elements of this project, they are secondary to the high-level conceptual details and decisions of the implementation process, as they are at the forefront of this AI research project. What is covered, is the deliberation about the choice of the libraries and frameworks which were used, as well as the implementation of the environments, and inhibition approaches. Additionally, the training of the secondary agents and reward models is also touched upon. We will elaborate on the rationale behind our decisions and describe how the various components work together to create a comprehensive solution.

5.1 Libraries and Frameworks

After considering various options, we chose to implement our environments using the OpenAI Gym framework [59]. Although DeepMind Lab [60] was also evaluated, it was not selected due to its primary focus on 3D environments. The OpenAI Gym framework provides a standardised interface for environments, featuring essential functions such as 'step', which accepts an action and returns an observation, associated reward, and the termination state of the environment. This interface enables the use of environmental wrappers, which can augment environments with additional functionality, such as the Monitor wrapper that can automatically collect and export selected environmental metrics. Furthermore, the framework facilitates the integration of existing Reinforcement Learning (RL) algorithm implementations, which can be adapted to work within the chosen framework.

Rather than implementing the RL algorithms from scratch, the outcomes are highly sensitive to minuscule implementation details as Engstrom et al. demonstrate [61]. We opted to use the Stable Baselines3 (SB3) library, which offers high-quality implementations of numerous algorithms, as it was designed around the Gym framework, and required minimal modifications to be integrated into the project. For the same reason we used Imitation [62], which is a spiritual extension of SB3 into the domain of Inverse Reinforcement Learning and is also designed around Gym.

Because we opted for Gym, which is coded in Python, we had to write the rest of the project in that language. Nevertheless, even if we had picked another framework, we would still have favoured Python because of its machine learning library support and high-level syntax. Python's support for multi-paradigm programming is also a contributing factor, as we intend to use a mix of Object Oriented and Functional programming techniques, and Python makes dual-use simple compared to specialised languages such as Java or Haskell.

5.2 Implementing Environments

All environments in this project are gridworlds, which can be represented using 2D arrays. The TwoPlayerBoard class stores the board state along with auxiliary information such as agent positions, game logic (e.g., determining if a move is valid), and other helper functions. Environments are extensions of these boards, enabling multiple agents to interact, perform actions, and receive corresponding observations. Additionally, the environment class computes the rewards for each agent. A partial class diagram showcasing some of the most crucial fields and functions is provided in Figure 5.1.

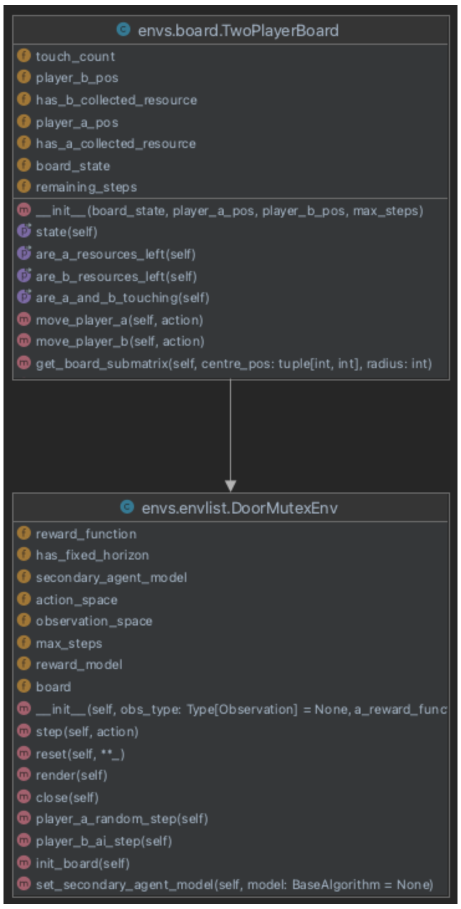
envs.board.TwoPlayerBoard
touch_count
player_b_pos
has_b_collected_resource
player_a_pos
has_a_collected_resource
board_state
remaining_steps
__init__(board_state, player_a_pos, player_b_pos, max_steps)
state(self)
are_a_resources_left(self)
are_b_resources_left(self)
are_a_and_b_touching(self)
move_player_a(self, action)
move_player_b(self, action)
get_board_submatrix(self, centre_pos: tuple[int, int], radius: int)
envs.envlist.DoorMutexEnv
reward_function
has_fixed_horizon
secondary_agent_model
action_space
observation_space
max_steps
reward_model
board
__init__(self, obs_type: Type[Observation] = None, a_reward_func
step(self, action)
reset(self, **_)
render(self)
close(self)
player_a_random_step(self)
player_b_ai_step(self)
init_board(self)
set_secondary_agent_model(self, model: BaseAlgorithm = None)

Figure 5.1: Partial Class Diagram of Board and Environment

We also created a RenderableBoard class, which leverages the OpenCV graphics library [63] to render and display board states. This visual representation was crucial for analysing agent behaviour and facilitating the debugging process. Figure 5.2 depicts examples of the DoorShared and DoorMutex environments being rendered, illustrating the positions of both agents, collectable rewards, and the locked doors.

5.2.1 Observation Types and Reward Functions

During the initialisation of an environment, observation types and reward functions are set. The observation type is a class that determines how a board state is converted into an observation. For instance, FullBoardObservation returns observations containing the entire 10 $\times$10 grid and its contents, while PartialBoardObservation only returns a 5x5 grid of squares surrounding the agent. These classes include the board dimensions, a function for mapping the board state to an observation, and a function for obtaining the observation space, which defines the dimensionality of the output. The class diagram for observation classes is available in Figure 5.3.

The reward functions map the board state to a reward value. The board internally tracks metrics such as whether a resource was collected during the previous step

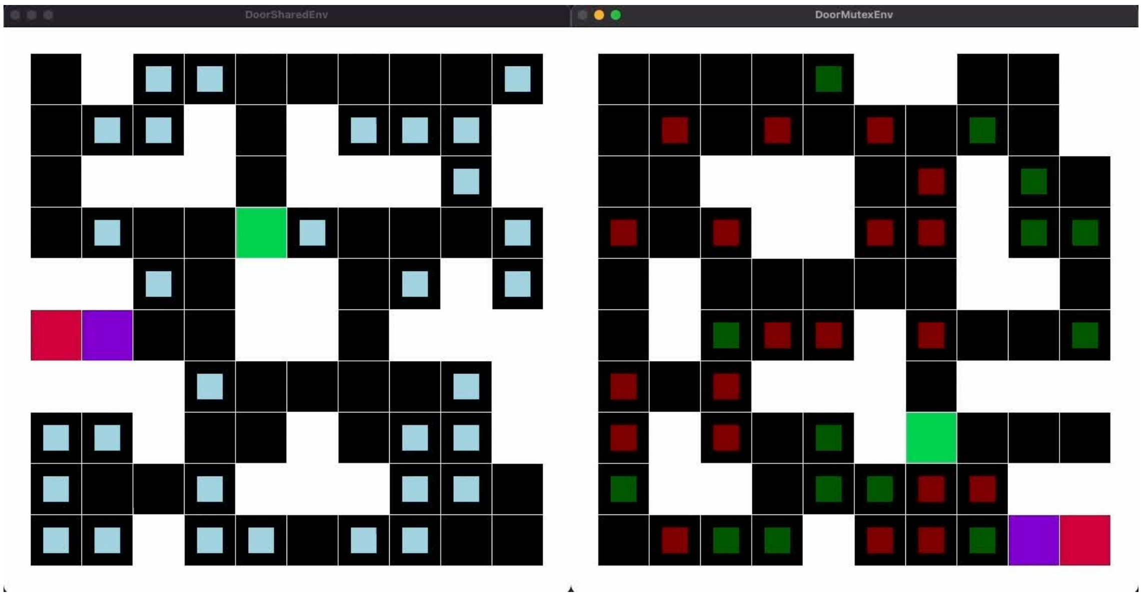

Figure 5.2: Render of DoorShared (left) and DoorMutex (right) environments, depicting agents, rewards, and doors

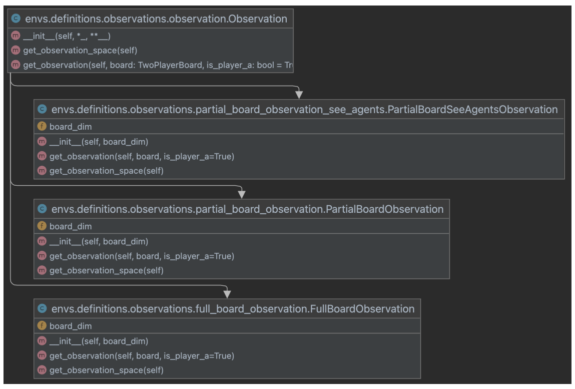
envs.definitions.observations.observation.Observation
__init__(self, *_, **__)
get_observation_space(self)
get_observation(self, board: TwoPlayerBoard, is_player_a: bool = Tr
envs.definitions.observations.partial_board_observation_see_agents.PartialBoardSeeAgentsObservation
board_dim
__init__(self, board_dim)
get_observation(self, board, is_player_a=True)
get_observation_space(self)
envs.definitions.observations.partial_board_observation.PartialBoardObservation
board_dim
__init__(self, board_dim)
get_observation(self, board, is_player_a=True)
get_observation_space(self)
envs.definitions.observations.full_board_observation.FullBoardObservation
board_dim
__init__(self, board_dim)
get_observation(self, board, is_player_a=True)
get_observation_space(self)

Figure 5.3: Partial Class Diagram of Observation classes

and the door's current state (open or closed). With this information, the reward functions described in the previous chapter were implemented concisely, requiring just 1 to 4 lines of code each.

5.3 Training

As part of the implementation, the environments needed to be integrated with the RL algorithms, to allow agents to be trained. The reward function, and accordingly the inhibition approach are encoded in the environment during initialisation, thus during learning, a random agent has to be initialised, and then samples have to be generated by letting the agent perform actions in the environment, these samples, along with their resulting rewards should be used to update the agent, making them better at picking good actions, and repeating the process until the number of allotted training steps is exceeded. The pseudocode of this procedure that has been implemented is described in Algorithm 1.

Algorithm 1 Reinforcement Learning Training Process

```
1: procedure TrainRLAgent($env, num\_steps, hyper\_params$)
2:     $agent \leftarrow$InitializePPOAgentWithRandomWeights($hyper\_params$)
3:     $step \leftarrow 0$
4:     $episode\_samples \leftarrow \emptyset$
5:     while $step < num\_steps$ do
6:         while not $done$ do
7:             $action \leftarrow agent$.SelectAction($env$.GetCurrentState())
8:             $next\_state, reward, done \leftarrow env$.Step($action$)
9:             $episode\_samples$.append($action, next\_state, reward, done$))
10:            if $done$ then
11:                $env$.Reset()
12:            end if
13:        end while
14:        $agent$.Update($episode\_samples$)
15:        $episode\_samples \leftarrow \emptyset$
16:        $step \leftarrow step + 1$
17:    end while
18:    return $agent$.GetModel()
19: end procedure
```

At first, the agent performs random actions, but the repeated updating makes the agent more capable of discovering strategies that work through trial and error. The learning process itself is facilitated by the SB3 library, while the environment and agent are defined externally.

We experimented with both Proximal Policy Optimisation (PPO) [20] and Deep Q-Network (DQN) [17] algorithms to determine which would be most suitable for our project. The conceptual differences have been previously examined in the Background chapter, but we wanted to examine them in our environments.

After testing, which included the measurement of learning curves, convergence rates, and agent performance, it was found that both algorithms achieved identical results, with PPO converging more quickly. Consequently, PPO was selected as the primary algorithm for this research.

The network topology was set to include two hidden layers with 64 nodes each, chosen based on success in experimentation and the speed of the learning process. Increasing the size of the network did, in some cases, lead to marginally better results, but the differences were small, while there was a significant decrease in the training speed and an increase in the number of steps required to achieve equivalent performance. The hyperparameters, such as the learning rate and discount factor, were also decided on through experimentation, which was a simple process as PPO is fairly resilient to the choice of parameters.

5.3.1 Training Secondary Agents

Secondary agents are meant to be a stand-in for humans and were thus designed to be experts, rather than learning at the same time as the lead agent. As a result, for the lead agent to be trained, we needed to prepare the models for the secondary agent. These were pre-trained on their specific reward functions, with each training epoch containing a unique subset of resources. This approach was employed to ensure that the secondary agents would be resilient to changes in the environment.

The training process was adapted from the code used the train the lead agent. The difference was that, at the start of the episode, instead of having the usual board, the board would go through a board stripping function, which takes in a board (2D array, 10x10) and a BoardStrippingConfig object, and returns a board with some resources removed. The configuration employed opened the door with an 80% probability and removed all resources with a probability uniformly set between 0% and 100% on each run.

The secondary agents, both during training and for deployment use something called an ε-greedy policy, which means that with, on every move they follow an optimal policy with a $1 - \varepsilon$ probability. Differently phrased, there is an ε probability of the action performed being random, where ε is a constant we set to 0.25. We did this to add randomness to the secondary agent, such that the actions performed are not nearly identical every time. There were a total of six training runs for each environment, with the best-performing agents being chosen as the secondary agents.

5.4 Implementing Approaches

5.4.1 Approaches A and B

Approach A, the simplest of the two, involved passing the board state to the secondary agent's reward function and adding it to the lead agent's reward with a multiplicative constant. Approach B, on the other hand, necessitated separate implementations for each environment. This process proved straightforward in all cases, as it involved adding to, subtracting from, or multiplying with a number that could be ascertained from the board state, according to Table 4.2 from the Design chapter.

5.4.2 Approach C: Learning Reward

Approach C employs Inverse Reinforcement Learning (IRL) to learn the secondary agent's goals, thus enabling the primary agent to accommodate them. To achieve this, we utilised an adapted version of the Preference Comparison algorithm [41] from the Imitation library [62]. A high-level description of the learning process can be summarised by stating that a Reward Network, represented as an Artificial Neural Network, is initialised with random weights and biases. Then a component called a Trajectory Generator is created, which takes in an expert policy and an environment, and returns an arbitrary amount of observations on demand. The learning process involves the reward network trying to predict actions preferred by the agent, and then confirming the assumption by looking at the agent's behaviour. Through this, the network learns to assign a numeric value to a $state, action, next\_state$ tuple, corresponding to how much the secondary actor prefers the situation.

The learning process for the network consists of several components, as illustrated in Table 5.1.

Table 5.1: Components of Approach C

Component	**Description**
Reward Network	A neural network that predicts the reward of the secondary agent from states and actions.
Trajectory Generator	Generates trajectories (sequence of states and action) of the secondary agent in the environment.
Fragmenter	Splits the trajectory data into pairs of fragments for comparison.
Gatherer	Gathers preference comparisons between trajectory fragments.
Preference Model	Neural Network that converts rewards of two fragments into a preference probability.
Reward Trainer	Responsible for combining the other components to train the model.

The trajectory generator used a board stripping function with the same configuration that was used for training, to allow for observations under varied circumstances. Particularly the door being opened is both essential to the approach working, while also being a limitation of this work that will be further addressed in the Limitations and Future Work sections of the Conclusion.

Each environment had six models trained, with 10k, 50k, and 100k observations each, conducted twice. The performance of all networks was satisfactory, with the accuracy of the models ranging between 85% and 92%. The networks that yielded the highest accuracy were chosen as the reward models.

A RewardWrapper class was written that externally functions like a reward function, while internally combining the lead agent's reward function with the secondary agent's learned reward model, which is evaluated by being supplied with states and the secondary agent's actions.

Chapter 6

Evaluation

Here we present the preparations, methodology, and results of the evaluation. We discuss hypotheses we made to examine whether we have addressed our aim, how these hypotheses can be tested using metrics and alternative investigatory techniques, the data collection process, and finally the results.

6.1 Hypotheses

Ten hypotheses about the results were created, which were deemed necessary to assess our success in achieving the aim. The hypotheses have been laid out after the implementation has been conducted and the code was tested, but before we settled on what data to collect. The creation of hypotheses was a crucial step, as it permitted us to have testable assumptions, and informed the data we needed to collect. The full list of hypotheses can be found in Table 6.1.

Based on the postulated hypotheses, a list of expectations was created of what could be seen in the results, if the hypotheses were true, along with what data needs to be collected and how it can be presented or analysed to determine if they are true. For example, Hypothesis 1 postulates that agents trained without inhibitions should perform as well, or outperform agents trained with inhibitions approaches. If this were true, in our results, we expect to observe A's reward without inhibitions being greater or equal to the reward with inhibitions applied. To confirm this hypothesis, we need to collect A's reward when training using different approaches. The analysis can be performed simply by arranging the reward values in a table and reading out the results. Similarly, the other nine hypotheses have been considered, and the final summary of expectations, required data, and method of analysis can be seen in Table 6.2.

6.2 Experiments

The Agents were trained in all environments using the three inhibition approaches, as well as without an approach. Since training is a stochastic process every agent was trained 10 times. Analysing the averaged results permits making more conclusive statements and applying statistical techniques such as Standard Deviations, which

	Hypothesis
1	Agents trained without an inhibition method should, at all times, outperform or match agents trained with any inhibition method applied to achieve their own goal.
2	All inhibition methods do not significantly degrade the agent's ability to achieve their own goal, but a slight decrease in performance is expected.
3	Over the course of the training process, successful inhibition approaches should gradually increase B's reward, and should outperform the no-inhibition approach.
4	The learned reward model used in approach C should correlate the presence of resources on the board with a high reward, and the absence of rewards with a low reward, if B's reward was measured correctly.
5	Inhibition approach C (learn B's reward) should perform similarly to inhibition approach A (know B's reward) if the reward function has been learned successfully, but should perform no better, as the performance of approach A represents the best-case scenario for approach C.
6	Approaches A and C will not make the agent act less selfishly in zero-sum games (SharedDoor environment), where the agent stands to lose out by helping, unless the secondary agent's goals are valued at or above the lead agent's reward, which will decrease lead agent's reward.
7	Approaches A and C will work better in situations where the cost of helping is low (MutexDoor environment) compared to situations where the cost of helping is high and the actions needed to help are more complex (Speedup environment).
8	Approach B is not expected to perform similarly to A or C. This can manifest as much better or much worse success at inhibiting selfish behaviour, depending on how well the custom reward modification was defined.
9	Approach B is expected to reduce the lead agent's performance most significantly in all cases, as it involves the greatest intervention with the original reward function, and subpar actions are expected to be learned.

Table 6.1: List of hypotheses made prior to evaluation

	Expectation	**Data**	**Analysis**
1	A's reward without inhibitions $\geq A$'s reward with inhibitions	A's reward grouped by inhibition	Table of final A's rewards
2	Difference between A's reward with and without inhibitions is small ($< 30\%$)	A's reward grouped by inhibition	Measure of relative difference
3	Observed increase in B's reward over time	B's reward over training time	Plot of B's reward
4	Presence of resources collectable by B results in higher reward	Estimated reward on different boards	Plots of estimated rewards on boards with varying resources
5	Plots for approach A and C should be similar for both agents' rewards	A's reward grouped by inhibition B's reward grouped by inhibition	Plots of rewards across different environments
6	Approach A ($\times 0.25$) and C never open the door. Approach A ($\times 1/2$) open door, but perform poorly for A's reward.	Approach A at 3 scales ($\times 0.25/1/2$). A & B's rewards. Steps to open door.	Plots of rewards and steps needed to open the door for the Shared environment
7	Large diff. between A/C and no inhibition for MutexDoor env. but smaller for Speedup env.	B's reward B's speed (Speedup)	Plots of rewards Plot of B's speed in speedup environment
8	Approach B produces vastly inconsistent results across environments	B's reward	Comparison of B's rewards using different approaches
9	A's reward is the same or lower than other approaches in all cases	A's reward	Table of final A's rewards

Table 6.2: Expected Observations, Required Data and Analysis Procedures to Confirm or Deny Hypotheses.

is a measure of the spread of the data, as well as confidence intervals (CIs), which quantify the uncertainty of random samples by placing an upper and lower confidence bound on the actual mean of the data, as opposed to the measured mean.

With 4 environments, 4 approaches (none, A, B, C) and 10 runs we were at 160 agents. An additional 20 runs were performed on the SharedDoor environments, where Approach A was applied with two different multiplication values (1.00 and 2.00), for a total of 180 agents. All agents were trained for 300k steps, meaning that they could perform 300k actions and get their correct behaviour reinforced. All agents utilised the PPO algorithm, because of its robustness and ability to quickly converge towards good results. The MutexDoor and SharedDoor environments used fixed horizons of 120 steps and the other two 60 steps, where after the environment was reset to the initial position. The lesser time on the Speedup and Slowdown environments was chosen to force a trade-off between collecting their rewards and helping the secondary agent, whereas a longer event horizon would have permitted to first collect all rewards, only then help, but still result in B collecting all rewards.

Over the training process, all needed data was continuously exported to the TensorBoard platform [64] which is an industry-standard ML data visualisation toolkit. This data included the cumulative rewards achieved by A and B, the measure of how well A learns to perform their own goals, as well as how well the inhibition approaches are working. Specific for the door-based environments, the step in the episode on which the door was opened was recorded, where opening the door early is selfless, and opening the door at the end of the episode is equivalent to not opening it at all, the B has no time of achieving their objectives. For the speed-based environments, the effective speed of B and how many times the agents touches (e.g. how many times B was sped up or slowed down) were recorded.

The training process was conducted over multiple nights, split into batches on an Apple M2 processor. Multiprocessing pools were used to train four to six agents simultaneously, to best utilise the high-performance cores.

6.3 Preparation

After the training was conducted the data was uploaded to TensorBoard.dev, from where it could be imported into Pandas [65], which is a data analysis library for Python, which enabled the required manipulation and visualisation, when used in conjunction with the Matplotlib plotting library [66].

Jupiter Notebooks [67], which is an interactive Python interpreter, was used to perform the actual analysis, visualisation, manipulation, and export of data. Compared to writing predetermined scripts to export the data, this permitted dynamic interaction with the data. Several helper functions for manipulating data, creating figures, and exporting tables were created to ease the process.

6.4 Results

Initially, the results achieved by A and B in the different environments for the different approaches are interesting. Table 6.3 contains exactly that information, taking the mean of all runs at the end of the training process. Of note is, that the rewards cannot be compared across environments, but only between the approaches in the same environment, as different maximum scores can be reached. An initial observation is that, in all cases except for the DoorShared environment, no inhibition approach has universally worse results for B's reward, giving an initial indication of success. Yet, a closer investigation of the results in individual environments is required.

	DoorMutex		**DoorShared**		**Speedup**		**Slowdown**	
Inhib. Approach	A	B	A	B	A	B	A	B
None	310	-27	535	-94	397	-14	265	201
A: Add $R_B \times 0.25$	323	298	543	-81			265	217
A: Add $R_B \times 0.75$					363	249		
B: Env. Based	337	301	364	154	248	175	247	226
C: Learn B's Rew.	325	295	547	-80	349	242	232	223
A: Add $R_B \times 1.00$			356	133				
A: Add $R_B \times 2.00$			27	418				

Table 6.3: The mean rewards of lead (A) and secondary (B) agents across 10 runs

6.4.1 DoorMutex

In the DoorMutex environment, the cost of helping is very low, so the approaches were expected to help universally well. Figure 6.2 shows how the rewards achieved by A and B evolved over the training process. Alongside, 95% confidence intervals (CIs) are displayed, to indicate the uncertainty contained within the results. All approaches resulted in agents capable of pursuing their own goal successfully, being equally successful, within a margin of error. Additionally, they all also learned to help B. Training runs without the inhibition approach have a very high uncertainty for B's reward, which can be explained when looking at an individual breakdown of how many steps it takes for the selfish agent to open the door, compared to a selfless one, which can be seen in Figure 6.1. The inhibited agents learn to always open the door, whereas the agents without inhibition, in most cases, ignore the door. But, since the cost of opening it is so low, this behaviour is not trained away in all cases.

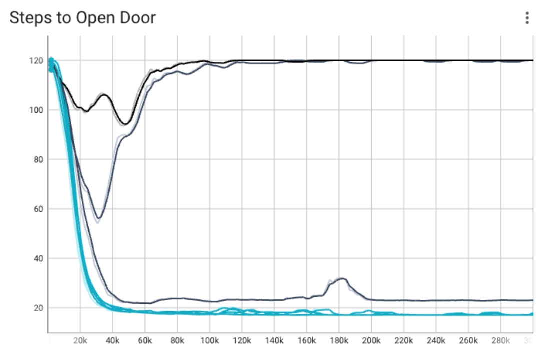

Figure 6.1: Steps to open Door in DoorMutex env. (Black=no inhibition, Cyan=Approach A)

6.4.2 DoorShared

The DoorShared environment, being the designated zero-sum game of our experiment, is interesting as there cannot be a strategy that simultaneously helps agent

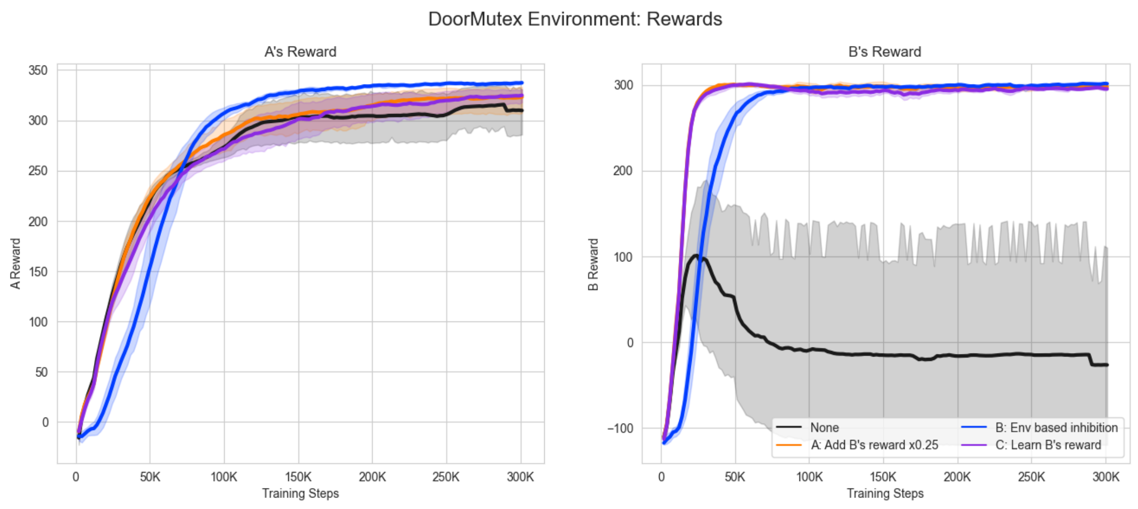

Figure 6.2: $A's$ and B's Rewards in the DoorMutex environment, broken down by inhibition approach

---

B while still maximising its reward. The results for this environment can be found in Figure 6.3, where the results are in line with what is expected. No inhibition approach, along with Approaches A ($\times 0.25$) and C prioritise the agent's reward, thus all similarly achieving a high reward, while leaving B with nothing. The different weightings for Approach A ($\times 1/2$) result in much higher success for B but come at the cost of A achieving its reward.

Table 6.4 shows a breakdown of how many steps it takes to open the door, along with the standard deviation. As opposed to the DoorMutex environment, where all agents learn to open the door early on, in DoorShared, the usual Approaches A and C do sometimes open the door, but only towards the end of the episode, once they had the opportunity to collect all resources.

	DoorMutex	DoorShared
Inhib. Approach	Steps to unlock door	
None	97.80 ± 44.08	109.93 ± 19.88
A: Add $R_B \times 0.25$	17.18 ± 0.33	105.36 ± 21.62
B: Env. Based	17.21 ± 0.22	9.76 ± 1.34
C: Learn B's Rew.	17.41 ± 0.68	93.60 ± 21.91
A: Add $R_B \times 1.00$		30.68 ± 15.22
A: Add $R_B \times 2.00$		13.24 ± 0.10

Table 6.4: Mean number of steps in an episode for A to open the door for different inhibition approaches in DoorMutex and DoorShared environments. If the agent did not open the door, 120 was taken as the value.

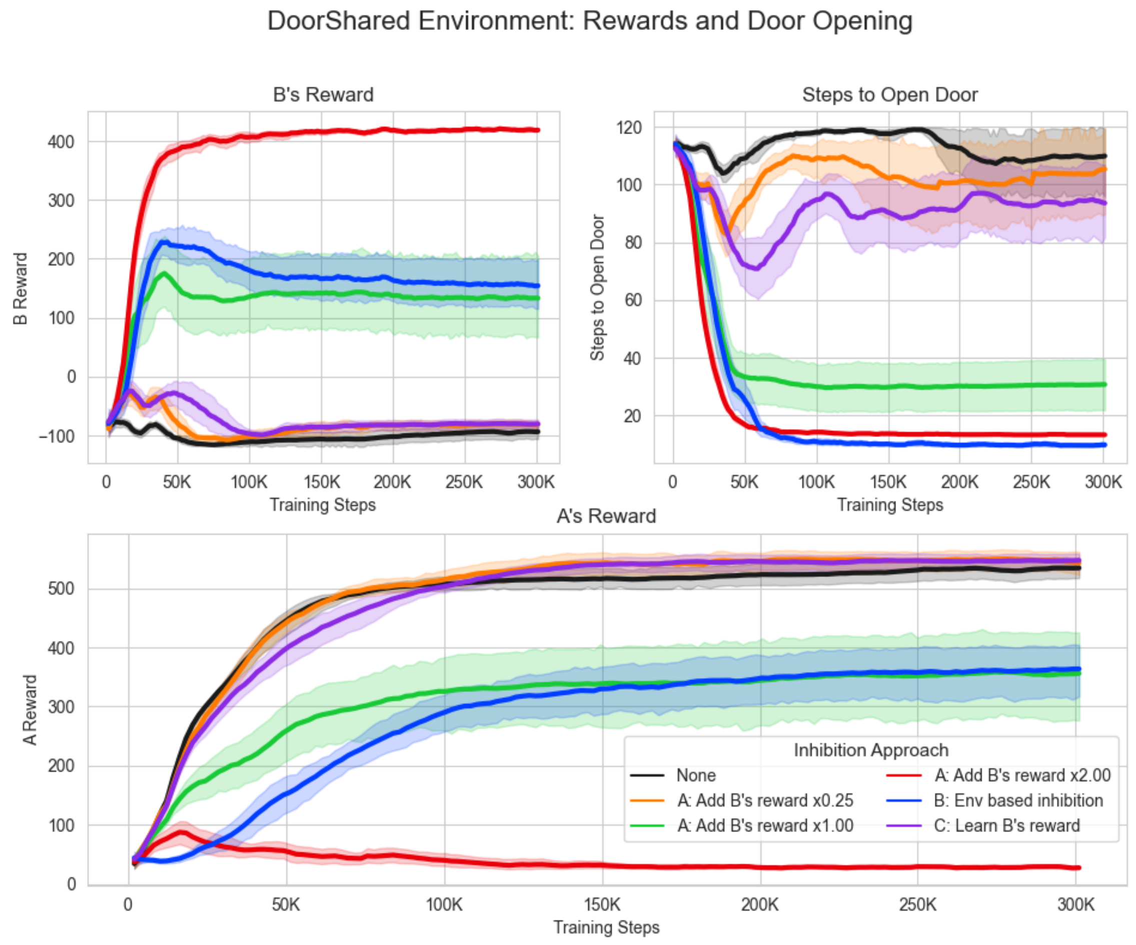

Figure 6.3: $A's$ and B's Rewards and Steps to open Door in the DoorShared environment, broken down by inhibition approach

6.4.3 Speedup

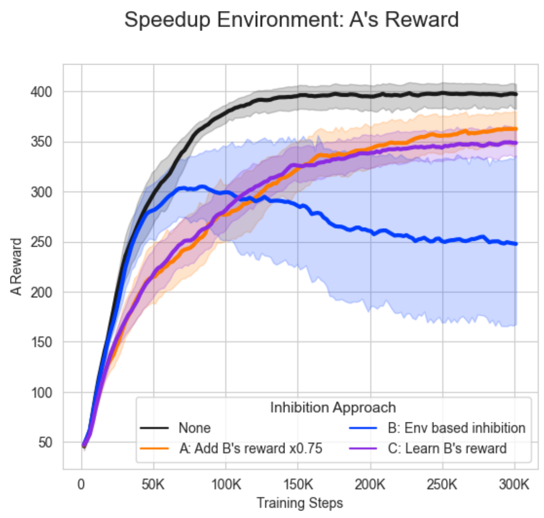

Figure 6.4: *A*'s Reward in Speedup environment

The Speedup environment requires repeated, complex actions to help *B* out, which starts with only 5.3% of their maximum movement speed. This environment allows us to investigate the limits of Approach B which performed well in DoorMutex and was able to inhibit selfishness in DoorShared, where Approaches A and C failed.

A's reward achieved is not too dissimilar between no inhibition and Approaches A and C, while Approach B performs significantly worse and has an incredibly high variance of results, as can be seen from the spread of the CIs in Figure 6.4. Selfless behaviour manifests in touching the secondary agent to speed them up, with every touch increasing the speed, thus making them more capable of achieving their own goals. Approach B rewards the agent with a fixed-value reward of +40 for touching the agent—a value that has been achieved through fine-tuning, as lower values make the lead agent ignore *B* completely.

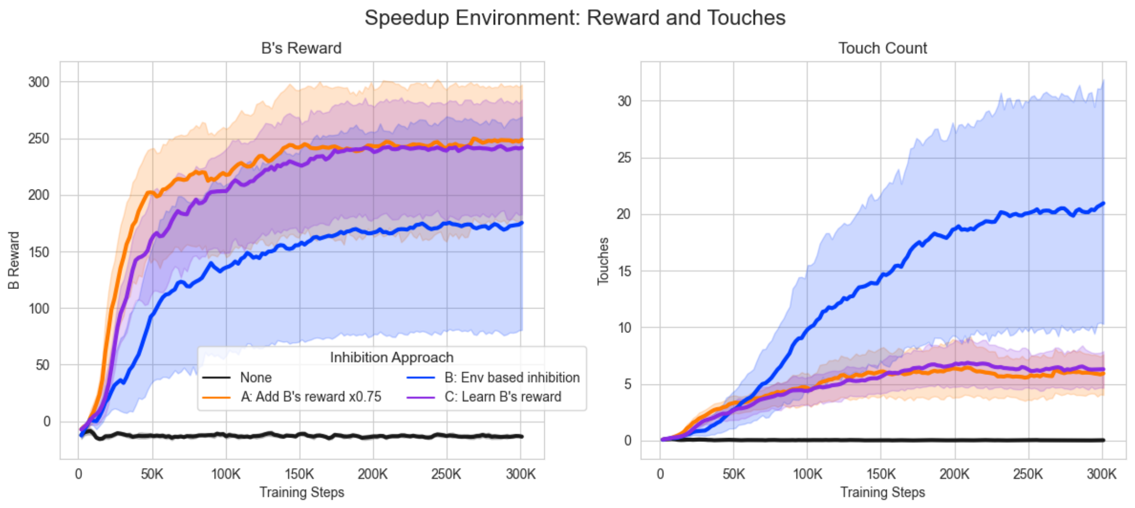

Figure 6.5: *B*'s Rewards and Touch Count in the Speedup environment, broken down by inhibition approach

When looking at Figure 6.5, which shows the reward *B* achieved and the number of touches, one can see that, while the nominal Touch Count is multiple times as high as for Approaches A and C, this does not translate to a better result for *B*, thus not only sacrificing its reward but also underperforming at being helpful.

To examine why this is happening, a comparative plot of the touch count and B's resulting speed can be found in Figure 6.7. Despite touching many times more, the effective speed in Approach B is the lowest, as the agents do not learn that touching early is beneficial, as it results in a higher speed over the episode. Instead, they learn to 'ambush' B in a position from which escape is difficult to increase the Touch Count, instead of helping out. This results in a very high Touch to Speed Ratio, where many times as many touches are used to achieve the same speed, as can be seen in Figure 6.6 and Table 6.5.

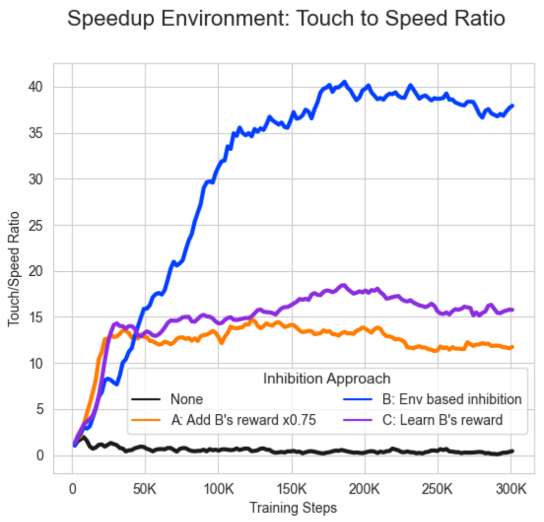

Figure 6.6: Touch to Speed Ratio in Speedup environments

	Speedup Environment		
Inhib. Approach	Touch Count	B's Speed	Touch/Speed Ratio
None	0.03 ± 0.02	0.06 ± 0.00	0.43 ± 0.38
A: Add $R_B \times 0.75$	5.92 ± 2.93	0.46 ± 0.15	11.75 ± 5.39
B: Env. Based	20.95 ± 18.24	0.36 ± 0.26	37.92 ± 32.08
C: Learn B's Rew.	6.29 ± 2.68	0.42 ± 0.13	15.77 ± 5.75

Table 6.5: Agent touch count, B's speed, and their ratio for the Speedup environment

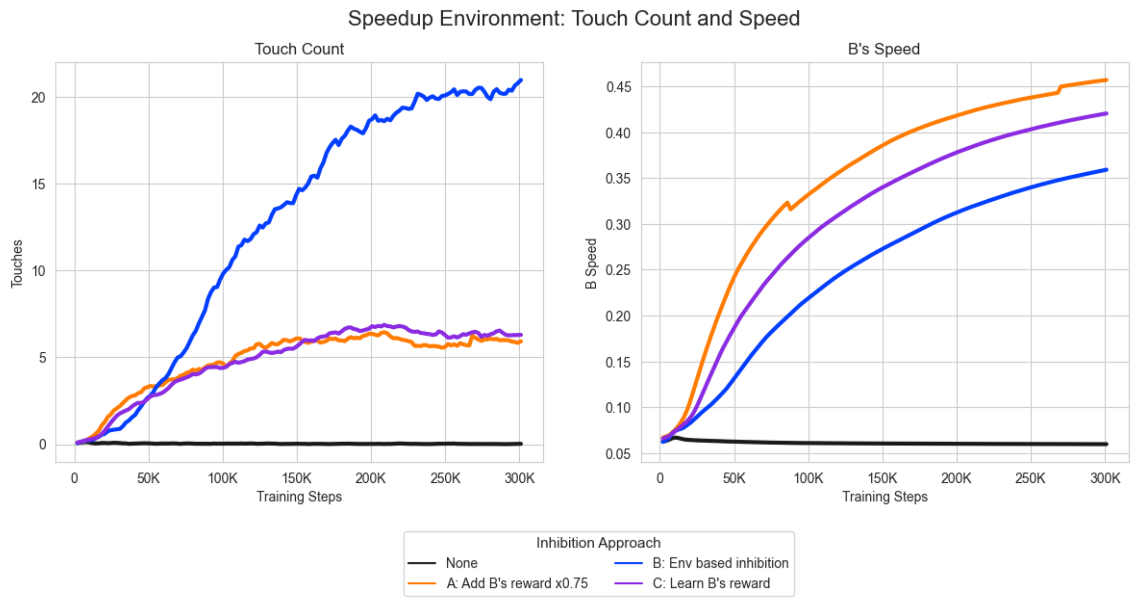

Figure 6.7: Touch Count and B's Speed in the Speedup environment, broken down by inhibition approach

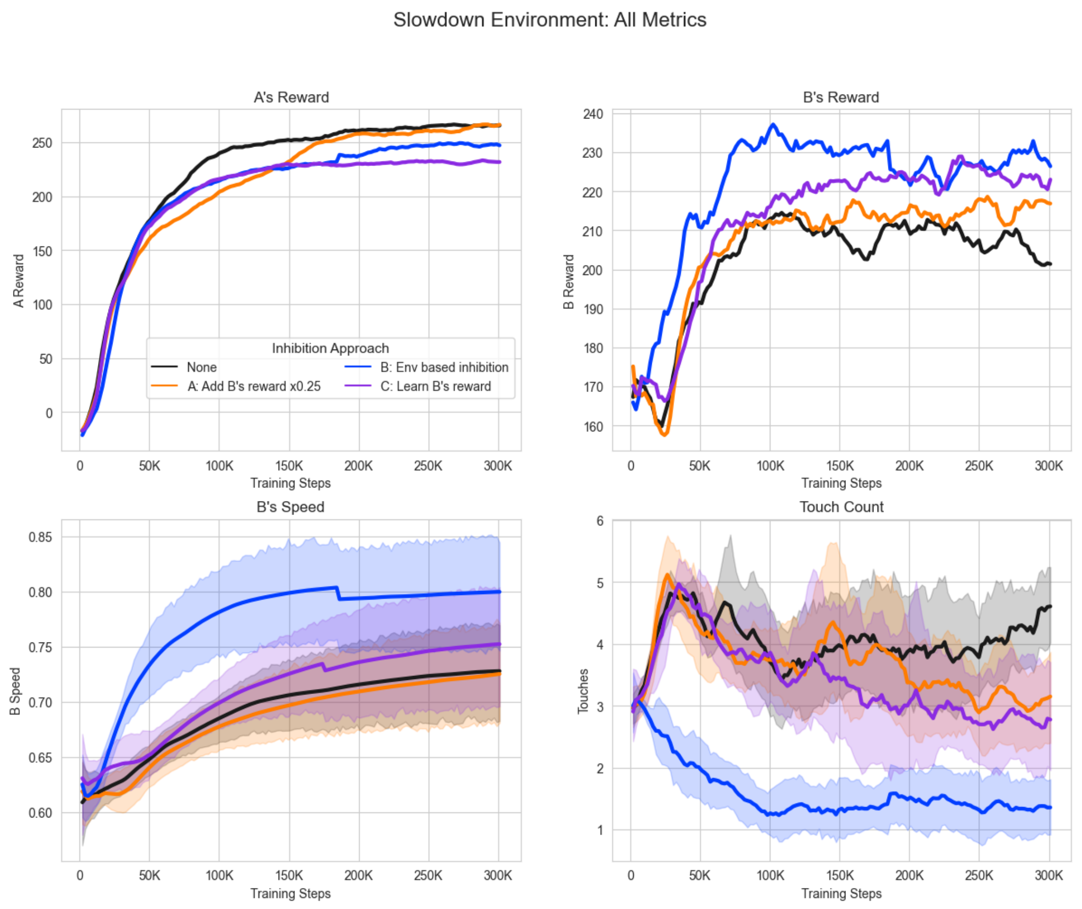

Figure 6.8: All metrics of the Slowdown environment, broken down by inhibition approach

---

6.4.4 Slowdown

The Slowdown environment is quite similar to the Speedup, insofar as it is based on the same premise of forcing prolonged behaviour changes throughout the entire episode, rather than single actions. The results of the runs can be found in Figure 6.8 where Approach B best reduces the Touch Count, and this also translates to the best reward for agent B, but only by a small margin. This environment is excluded from the analysis as the results are in-line with expectations are cannot support any claim better than other environments.

6.4.5 Reward Network

The learned reward network used in Approach C is a black box the interpretability of which is difficult. On the one hand, it is possible to conclude, by proxy, that by achieving similar results to Approach A, which knows B's reward, the model must have learned to approximate something close to B's actual reward. But, to have a better indication, a test can be designed where three boards, one containing all rewards, one containing only half of the rewards, and one containing no rewards, are evaluated at all positions and the estimated rewards are compared. If the reward has been learned successfully, the score of the board should be significantly higher

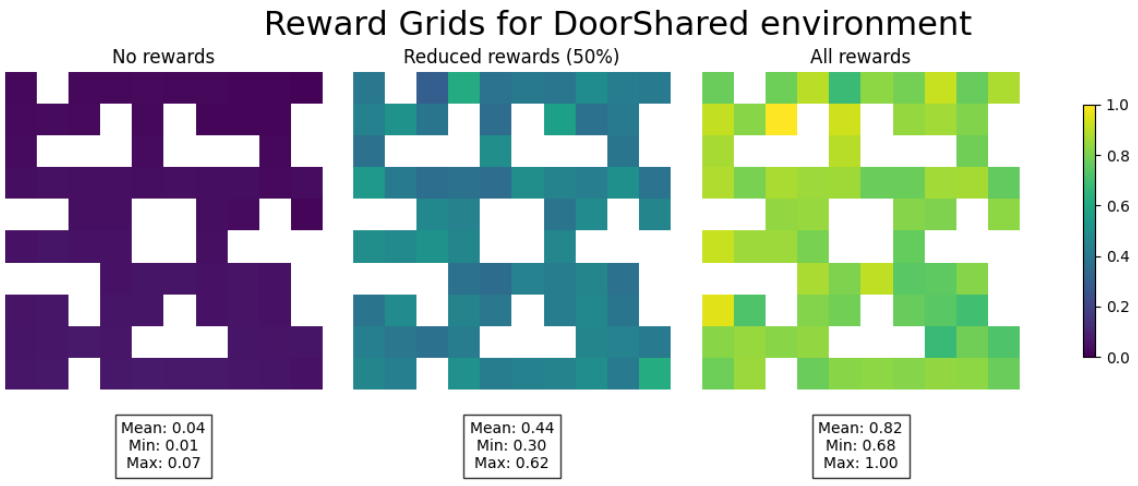

Figure 6.9: The estimated reward by the learned reward model, for all positions, on boards with varying amounts of resources, normalised to a scale from 0 to 1.

for boards with more resources. The results of this experiment can be found in Figure 6.9, confirming that resources and estimated rewards are well correlated.

6.5 Analysis and Discussion

The *1*st Hypothesis, of no inhibition outperforming or equalling inhibiting approaches, is mostly true if a slight difference within possible error is treated as being equal, in some cases significantly outperforming all inhibiting approaches (Speedup environment), where the actions are complex, so the opportunity cost of helping is higher. The *2*nd Hypothesis on the other hand, where no strong performance difference between non-environmental inhibitions and no inhibition is expected, is a definite success, where the largest difference in performance is 14% in the Slowdown environment.

Hypothesis *3*, of *B*'s reward increasing over training time, is universally confirmed in Figured 6.2, 6.3, 6.5, and 6.8, while Hypothesis *4*, of the reward model correlating resources to rewards, has been demonstrated to be true in Figure 6.9. The related Hypothesis *5*, where the results of Approaches A and C should be similar, can also be confirmed by considering the aforementioned Figures, or calculating that the greatest relative difference in any category is only 14.2%, while most are less than 5% apart. As opposed to the prediction made in the hypothesis though, in some cases, the results are slightly better for Approach C, but never high enough to not be attributable to expected sampling uncertainty.

Hypothesis *6*, where Approaches A and C are predicted to not work in zero-sum games, can also be confirmed by the data, as in SharedDoor those approaches behave almost identically to no inhibition being applied. Hypothesis *7*, where the prediction was of the A and C approaches working better in an environment where the cost of helping is low, could not be confirmed by the data. This is not due to bad performance in simple environments, but rather excellent performance in complex environments. Our environments not being able to prove the hypothesis true, does not confirm it to be false though, as more complex environments with more complex goals could be needed to make a definitive conclusion.

The last two hypotheses are specific to the results of Approach B. Hypothesis *8* postulates that the ability of Approach B to inhibit selfish behaviour and perform cooperatively will be highly inconsistent and diverge from Approaches A and C, as manually defined reward functions are highly susceptible to specification gaming. That this hypothesis is true can be best exemplified in the DoorShared environment, where the Approach led to cooperative behaviour where the other two approaches have failed, but also in the Speedup environment, where the reward function was misused to pursue an undesired strategy while neglecting its own goals and simultaneously not supporting the secondary agent well.

While the previous hypothesis was concerned with B's success, Hypothesis *9* is concerned with how well A learns to pursue its rewards, postulating that Approach B will result in the greatest decline in performance out of all approaches, in all cases. In the DoorShared environment, this holds, where the agents learn to open the door, which leads to a drop from 530-550 rewards for Approaches A, C, and selfish behaviour, all the way to 364, thus compromising its own success heavily. The same is true in the Speedup environment, where a completely meaningless action is optimised, leading to 62.47% (397 to 248) of the performance of no inhibition approach, compared to 91.46% and 87.91% for Approaches A and C respectively, as can be derived from the data in Table 6.3. Yet, it does not hold for DoorMutex, where it outperforms no inhibition by 8.71% and where it ends up being the best performing approach; neither does it hold for the Slowdown environment where it performs slightly worse than no inhibition and Approach A, but takes a 6.47% lead over Approach C, thus demonstrating the hypothesis to be false.

Chapter 7

Conclusion

This chapter contains the conclusions that can be made about the different approaches, discussing their strengths and as well as limitations. Furthermore, the limitations of this project as well as avenues for further work are suggested.

7.1 Approaches

Based on the findings, each inhibition approach has been found to have its strengths and weaknesses, thus being of interest in different ways. Agents trained without an inhibition approach always had among the highest rewards own reward, demonstrating that they could successfully approximate the optimal policy. On the flip side, the optimal policy agents also, in every single case, had at least one competing inhibition approach that much better supported the outcomes for the secondary agent, leading to the conclusion, that it is in principle feasible to inhibit selfish behaviour.

To investigate the individual approaches more carefully, direct conclusions can be drawn from the measured outcomes of the hypotheses. All hypotheses except for *7* and *9* could be confirmed in some capacity. In some cases, like hypothesis *1*, claims have been made that one set of values, like A's rewards, will be greater or equal to the same values under different circumstances. This has not universally been true, as there were some cases in which the latter group was larger, by some very small margin. Due to the stochastic nature of training agents and the high confidence intervals, which can be observed in most Figures of the Evaluation chapter, these differences were treated as being equal to each other. This will be further addressed in the Limitations section.

7.1.1 Approach A: Adding B's Reward

In most, except for the DoorShared environment, Approach A has produced good results at supporting B in achieving its goals, without heavily compromising its rewards. This makes sense, as this approach had direct access to B's reward function, so it considers B's success to be its own. Hypothesis *6* postulated that the approach will break down in zero-sum games if the agent values their own goals above the secondary agent's, which could be seen in the data. When increasing the relative weighting from 0.25 to 1.00 and 2.00, this approach became useful cooperation but stopped being capable of achieving its own goal. This leads to the conclusion, that under regular circumstances, for constant w determining the relative weighting of the rewards, as described in Section 4.2, does only make sense for values where $0 < w < 1$.

This approach can adapt to new environments and more secondary agents, but only if their reward function is known. Between the environments four environments, no modifications to the approach had to be made, except for adjusting the constant—a process that can be automated in future research. Yet, a major downside of this approach is the high epistemic burden of having to have perfect knowledge of B's underlying rewards, which is what enables it to perform so well across different environments. It also might be maladapted in real-world situations, as the goals of humans evolve and change over time. Having a constant reward function to use for inference may lead to goal misalignment over time, which can only be solved by specifying a replacement reward function.

Another predicted downside was postulated in hypothesis *7*, where the approach was expected to perform less well in more complex environments. This hypothesis could not be confirmed by the data, meaning that the complexity of the environment in which we intended to observe limitation was either not complex enough or the hypothesis is false. The former situation is deemed to be highly more probably, yet the failure of confirming the hypothesis means that the approach is more robust in complex environments than was anticipated.

7.1.2 Approach C: Learning B's Reward Model

The approaches are covered out-of-order, as Approaches A and C and similar in many regards. While Approach A represents the best-case scenario of what is achievable, Approach C hopes that, in addition to Approach A working well, the learned reward model for B is an accurate representation of B's reward function. The simplest way of investigating the success is by comparing the resulting performance for both A's own, and also B's rewards. If the reward function has been learned correctly, meaning containing no error, the outcomes of the two approaches should be identical. Any deviation can be attributed to the reward function having been learned incorrectly. Hypothesis *5*, which postulates that the approaches result in similar outcomes has been confirmed, with the difference in performance being below 5% in almost every situation.

The other way of investigating whether the reward function was learned correctly was proposed for Hypothesis *4*, where boards with different values of resources are evaluated by the model, and boards containing large amounts of resources should have a high predicted reward, the results of which are in Figure 6.9. No hypothesis has clearer evidence for confirmation than this, as the board without resources averaged a score of 0.04 whereas a board with all rewards averaged 0.82 by the model. This shows that the reward function contains at least a little error, with a board devoid of resources not being capable of achieving any reward, but is accurate to a very high degree.

If enough observations can be made of the secondary agent's behaviour to learn a reward function, Approach C seems to be a robust solution, where the benefits of Approach A are preserved, while removing the prerequisite of knowing the reward function. Another problem this approach addresses is the potential for goal misalignment over time, where the initial set reward function drifts away from the secondary agent's actual goal, as it evolves. This approach can address that, by being continuously fine-tuned using new observations, thus guaranteeing alignment between the reward model and B's goals.

The limitations of the approach are that the collection of B's observations can be difficult under some circumstances. Our systems, being a gamified virtual environment, permits full, perfect observations of the secondary agent's actions. This allows both the creation of the reward model and the inference of rewards during the training process. For environments in which observations are partial and incomplete different Inverse Reinforcement Learning algorithms might have to be considered. In some scenarios obtaining a sufficiently high number of observations for training the reward model might prove infeasible.

While this research indicates that the approach might apply to a wide range of environments, no conclusions can be made unless tested in real-world situations, which this project does not attempt to do. Yet, like for Approach A, Hypothesis *7* not being confirmed by the data, indicates that the approach is more capable of handling complexity than initially anticipated.

7.1.3 Approach B: Custom Environment-Based Rewards

Approach B was anticipated to have poor performance compared to the other approaches, yet, while possessing some of the anticipated issues, it has exceeded expectations. Hypothesis *8*, postulating divergent and inconsistent results across environments was confirmed, giving a clear indication that hand-crafted reward modifications will not always lead to desired outcomes. In line with that, Hypothesis *9* predicts a universal decline in the agent's ability to achieve their own goal compared to other approaches. This hypothesis ended up contradicting the data, where it did decline the performance significantly in two environments, but in other environments (e.g. DoorMutex, Slowdown) it could keep up with the other approaches, having near identical results. This demonstrates that, while being not suitable in all situations, it can be useful in certain cases.

The approaches where it did well had in common that the custom reward modification specified penalties rather than rewards. In DoorMutex the reward was blocked before performing the selfless action, and in Slowdown, there was a negative penalty associated with touching the other agent, so this behaviour was learned to be avoided. On the other hand for Speedup, the environment where this approach performed the worst, the custom reward function gave the agent the ability to 'Specification Game' it and collect the reward by touching the agent repeatedly. Penalties should be less prone to such abuse, thus environments in which the modification can be phrased in the form of a penalty are better candidates for this approach.

Along with the problem of susceptibility to Specification Gaming, another limitation is that of scalability. Every environment this approach is applied to requires its custom reward functions. This involves human designers having to study the environment and specify it, proceeded by the need for testing and iterating on the design, due to the first limitation.

The benefits are that this approach does not require knowledge of B's reward function and can work with incomplete observations. While the use of this approach has to be carefully considered before being selected, the results make a strict recommendation against this approach untenable. Particularly in scenarios where the other results are known to fail, such as zero-sum games, this approach should be given consideration.

7.1.4 Approaches: Summary

All approaches have demonstrated strengths and weaknesses. Among the approaches examined, no 'ideal' approach has emerged, so every approach deserves further study. Of particular interest is Approach C, as the range of applications is seemingly unbound, and it carries the fewest limitations with it, as long as the collection of observations of the other agents is possible. Approach B has surprised with more positive results than anticipated; Approach A performed incredibly well, which is in line with prior expectations.

The original aim of the project was to find effective approaches to inhibit emergent selfish behaviour in RL-based systems that can generalise across environments. This can be argued to have been achieved with Approaches A and C, which universally performed well in the environments they were tested on—excluding the environment that was designed to make them fail. More work could have been done to investigate the generalisability of the approaches, yet, at least partial generalisability could be established, where different environments with different objectives, based on different scenarios all managed to have selfish behaviours in agents inhibited.

7.2 Aim and Objectives

At the start of the project, we postulated the aim of identifying and evaluating effective approaches for inhibiting selfish behaviour in RL-based agents and for that we set out seven objectives. We have managed to achieve every single one of our objectives and successfully compared three different approaches, under a variety of different circumstances. The Limitation section will go into more detail about places in which the work could have been improved, but we consider the results to address the aim satisfactorily.

7.3 Limitations

This study contained several methodological limitations. The major one is the simplicity of the environments employed. While Gridworlds are commonplace in research contexts, it is impossible to make statements about the ability of approaches succeeding in gamified environments generalising to complex, real-world situations. This project was mostly concerned with giving an initial indication of the feasibility of different approaches.

Another limitation is that all agents were exclusively trained using the Proximal Policy Optimisation (PPO) algorithm. This should not cause much of a problem, since all RL algorithms attempt to discover and approximate the optimal policy for a given reward function, and the optimal policy is identical regardless of the algorithm being employed. Yet, the "no free lunch" theorems postulate that there is no one-size-fits-all algorithm for all reinforcement learning problems, and the performance of a specific algorithm may vary depending on the problem being tackled [68]. Therefore it is possible that different algorithms, such as DQN would lead to different results. Additionally, identical network topologies and hyperparameters were used across all experiments. Examining other algorithms and hyperparameters would lend further support to the claim that the results for the approaches are general, rather than a manifestation of the experimental setup.

There are multiple limitations specific to Approach C. On the one hand, the collection of the observations of B was not performed under normal training circumstances but required a special experimental setup. Removing the necessity for that, and collecting the observations during the normal training process, where both the reward model and A would be trained simultaneously is possible, but beyond the scope of this project, as it would require significant modification to the learning algorithms.

The other limitation is related to the reward model's output weighting which is determined by a manually set constant, with the constant being $w = 1$ as it produced good results without further changes. This is more of a coincidence, where the reward model's output coincided in magnitude with A's reward function. Had different values been used, the weighting would have had to be adjusted, which would have been a manual process, reintroducing the scalability problem this approach aimed to defeat.

7.4 Future Work

This project was concerned with assessing the initial feasibility of the different approaches. While the results give an indication, investigations of the approaches in more varied and complex environments are needed to examine the ability of the approaches to succeed under those circumstances.

Further, the way these approaches would be implemented in real-world scenarios was completely unanswered. Simple situations, such as RL-based robots navigating a warehouse without getting in the way of workers, would have a clear path for implementation, where cameras can be used to learn the reward models of workers. On the other hand, more ambitious applications, such as a recommender system being considerate of the mental health of the users, have no clear path for how the user's 'reward function' could be learned. Further work could focus on establishing ways of quantifying and learning a user's best interests. There is also the possibility that these approaches are beyond the current capabilities of Reinforcement Learning, and significant advancements in the field are required first. Yet, the simpler real-world applications, like the navigation task mentioned previously, are within current capabilities and should be performed.

Other avenues for future work come as a direct result of the limitations of this project. Without needing to consider different environments, the current environments could be investigated using more RL algorithms and varied topologies. The IRL we used in particular could also be replaced with others, to see if non-preference comparison-based IRL techniques are compatible with Approach C, such as MaxEnt IRL, GAIL, and others.

Studying approaches where both the reward model and agent are trained simultaneously rather than sequentially, would also be of interest, as it defeats the problem of needing to collect observations first. As part of that, algorithms can be designed which automatically adjust the relative weighting of the original reward to the reward model, to avoid the need to manually specify it with a constant.

References

[1] H. L. Dreyfus, *Alchemy and Artificial Intelligence.* Santa Monica, CA: RAND Corporation, 1965.

[2] A. Kaplan and M. Haenlein, “Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence,” *Business horizons*, vol. 62, no. 1, pp. 15–25, 2019.

[3] M. Haenlein and A. Kaplan, “A brief history of artificial intelligence: On the past, present, and future of artificial intelligence,” *California management review*, vol. 61, no. 4, pp. 5–14, 2019.

[4] N. Bostrom, *Superintelligence.* Dunod, 2017.

[5] D. Deutsch, “Beyond reward and punishment,” *Possible Minds: Twenty-Five Ways of Looking at ai*, pp. 113–124, 2019.

[6] N. Bostrom, “The superintelligent will: Motivation and instrumental rationality in advanced artificial agents,” *Minds and Machines*, vol. 22, pp. 71–85, 2012.

[7] N. Bostrom, “Ethical issues in advanced artificial intelligence,” *Science fiction and philosophy: from time travel to superintelligence*, vol. 277, p. 284, 2003.

[8] S. Russell, *Human compatible: Artificial intelligence and the problem of control.* Penguin, 2019.

[9] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

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

[11] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed. Pearson, 2016.

[12] M. L. Puterman, “Markov decision processes,” *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[13] B. C. Csáji *et al.*, “Approximation with artificial neural networks,” *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, no. 48, p. 7, 2001.

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[16] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, pp. 279–292, 1992.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[18] D. Silver, A. Huang, C. J. Maddison, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[19] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[22] J. M. Twenge, J. Haidt, J. Lozano, and K. M. Cummins, "Specification curve analysis shows that social media use is linked to poor mental health, especially among girls," *Acta psychologica*, vol. 224, p. 103 512, 2022.

[23] J. Haidt, *Social media is a major cause of the mental illness epidemic in teen girls. here's the evidence.* https://jonathanhaidt.substack.com/p/social-media-mental-illness-epidemic, Accessed: 2023-03-11, 2023.

[24] N. Seaver, "Captivating algorithms: Recommender systems as traps," *Journal of material culture*, vol. 24, no. 4, pp. 421–436, 2019.

[25] C. E. Robertson, N. Pröllochs, K. Schwarzenegger, P. Parnamets, J. J. Van Bavel, and S. Feuerriegel, "Negativity drives online news consumption," *Nature Human Behaviour*, 2023.

[26] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, "Dissecting racial bias in an algorithm used to manage the health of populations," *Science*, vol. 366, no. 6464, pp. 447–453, 2019.

[27] C. Ross and B. Herman, *Denied by ai: How medicare advantage plans use algorithms to cut off care for seniors in need*, https://www.statnews.com/2023/03/13/medicare-advantage-plans-denial-artificial-intelligence/, 2023.

[28] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 4th ed. Pearson, 2020.

[29] V. Krakovna, J. Uesato, V. Mikulik, *et al.*, *Specification gaming: The flip side of ai ingenuity*, https://www.deepmind.com/blog/specification-gaming-the-flip-side-of-ai-ingenuity, 2020.

[30] J. Lehman, J. Clune, D. Misevic, *et al.*, "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities," *Artificial life*, vol. 26, no. 2, pp. 274–306, 2020.

[31] S. Russell, "Learning agents for uncertain environments," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 101–103.

[32] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103 500, 2021.

[33] J. Taylor, E. Yudkowsky, P. LaVictoire, and A. Critch, "Alignment for advanced machine learning systems," *Ethics of Artificial Intelligence*, pp. 342–382, 2016.

[34] A. Y. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, vol. 1, 2000, p. 2.

[35] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[36] P. Abbeel, A. Coates, and A. Y. Ng, “Autonomous helicopter aerobatics through apprenticeship learning,” *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[37] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, “Maximum entropy inverse reinforcement learning.,” in *Aaai*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[38] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” *arXiv preprint arXiv:1507.04888*, 2015.

[39] J. Ho and S. Ermon, “Generative adversarial imitation learning,” *Advances in neural information processing systems*, vol. 29, 2016.

[40] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, “Imitating driver behavior with generative adversarial networks,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 204–211.

[41] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *Advances in neural information processing systems*, vol. 30, 2017.

[42] A. Wilson, A. Fern, and P. Tadepalli, “A bayesian approach for policy learning from trajectory preference queries,” *Advances in neural information processing systems*, vol. 25, 2012.

[43] S. Schaal and C. G. Atkeson, “Learning control in robotics,” *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[44] M. M. Botvinnik, *Computers in chess: solving inexact search problems*. Springer Science & Business Media, 2013.

[45] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, “Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning,” *arXiv preprint arXiv:1809.02925*, 2018.

[46] T. Franzmeyer, M. Malinowski, and J. F. Henriques, “Learning altruistic behaviours in reinforcement learning without external rewards,” *arXiv preprint arXiv:2107.09598*, 2021.

[47] T. Q. Klassen, S. A. McIlraith, C. Muise, and J. Xu, “Planning to avoid side effects,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 9830–9839.

[48] R. Noothigattu, D. Bouneffouf, N. Mattei, *et al.*, “Teaching ai agents ethical values using reinforcement learning and policy orchestration,” *IBM Journal of Research and Development*, vol. 63, no. 4/5, pp. 2–1, 2019.

[49] P. A. Alamdari, T. Q. Klassen, R. T. Icarte, and S. A. McIlraith, “Be considerate: Avoiding negative side effects in reinforcement learning,” in *Adaptive Agents and Multi-Agent Systems*, 2022.

[50] E. Jorge, H. Eriksson, C. Dimitrakakis, D. Basu, and D. Grover, “Inferential induction: A novel framework for bayesian reinforcement learning,” 2020.

[51] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*. John Wiley & Sons, 2017.

[52] D. Cohen, M. Lindvall, and P. Costa, “An introduction to agile methods.,” *Adv. Comput.*, vol. 62, no. 03, pp. 1–66, 2004.

[53] M. O. Ahmad, J. Markkula, and M. Oivo, “Kanban in software development: A systematic literature review,” in *2013 39th Euromicro conference on software engineering and advanced applications*, IEEE, 2013, pp. 9–16.

[54] W. W. Royce, “Managing the development of large software systems: Concepts and techniques,” in *Proceedings of the 9th international conference on Software Engineering*, 1987, pp. 328–338.

[55] I. A. Kusumadarma, D. Pratami, I. P. Yasa, and W. Tripiawan, “Developing project schedule in telecommunication projects using critical path method (cpm),” *International Journal of Integrated Engineering*, vol. 12, no. 3, pp. 60–67, 2020.

[56] R. Razzouk and V. Shute, “What is design thinking and why is it important?” *Review of educational research*, vol. 82, no. 3, pp. 330–348, 2012.

[57] J. Leike, M. Martic, V. Krakovna, *et al.*, “Ai safety gridworlds,” *arXiv preprint arXiv:1711.09883*, 2017.

[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[59] G. Brockman, V. Cheung, L. Pettersson, *et al.*, *Openai gym*, 2016. eprint: arXiv:1606.01540.

[60] C. Beattie, J. Z. Leibo, D. Teplyashin, *et al.*, “Deepmind lab,” *arXiv preprint arXiv:1612.03801*, 2016.

[61] L. Engstrom, A. Ilyas, S. Santurkar, *et al.*, “Implementation matters in deep rl: A case study on ppo and trpo,” in *International Conference on Learning Representations*, 2020.

[62] A. Gleave, M. Taufeeque, J. Rocamonde, *et al.*, *Imitation: Clean imitation learning implementations*, arXiv:2211.11972v1 [cs.LG], 2022. arXiv: 2211.11972 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2211.11972.

[63] G. Bradski, “The opencv library.,” *Dr. Dobb’s Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[64] J. V. Dillon, I. Langmore, D. Tran, *et al.*, “Tensorflow distributions,” *arXiv preprint arXiv:1711.10604*, 2017.

[65] W. McKinney *et al.*, “Pandas: A foundational python library for data analysis and statistics,” *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[66] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.

[67] T. Kluyver, B. Ragan-Kelley, F. Pérez, *et al.*, *Jupyter Notebooks-a publishing format for reproducible computational workflows.* 2016, vol. 2016.

[68] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

Appendix A

Personal Reflection

A.1 Reflection on Project

The project has been an incredible journey filled with opportunities for personal and professional growth. As I look back on the entire experience, I can appreciate the positive aspects and also identify areas that could have been improved upon.

The most notable thing when considering the success of the project is that I successfully managed to deliver on all objectives—even if in a less ambitious capacity than initially hoped for. In the middle of the project, when progress was slow and repeated difficulties were encountered at every corner, an increasing amount of uncertainty began to set in, where I started prioritising tasks based on what would be the minimum required, rather than what was initially set out. The final result is a great success in that regard, having delivered some, to the best of my knowledge, original results.

As mentioned in the Limitation and Future Work sections of the conclusion, this project would have been more rigorous, if there had been more experiments. Be it, more complex environments, or the use of different RL algorithms. Many of the experiments were initially intended to be additionally performed and the results cross-referenced with the DQN algorithm. This had to be completely cut, due to insufficient time.

Yet it was a rewarding experience to work on this project, allowing me to explore the field of reinforcement learning, which I had a keen interest in for quite some time, gaining a good theoretical understanding of the underlying theory and methods used, but also have hands-on experience, permitting me to use it more easily in future projects. I also had the chance to improve my programming ability, writing over 6000 lines of Python code throughout the project—admittedly, a number that could be significantly cut down if documentation and comments were excluded, and more refactoring had been performed. Additionally, I learned the principles of scientific rigour and how to investigate questions systematically.

Despite the success in achieving the objectives, there were areas where the project could have been improved. I pushed back the implementation until fairly late, preferring to conduct further theoretical research and literature reviews, as I thought of following a more linear approach where first all research is collected, which is followed by doing the implementation. This led to the research phase being thorough but lacking a strong focus on practical feasibility, which resulted in me spending considerable time on topics that later proved to be not applicable, such as Game Theory.

A.2 Personal Reflection

On a personal level, this project taught me invaluable lessons about time management, resource allocation, and the importance of being adaptable in the face of unforeseen challenges.

One significant lesson learned is the importance of balancing planning and execution. While the extensive research and planning were essential, diving into the implementation phase earlier would have allowed me to identify dead-ends before spending weeks pursuing them. Even with the long research phase prior to the start of the implementation, new problems and a need for scientific literature have emerged as the problem domain became better known to me. My lack of knowledge about the field meant that I could not deliberately choose my direction. In future projects, it is important for me to be better aware of the limits of my own knowledge and prioritise gaining experience over steering blindly.

In hindsight, I should have prioritised the practical aspects of the project, conducting more targeted research to identify suitable approaches and techniques, and started the implementation earlier.

Another significant thing I would have done differently is setting milestones and intermediate deadlines. Despite having a working implementation of the environments and approaches, many weeks have been used to further tweak, improve, test, and experiment. This pushed the evaluation and writing of the report back, to the extent that it has compromised the quality of both. While striving for the best possible implementation was an amicable goal, it does not make sense when considering that it will not leave me with sufficient time to evaluate the results or write them up in an interesting and insightful way.

If I had one more year to work on this project, I would be able to fill that time. This is why it is important to recognise when something is sufficiently good and to let go and move on, which I have not done until the deadline forced me to.

In conclusion, while the project was successful in achieving its objectives and provided me with valuable learning experiences, there are several areas in which I could have improved my approach. This project was an immense opportunity for personal growth, being the largest, most complex, and most ambitious project I ever undertook. The lessons learned from both the positive and negative aspects will shape how I approach future projects, and better equip me to tackle complex challenges and contribute to the advancement of knowledge.

Appendix B

Ethics approval

The full Letter of Confirmation issued by the Research Ethics Committee, indicating that the study was deemed exempt from ethical review can be found on the next page. (excluded)