Environment Design for Cooperative AI

Jake Amo

Master of Science
School of Informatics
University of Edinburgh
2023

Abstract

As artificial intelligence becomes more integrated into our society, the need for strong cooperation between agents, both virtual and real, becomes increasingly evident. Cooperative multi-agent reinforcement learning looks to promote vital cooperative capabilities by training agents on specific tasks across a range of environments, such that agents may learn to solve the task through cooperation. However, the current landscape of cooperative MARL environments on which agents can be trained and develop these capabilities is limited, hindering progress in the field. This project presents three novel environments each built for testing and developing different cooperative capabilities, simultaneously expanding on the collection of environments used in cooperative MARL research. Designed with cooperation and flexibility at their core, these environments are carefully designed, presented and benchmarked to act as a catalyst for future research and we hope will provide a valuable contribution to the cooperative MARL research community for many years to come.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jake Amo*)

Acknowledgements

There's so much to be grateful for in this short year I got to experience Edinburgh, thank you.

Dr. Stefano Albrecht, thank you. - Thank you for always showing your support and enabling me to strife for more, I appreciate all your wisdom and am grateful to have briefly been a part of the ever welcoming community you have grown.

Elliot Fosong, thank you. - Thank you for all your guidance throughout these months, for all your positivity and priceless insights. I have learned so much from you in this brief time, I will be forever grateful.

My friends and family, thank you. - Thank you for so much support, and from so many miles away. I have never felt all of you quite as close while being so far.

Arthur's seat, thank you. - The views of Edinburgh from your peak really put a lot of things into perspective. Thank you for being a place of peace and happiness on so many runs.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Traditional Reinforcement Learning (RL) has thrived and gained much popularity in recent years, thanks to a combination of computational advances as well as a passionate research community, it has achieved a multitude of impressive and recognisable achievements across a range of sectors, perhaps most popularly performing impressively in multiple games (Brown and Sandholm (2018); Schrittwieser et al. (2020); Wurman et al. (2022); Bakhtin et al. (2022)). However many of these problems are intrinsically individual-focused and solitary, with a single machine tackling either a win/loss situation like in Chess, Shogi or Poker, or a non-cooperative environment like in speeding up matrix multiplication algorithms (Fawzi et al. (2022)). While these applications are important and exciting in their own right, for AI to be functionally and safely integrated into multiple parts of society it needs to develop more cooperative intelligence to engage with other agents in multi-agent settings and, especially, other humans (Dafoe et al. (2021)).

At its core, Multi-Agent Reinforcement Learning (MARL) is an extension of traditional reinforcement learning. In extending RL to the multi-agent case we encounter new challenges not present in traditional RL such as equilibrium selection problems, multi-agent credi assignment or the non-stationarity of environments from the agent's perspective. Here, rewards for an agent no longer only depend on the environment and the action taken from that agent, but also the actions of other agents (Canese et al. (2021)). The challenges in MARL often require more complex solutions as agents need to develop policies that are viable and robust not only as the environment changes but also as the strategies of other agents change.

As the results and efficacy of integrating MARL across sectors is becoming more apparent, research in MARL is rapidly expanding, giving rise to applications like it's vital role in autonomous vehicles learning to drive (Dinneweth et al. (2022); Zhou et al. (2022)) to interests in using MARL to model, navigate and optimise financial markets (Liu et al. (2022); Lussange et al. (2021)). The more specific field of fully-cooperative MARL (focusing on common-payoff tasks where every agent in a team receives the same reward) is also being recognised for its importance in areas such as warehouse management (Krnjaic et al. (2022)) and has also achieved impressive success in competitive gaming (Vinyals et al. (2019); Berner et al. (2019)).

It is important to advance research in cooperative MARL for multiple reasons, from an AI perspective it is relevant in fostering cooperation across populations (Dafoe et al. (2020)) as examples; cars should learn to cooperate and coordinate actions with other cars, cyclists or humans, while in manufacturing lines robots and humans should work together to increase efficiency and reduce accidents. However value in cooperative MARL research could extend to other fields too providing insights in neuroscience, game theory or social choice.

There is a significant challenge in the field of cooperative MARL which unless addressed could inadvertently lead to a myopic understanding of it's potential and relevance. Namely, like many RL/MARL settings, cooperative MARL relies on environments as artificial 'training grounds' for agents to learn on. Currently the number of relevant environments adopted in cooperative MARL research remains extremely limited, posing a restrictive barrier on research. A meta-analysis of the field (Gorsane et al. (2022)) found that over 60% of the papers analysed and published in cooperative RL in 2021 used only one of 2 environments. This is a problem as new algorithms may be inadvertently environment-overfitting or producing unrealistic results with respect to the ability of new algorithms to generalise across tasks. As such we note that the current landscape of environments would greatly benefit from new contributions to allow the testing of new, relevant, cooperative capabilities for agents to learn.

Part of the scarcity of adequate cooperative MARL environments also stems from difficulty in actually defining 'cooperative capabilities' and how these may be targeted and developed in an environment. Humans are generally good at identifying when something is cooperative and when it is not, and a great part of our success as a species is likely due to our ability to cooperate. However, *defining* 'cooperation' itself is a more complex task. These challenges; of identifying cooperative capabilities and implementing environments to test these serve as the catalyst for this project.

1.2 The goal of the project

After having presented the importance of cooperative MARL and the role of environments and cooperative capabilities in this context we pose the following question:

How can we design, implement and test novel MARL environments to enable a deeper understanding and exploration of new cooperative behaviours which remain underexplored due to the constraints of limited existing environments?

This question can be broken down into three distinct goals and contributions:

- A comprehensive review of the most popular current MARL environments and the cooperative behaviours they test as well as the identification and analysis of which cooperative behaviours remain underexplored or challenging to investigate.
- The careful design and Python-based implementation of streamlined and computationally efficient MARL environments tailored to challenge and test the identified cooperative behaviours.
- The integration of these environments within the EPyMARL framework (Papoudakis et al. (2021)) followed by benchmarking and analysis of the performance of existing MARL algorithms when applied to these new environments. This will provide insights into the effectiveness of current algorithms in these environments and allow researchers to expand upon the baselines set.

1.3 Structure

The remainder of the report has the following structure: Chapter 2 begins with a presentation of the Multi-Agent RL framework that will be used, followed by a a review of current popular cooperative MARL environments and the cooperative behaviours they test. Chapter 3 will dive deeper into cooperation, presenting how one should think of cooperation in the context of cooperative MARL research and identifying capabilities that current environments struggle to test. Chapter 4 discusses the creation of the environments, discussing environment dynamics as well as how it is hoped they will contribute to testing new cooperative behaviours. Chapter 5 presents the results of testing these environments on the EPyMARL framework and and analyses the performance of the algorithms tested. Finally, Chapter 6 provides the conclusions to the project and discusses possible directions for future work.

Chapter 2

Background and Related Work

2.1 Defining the Multi-Agent Reinforcement Learning problem

We begin by formalising the Reinforcement Learning problem in a Multi-Agent setting as a Partially Observable Stochastic Game (POSG) (Hansen et al. (2004)), a generalisation of the traditional Markov Decision Process (MDP) used in single agent RL, and follow the notation of (Christianos et al. (2020); Lowe et al. (2017); Foerster et al. (2018)) in the definition. The game progresses in discrete timesteps and is defined as a tuple $(\mathcal{N}, \mathcal{S}, \{O^i\}_{i\in\mathcal{N}}, \{A^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{R^i\}_{i\in\mathcal{N}})$ where $\mathcal{N}$ = {1,...,N} denotes the set of N agents and $\mathcal{S}$ denotes the state space. A set of actions $A^1, ..., A^N$ for each agent forms the joint action space denoted as $\mathcal{A} = A^1 \times ... \times A^N$ and a set of observations $O^1, ..., O^N$ for each agent forms the joint observation space denoted $O = O^1 \times ... \times O^N$. Function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is the state transition function which takes as input a current state along with a joint action and returns a distribution over successor states. Finally $R^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ denotes the reward function giving each agent i's individual reward r^i. Usually in single-agent Reinforcement Learning each agent would seek to maximise it's own discounted returns given by $R^i = \Sigma_{t=0}^{T} \gamma^t r_t^i$ by finding an optimal policy at each state, with $\gamma \in (0, 1]$ representing the discount factor and T the total timesteps of an episode. However, in a POSG, the reward of one agent's policy frequently depends on the policies of other agents and no longer on just it's own policy, thus agents should instead seek to find a policy that maximises their expected reward by also taking into account the policies of other agents. We use the notation recommended by (Albrecht (2021)) which is common place in game theory, where the problem statement can be

described instead as each of the N agents looking to find policies $\pi = (\pi_1, ..., \pi_N)$ such that $\forall_i : \pi_i \in \arg\max_{\pi_i} \mathbb{E}[R_i|\pi_i, \pi_{-i}]$ where $\pi_{-i} = \pi \backslash \pi_i$ and denotes the policies of all other $N-1$ agents not including agent i.

While it is the task of the learning algorithms (and indeed the objective in MARL as a whole) to find an optimal set of policies π, it is the environment's role in reinforcement learning to provide all of the necessary information (contained in the tuple above) to facilitate and encourage the algorithm's learning, as such each environment presented in this project will individually elaborate on each of the relevant entries. Environments will thoroughly describe the state space $\mathcal{S}$, the observation and action space for each agent given by $\mathcal{O}$ and $\mathcal{A}$ respectively, any relevant transition functions $\mathcal{P}$ as well as the reward structure $\mathcal{R}$ for each environment. The reward structure in this framework will be presented primarily as POSG games since these are more general cases of stochastic games and often are more intuitive to explain for each individual agent. However, for the purpose of testing these environments as fully cooperative, as well as the implementation of the EPyMARL framework used for evaluation (which requires rewards to be summed amongst all agents and returned as a single value), environments will also have a description of them as Decentralised Partially Observable Markov Decision Processes (Dec-POMDP). Briefly, Dec-POMDP is a specialised case of POSG where agents share the same reward function. This means that while in POSG agents may have different objectives, leading to combinations of competition and cooperation, in a Dec-POMDP there is a single reward function for all agents and thus agents are aligned entirely on the same objective. Using a Dec-POMDP structure where all the agents receive the same reward (the sum of all individual rewards) allows us to make use of the many benefits the EPyMARL framework offers to test and train different algorithms on the proposed environments.

2.2 Current Environments

It is valuable to explore and understand some of the most popular and widely used cooperative multi-agent environments. Coming up with a good environment is a non-trivial task and takes careful planning of what exactly the goal of an environment should be as well as how it should (or could) be reached by agents. Analysing these environments will provide valuable insights into what makes an environment valuable for research, what makes them popular amongst researchers and how an environment's longevity and value may be preserved as new advancements are made. Similarly we

will analyse the cooperative behaviours each environment aims to test and discuss in the next section the gaps that could be tested.

StarCraft Multi-Agent Challenge (SMAC): [1] By far the most used and tested co-operative environment in MARL (in terms of number of published papers (Gorsane et al. (2022))), SMAC (Samvelyan et al. (2019)) is an environment based on the real-time strategy (RTS) game StarCraft II focusing on micro-management and multi-agent credit assignment challenges. The game consists of two teams of units, one consisting of learned agents and the other of an automatic game AI, with the goal of destroying eachother. SMAC offers customisation through a variety of 'scenarios' based on combinations of team sizes and units available, for example "10 Marines vs 11 Marines" or "3 Stalkers vs 5 Zealots", each varying in difficulty and target cooperative behaviours. A big part of the cooperative behaviours tested in SMAC focus on learning coordination including developing formations through spatial coordination for better defensive or offensive capabilities. Agents also learn cooperative capabilities through various micro-management techniques like the temporal coordination required for agents to 'focus' enemies (a term used for agents targeting a singular enemy simultaneously, thus removing them quickly and reducing any future damage they would have dealt) agents can then move on to the next target and defeat enemies like this one by one. This also tests the temporal coordination of agents in terms of 'overkill', as agents would ideally learn to not waste shots on targets that will be finished by other agents, when they could use these to start an attack on a new enemy instead. Finally, enemy agents can also change their policy at any point, this actively tests the ally agent's abilities to cooperatively adapt their strategy to counter any changes in strategy made by the enemies. For example, ally agents may adapt by changing targets or reorganising their formations. As we see, this final cooperative capability of adaptability or 'flexibility' can be expressed in different forms and is quite broad. It is generally applicable to many MARL environments and rightly so, as understanding what is required of oneself at different points in a process and being able to adapt for the greater good of the team is undeniably a vital element of cooperation.

For completion we note two alternative versions of SMAC also present; SMAClite [2] offers a more efficient (nearly identical) re-implementation of the environment. More recently SMACv2 was released (Ellis et al. (2022)) to tackle some previously unrecognised limitations in the original environment relating to it not being sufficiently

[1] SMAC repository: `https://github.com/oxwhirl/smac`

[2] Code for SMAClite repository: `https://github.com/uoe-agents/smaclite`

stochastic and showing how the building and designing of a robust environment is often difficult where unpredictable vulnerabilities or exploitative policies are not entirely uncommon.

Multi-agent Particle Environments (MPE): [3] A framework containing multiple (often) simple but interesting and useful environments for testing cooperative capabilities, specifically in contrast to SMACs complex and computer intensive environment, MPE (Lowe et al. (2017); Mordatch and Abbeel (2017)) is significantly less computationally expensive (Papoudakis et al. (2021)). MPE also focuses on very different cooperative behaviours, with various environments focusing on communication, for example "Speaker Listener" contains an agent which must navigate as close as possible to a landmark (which is unknown to that agent) and should be 'guided' by the speaker (an agent who knows the location of the landmark) and has a discrete action space of things they can communicate to the first agent. "Tag" is another example of an environment in MPE, a 'predator-prey' environment where the prey agents want to evade the predator agents trying to catch them. Here the prey may learn to make use of spatial coordination as a cooperative capability and use their actions to constantly misdirect the predators or spread out to avoid being captured. The predators on the other hand may coordinate their actions to block off any escape routes for the prey through careful coordination or the use of 'obstacles' to also block off exits. Due to the simplicity in many of the environments, MPE offers good options to test cooperative capabilities in near isolation, like the communication or spatial coordination described above.

Level-Based Foraging (LBF): [4] A mixed cooperative-competitive game based in a grid-world, LBF (Christianos et al. (2020); Papoudakis et al. (2021)) also focuses on coordination however agents (each with a randomised associated level) must now collect food (also with associated levels) that is scattered around the environment. Successful food collection depends on the sum of the agents' levels matching or exceeding the food's level, with rewards based on the food's level and the contribution of each agent, any agent may also collect food individually as long as their level exceeds that of the food. Part of the complexity of this environment lies in it's mixed cooperative nature, where agents must not only learn when to cooperate with other agents and when to act alone but also, if they are to cooperate, which agent to cooperate with. Presenting a similar equilibrium selection challenge as the popular game theory game 'Stag Hunt'

[3] PettingZoo repository: `https://github.com/Farama-Foundation/PettingZoo/`

[4] LBF repository: `https://github.com/uoe-agents/lb-foraging/tree/master`

(Skyrms (2001)), agents should ambitiously target high level food in coordination, however, from the perspective of each individual agent this presents an opportunity cost, as another agent may never arrive and the agent would prefer to at least collect the food it can collect individually. We can identify in this game the cooperative capabilities of cooperative decision making, strategic partner selection as well as the difficult task of balancing cooperation and competition. It should be noted that this environment also offers variations allowing the game to be played fully cooperatively through agents being unable to collect food individually, as well as a mode where rewards are evenly distributed, thus more closely aligning the game to a Dec-POMDP.

Robotic-Warehouse (RWARE): [5] The most applicable environment for real-world scenarios out of the ones discussed, RWARE (Papoudakis et al. (2021); Christianos et al. (2020)) is an environment based on a standard warehouse where robots must collect and deliver shelves ('packages') to a determined destination and then return the delivered shelf to an empty location. Agents here need to cooperate to avoid potential deadlocks in spaces where only one agent can pass and thus require extensive spatial coordination, this is tested thanks to the complex and tight layouts of the environment itself which allows little room for error and requires complex joint path planning. Agents are also tested on their ability to dynamically prioritize collectively and decide which shelves to go for. As an example; an agent between two shelves (A, B) could choose to target a further away shelf (A) if going for the closer shelf (B) would mean 'taking it' from another agent who's intention was to collect shelf (B) and then forcing that other agent to travel even further (to collect shelf (A)) than they otherwise would have. As with other environments, sparsity of the rewards (as agents are only positively rewarded when a requested shelf is delivered) makes this environment particularly challenging for agents. As to obtain any reward they must first navigate the environment and form full, often complex sequences of actions. Similar to LBF, RWARE also offers rewards to be configurable between individual or cooperative rewards.

Overcooked: [6] an environment based on the popular multiplayer cooking game, Overcooked (Carroll et al. (2019); Wu et al. (2021)) has multiple agents prepare various recipes using a combination of tools and ingredients and then deliver them to a specific point. Recipes are semi-sequential in that the order of using tools on ingredients is important (for example, tomatoes should be chopped and then placed on a plate, not placed on a plate and then chopped), but the order of the placing of ingredients on a

[5] RWARE repository: `https://github.com/semitable/robotic-warehouse/tree/master`
[6] Overcooked repository: `https://github.com/DavidRother/cooking_zoo`

plate does not matter (chopped lettuce can be placed before or after chopped tomatoes when making a salad). Three cooperative capabilities are identified in the literature and referred to as 'Divide and Conquer', 'Cooperation' and 'Spatio-Temporal movement'. Divide and Conquer refers to the partitioning of sub-tasks between agents (for example to more efficiently use their time, each agent may cut different ingredients) and can easily be tested by visualising the environment to see if agents learn such behaviours. Cooperation here refers to agents working efficiently on the same subtask when required (for example if only one agent has access to lettuces and the other to the knifes they must work together to cut the lettuce in the middle counter). Finally, Spatial-Temporal movement here refers to agents avoiding blocking eachother in bottleneck points or generally obstructing the other's path (in a similar way to how this is a problem in RWARE). Overcooked also has a strong focus on capturing beliefs and intentions of other agents, agents should learn to recognise when another agent is approaching a task and recognise the intention of that agent, thus directing their attention to a different task.

Melting Pot 2.0: [7] Melting Pot 2.0 (Agapiou et al. (2023)) is a suite of environments used for testing generalisation to novel social situations that require a mixture of 'social behaviours'. Not all environments in this test suite are strictly cooperative, with many environments being competitive or specifically built where agents can be deceitful, stubborn or malicious, however all environments share the common goal that a successful solution to the environment must require some cooperation and a combination of 'social behaviours'. For instance, 'Commons Harvest: Open' is an environment that looks to promote 'Resource Sharing' and 'Teaching and Sanctioning' as two of the identified behaviours, but is intrinsically 'semi-cooperative / semi-competitive'. In this environment agents are rewarded for collecting fruit from trees, a common-pool resource which any agent can collect. Trees periodically regenerate fruit based proportionally on how much fruit remains on the tree, thus if a tree is depleted of fruit it will no longer generate further fruit. The solution here appears simple; agents should collect fruit from trees without depleting any one tree and would thus have an endless supply. In practice this environment has so far not been solved, instead leading to a 'tragedy of the commons' scenario, where agents rapidly collect as much fruit as possible in fear that if they don't others will take the fruit instead, thus collectively agents end up depleting the environment entirely. The variety of simple environments available in MeltingPot makes it a great resource for MARL research beyond pure cooperative games.

[7] Melting Pot 2.0 repository:`https://github.com/deepmind/meltingpot`

Figure 2.1: Renderings of the most popular MARL environments in the same order as listed above

It should be noted this list of environments is not exhaustive, indeed other environments such as Hanabi Bard et al. (2020) or the Google Research Football Kurach et al. (2020) environment and more exist and are also used in research, however many of these environments are limited in customisation (like Hanabi) or have already been 'solved' by previous research and thus the requirements for new environments remains. As the 'playground' and foundation for where future research on cooperative MARL algorithms should be conducted, the number of interesting and available environments must consistently increase to promote further innovation and growth in the field and minimise the risk of stagnation.

In analysing these environments the need for a structured framework of consistent naming and definitions of cooperative capabilities became apparent. Various environments have at least some overlap in the cooperative capabilities tested such as with Overcooked simply labelling a tested capability 'Cooperation' which as we've discussed could take various different forms. A more apt name could be 'Collaborative Tasking' refering to the actual concept of performing a specific task that requires collaboration between agents, and thus differentiating the behaviour from that of other cooperative capabilities. This is a topic that will be further discussed in chapter 3 but points to a potential direction for future work.

Chapter 3

Cooperative Capabilities

3.1 Identifying Cooperation

As we have discussed, defining cooperative capabilities is not an easy task and different literature takes diverse approaches to how to categorise cooperative capabilities. Perhaps the clearest partition comes from (Dafoe et al. (2020)) where it is argued that cooperation can be split into 4 primary categories: Understanding, Communication, Commitment and Institutions.

Understanding refers to the ability of agents to actually comprehend the environment around them, the ability to asses the value of certain actions and ideally be able to anticipate actions of other agents in the environment while forming beliefs of the environment. Indeed deep understanding of the environment is vital across all environments in RL and can be argued is the motivating factor in single-agent RL, where the agent's sole task is to understand the environment around it in isolation. In a multi-agent setting, understanding should also encapsulate an idea of 'beliefs' and preferences of other agents. The idea that each agent should have it's own beliefs, preferences and intentions is referred to in some literature as identity design (Conitzer (2019)) but for the purpose of cooperative MARL specifically it's important to understand that agents should be able to learn and deduce implicit information (beliefs, preferences and intentions) which we can think of as being attached to other agents, simply by observing their moves. Agents who can successfully do this and learn about the rest of the environment are adequately developing their understanding.

The next category is 'Communication', unlike understanding which may require implicit observation of the environment and other agents, communication is the explicit sharing of information between agents, often through very simple signals (due to

computational limitations) as is the case with "Speaker Listener". Communication under mixed-motive games can be problematic as agents also have to consider the possibility of deception or agents looking to misinform other agents, however in fully cooperative games communication can be an extremely powerful tool to guarantee the success of agents and promote cooperation.

Commitment is perhaps the category most tied to complex social behaviours as it is less tangible than the understanding of an environment or the sharing of information. Various games in game theory (such as the Prisoner's Dilemma) can be easily solved if agents learn to successfully commit to an action. Commitment is also powerful inversely (Deng and Conitzer (2017)), where agents instead commit to not playing certain strategies which might be detrimental to the group, thus avoiding a socially bad Nash Equilibrium (Nash Jr (1950)) and allowing agents to instead search for a new, more optimal equilibrium. Many of the environments above also depend on commitment of multiple agents, for example SMAC's 'focus fire' behaviour depends on all agents committing to attacking the same target simultaneously, if any agent defects from this strategy the result is a suboptimal performance by the agents.

Institutions is derived mostly from economics and political theory, referring to the a system that implicitly determines operational guidelines for agents through rules or norms. This is a particularly difficult cooperative category to analyse as it risks the form of negative coalitions or unfairness when a select group marginalise other agents. However institutions can play a crucial role in promoting cooperative dynamics, aligning incentives and providing a foundational structure for complex inter-agent interactions. Institutions might for instance appear in trading environments where agents who overprice their products may be ignored by other agents as an implicit idea of fairness in value traded arises.

3.2 Target Cooperative Capabilities

While these 4 categories form a strong foundation for how we can think of cooperative capabilities, often more depth and targeted analysis is required to really identify cooperative behaviours. Melting Pot (Agapiou et al. (2023)) besides providing the suite of environments previously discussed also provides a more comprehensive list of 14 different cooperative capabilities, including: 'Flexibility', 'Reciprocity', 'Resource Sharing' or 'Dynamic Coalition Forming' amongst others. All of these can infact be grouped in the 4 categories proposed by Dafoe et al. (2020), 'Flexibility' for example,

can be thought of as requiring a strong understanding of the environment and what is required of the agent in different situations (Overcooked is a prime example of an environment testing flexibility by requiring agents to adapt frequently to the task they are doing). Alternatively, 'Dynamic Coalition Forming' is precisely the sort of cooperative behaviour one would expect from the formation of institutions. It is very rare for an environment to only be testing a single cooperative capability, most environments test combinations of these and even the combinations might themselves be unique and encourage interesting dynamics between agents. Based on the examples given in the MeltingPot list and cooperative capabilities absent from it, as well as cooperative capabilities identified previously in current environments, we briefly identify the key cooperative capabilities that we want to test and build environments around:

The first cooperative capability adheres to a form of 'trust' (a subcategory of commitment) as well as an 'understanding of limitations' of other agents which we can think of as the cooperative capability of empathy and clearly falls under understanding. This environment, although simple, requires the precise coordination of actions between agents and will require temporal coordination. For agents to fully maximise on their returns they will need to take into consideration how their actions will impact the locations of other agents and, as a consequence, the actions those agents can take at the next timestep. Although it will be explored in a fully cooperative setting, in a mixed-motive game this environment has further interesting dynamics that will be discussed as it's presented.

The second environment targets a new combination of cooperative capabilities, including 'adaptive risk assesments and distributions' where agents need to determine which sections of the environment or areas of the grid are at higher risk and distribute themselves accordingly. In particular some agents may have to change their strategies to allow other agents to improve their positioning for the general benefit of the team.

We identify the idea of 'territory and ownership' as an important cooperative capability which has been overlooked in popular MARL environments. In the context of this environment specifically the idea of the performance of agents depends on third party 'property' which can be interacted with by other agents. This will be the key cooperative capability tested in the third environment along with many of the traits also explored in the second environment, due to both environments sharing many properties.

Each of the cooperative capabilities for each environment will be explained in more depth and with examples as we present each environment in the following chapter.

Chapter 4

Co-Operative Environments

This chapter focuses on the creation and development of three proposed environments. Environments will be presented by an explanation of the basic premise of the game, along with the action and observation spaces of each agent followed by the reward structure. Finally we will discuss interesting dynamics of each environment and how we hope the identified novel cooperative capabilities are learned and tested. We begin with an analysis of key general environment traits identified in the literature which are valuable for environment design and can provide motivation behind how we implement our environments:

- An environment need not be overly complex in its action space or in possible scenarios to be interesting or encourage interesting cooperative behaviours. LBF is a clear example of an environment that appears deceptively simple in it's premise but is effective at promoting interesting cooperative behaviours.
- Environments should be intrinsically easy to understand from a tactical perspective, it should at least be clear how a possible cooperative solution might come about, even if better solutions also exist. A perfect example of this is 'Tag' from MPE, the premise of the game is clear as most people have some reference to the game or have played it themselves and yet multiple possible cooperative solutions exist to trap the prey.
- Customisation is key for longevity. Most environments contain at least a selection of scenarios to test algorithms on, with further customisation on other aspects like 'RWARE's' ability to modify the number of requested shelves. Customisation allows researchers to experiment with different configurations to really understand parts of cooperation algorithms may be failing or excelling at.

One of the design constants that is present in all three environments is the option for agents to have a full observation of the environment or a simpler and more efficient ego-centric observation space. Experiments ran in this project were all done using this ego-centric observation space.

4.1 Collision Dynamics

All three environments require dealing with collision dynamics between agents, and they were managed in a very similar way. Collision dynamics refers to what happens when two or more agents would come into contact in the same cell or take actions that would lead to conflicts in the environment. Besides an agent trying to move out of bounds (which would simply leave an agent in place and count the agent's action as invalid), three main collision problems were identified and tackled:

- Two or more agents trying to move into the same cell: This was implemented such that any one of the agents attempting to make the move would be randomly selected and complete the move, the other agents would then remain in place.
- Agents trying to swap places: this occurs when two adjacent agents are trying to move to eachother's cells, for instance if agent A is at (1,1) and tries to move to (1,2) while agent B is at (1,2) and tries to move to (1,1). These actions are considered invalid and both agents must remain in their current position.
- The problem of 'cascading actions': a consequence of the above, agent's movements are fully interdependent as the validity of one agent's action may be contingent upon the successful execution of another agent's action. For instance, if agent A intends to move agent B's position, this move is valid only if B can vacate its current spot. However, B's ability to move might, in turn, depend on another agent, say C. If C obstructs B, both A and B's actions are rendered invalid due to the cascading effect of the dependency.

4.2 Dance Crew Environment:

The first environment is the simplest of the three and is proposed as a simple environment for researchers to explore how agents consider the impacts of their actions. The environment is called 'Dance Crew' and consists of a number of agents (set to 5 by

default) that must learn a dance routine amongst themselves that will be interesting for the crowd and increase their rewards. Agents have 4 standard actions {UP, DOWN, LEFT, RIGHT} whereby the agents move through the grid in these cardinal directions, and another symbolic 'FLASHY' action which maintains the agent in the same position but puts a spotlight on them, this action, if executed correctly can allow all agents to receive a much greater reward. For Dance Crew with 5 agents, the observation space is a single ego-centric vector of length 19 where the positions of other agents are relative to the specific agent observing at that point. The first two entries correspond to the observing agent's (x,y) coordinates, the third entry corresponds to the same agent's action at the previous timestep. The following 12 entries can be thought of as 'sets' of three comprising of the relative (x,y) coordinates followed by the previous action for each the remaining agents. Finally, the last 4 entries in the observation vector are the North, South, East and West distances to the edge of the grid.

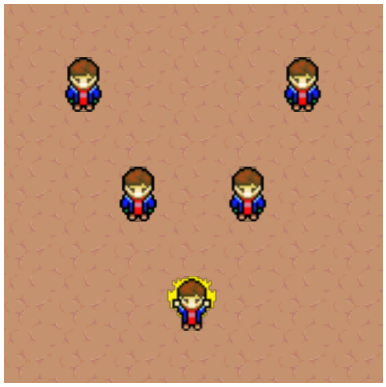

Figure 4.1: A succesfull timestep in Dance Crew; all back 4 agents moving forward and the frontmost agent performing a flashy Action.

The reward structure is explained below, for a Partially Observable Stochastic Game (POSG) setup:

- If two or more agents perform a 'flashy' action simultaneously, these agents get a reward of -2 as they are essentially competing for the spotlight at the expense of the group, all other agents get a reward of -1 regardless of their action.
- If agents are uncoordinated and two or more standard actions are performed between the 5 agents, each agent gets a reward of -1.
- If agents all perform the same standard action in unison each agent gets a reward of +1.

- If one agent performs a 'flashy' action, and all other agents perform the same standard action in unison, then the agent performing the flashy action will get a reward of +5, and the other agents will each get a reward of +2.
- To encourage diversity in the actions of agents (and promote interesting dynamics) agents are penalised for repeating the same action 3 or more consecutive timesteps. Agents doing this will receive a reward of -1 until the action is no longer repeated at which point the counter resets.

For a Dec-POMDP setup (which will be required to use EPyMARL) the rewards are simply given by the sum of what otherwise would have been the individual rewards. Thus, the minimum total reward is -10 (all agents performing a flashy action), -5 constitutes all agents performing different standard actions, 5 constitutes all agents performing the same standard action and 13 presents the maximum possible returns in a single timestep (1 flashy action and 4 synchronised standard actions).

To illustrate the target cooperative behavior of 'sympathy' towards other agents' limitations, consider the scenario with five agents where two agents are positioned on the grid's leftmost side. If any other agent chooses a leftward action, it disrupts coordination, irrespective of other agents' choices. This disruption occurs because at most one of the two agents can execute a flashy action and hope for coordination; the other agent is left with either an uncoordinated action (including also performing a flashy action) or an invalid leftward move. Therefore, agents should recognize these limitations in other agents and avoid such actions during that timestep. This principle can expand, with agents learning to avoid as much as possible actions that may compromise coordination or trap peers in restrictive positions in the future, like being trapped by other agents or near the grid's edge. The game also inherently embodies trust and commitment as agents must believe other agents will act in coordination and adequately, as well as granting opportunities for agents to perform flashy moves.

4.3 Active TD (ATD)

The environments that follow are both rooted in the iconic 'Tower Defense' (TD) game genre, each adapted to emphasize and promote significantly different cooperative dynamics between agents. While both aim for similar outcomes, they uniquely challenge agents due to their inherent differences in dynamics and the cooperative capabilities they evaluate. It is hoped that the widespread recognition of the TD genre will help

these environments be instantly understandable and accessible to researchers, promoting adoption and facilitating more intuitive analysis.

To clarify briefly, traditional TD games consist of preventing enemies which spawn at an initial position from reaching the end of their path by consistently zapping them and lowering their Hit Points (HP) until they despawn. In these environments the enemies are 'ghosts' and it is the goal of the agents to zap and despawn them before they reach the end.

Further expanding on this, in Active TD, these agents take on the role of dynamic towers. As well as zapping ghosts, agents can traverse the gridworld, though are restricted by the terrain of the environment, with agents being restricted to 'grass' cells and unable to enter cells with obstacles or cells marking the path of the ghosts.

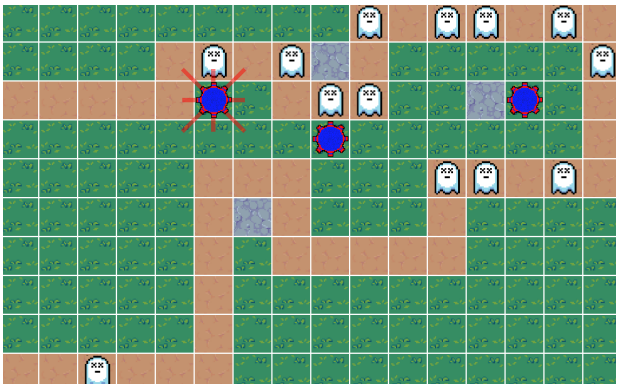

Figure 4.2: Render of the 'standard' TD map with 3 agents

Agents in the environment are homogeneous, each possessing an identical action space: of {NOOP, UP, DOWN, LEFT, RIGHT} as well as the same cardinal movements preceded by a 'ZAP' action (e.g., a ZAP-NOOP action can be executed and results in zapping surrounding ghosts followed by staying in the same position). This zapping action targets ghosts in all eight neighbouring cells reducing each ghost's HP by one. The deliberate choice to allow agents to zap and move in the same step introduces potential strategies like tailing ghosts along the path, however, this is often suboptimal and agents will require greater cooperation to maximise returns.

The observation space is ego-centric for each observing agent being considered. In order for the observation space to remain consistent throughout an episode a maximum possible number of ghosts present in the environment at any one timestep had to be established (with zero-filled entries when a ghost was absent). Hence, the observation space's dimensions depend on both agent count and the maximum number of ghosts

allowed in the environment at any one time. The first three entries of an observation capture the agent's grid position and respective 'zap cooldown timer' while each two subsequent entries detail the relative (x,y) coordinates of the other agents. Following this, each ghost is then represented by its own relative (x,y) coordinates to the agent and its remaining HP. The final four entries specify the nearest obstacle or path cell's distance in the four cardinal directions: North, South, East, and West. As an example, an environment with 5 agents and a max of 10 ghosts would have an observation space of dimension 37 $(3 + (5 \times 2) + (10 \times 3) + 4 = 37)$.

In the standard reward structure, agents receive +1 for each successful ghost zap, with the exception that, when multiple agents zap a low-health ghost causing it to despawn, rewards are instead split (e.g., three agents zapping a 2 HP ghost each receive a reward of 2/3). This is done to discourage 'overkill' (excessive zapping of ghosts) and deters potential worrying behaviors like collectively targeting low HP ghosts to artificially inflate rewards. An alternative sparse reward system exists where only the last agent to zap and remove a ghost receives a reward equivalent to the ghost's full initial HP (though, again, this is split amongst agents if multiple agents simultaneously despawn a ghost, like in the overkill cases above). In both standard and sparse reward structures all agents also receive a configurable penalty (default -15) when a ghost reaches the path's end. We note that for the Dec-POMDP case (and indeed for the EPyMARL implementation) agents simply share the total sum of what would otherwise have been the individual rewards at each timestep.

4.3.1 Customising the environment:

One key objective for these Tower Defense environments was to ensure adaptability and customization for future researchers, starting with map configuration. With strategic placement of obstacles and the path of the ghosts introducing diverse challenges and making the environment a versatile platform for assessing a range of cooperative capabilities based on its configuration. This versatility, is hoped, will help add long term value to the environments and provide multiple possible configurations used for benchmarking.

Customisation was made to expand to much more than just map configuration though and so in both TD environments researchers are able to customize:

- Full map configuration: Given a string input of G (Grass), P (Path), O (Obstacle) , A (Agent) and (S / E) for (Star / End) of path. Researchers can test maps of any

dimension and configuration (as long as the ghost path is valid).

- Ghost HP : Spawn HP of ghosts, increasing this significantly increases complexity of environment as agents may have to target the same ghost multiple times.
- HP threshold: A threshold on the sum total of HP among all ghosts (ghosts will only spawn when total HP is below the threshold).
- Max Ghosts: Maximum number of ghosts on the path at any one time. This is required to maintain constant length observation spaces, but also provides interesting interactions with the HP threshold configuration. Agents may learn for example to keep ghosts HP close to 1 until the end so ghosts do not despawn and allow a new full hp ghost to spawn, potentially leading to interesting "wave management" dynamics.
- Spawn Rate: Value in range $(0, 1]$ determining the probability of a new ghost spawning if it is possible.
- Zap Timeout: Cooldown for the tower's (in this case directly the agent's) zap action.
- Fully Cooperative: True / False, determines whether rewards are shared between agents (for a fully cooperative setting) or each agent gets it's own reward (mixed motive setting).
- Sparse Rewards: True/False, rewards as explained above.
- Penalty: The penalty given to agents when a ghost reaches the end of the path (default 15).

We note that the path for the ghosts to follow is automatically computed given a custom input string for the map by using a Breadth-First Search algorithm Bundy and Wallen (1984) and then stored for future ghosts. Thus, researchers do not have to explicitly list out all of the path in order.

Furthermore, the environments come with 4 more premade maps with varying levels of difficulty and posing different challenges for agents. Maps like 'Split' and 'Clover' separate which part of the map are accessible to different agents, while 'Circle' theoretically provides an easier map due to the high density of path cells. Meanwhile 'Maze' offers a slightly different challenge due to navigating a sort of maze with only

partial observability and will also be tested to show how customisation can change the dynamics of environments significantly.

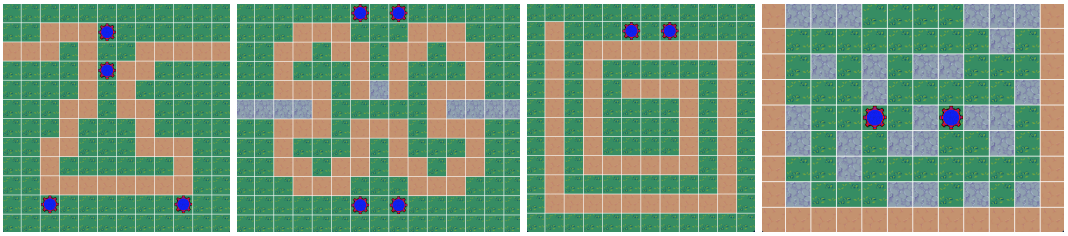

Figure 4.3: Predefined Map Configurations: ‘Split’, ‘Clover’, ‘Circle’, ‘Maze’

4.3.2 Cooperative capabilities of Active TD (ATD)

To understand the cooperative capabilities tested in this environment, consider several illustrative scenarios. Like Overcooked and RWARE, our environment demands rigorous spatial coordination, especially at ‘bottleneck points’, however, an additional layer of complexity emerges from the need for cooldown management and action timing, given the cooldown on the zap ability. For instance, an agent with a zap on cooldown should prioritize the mobility of fellow agents with available zaps rather than make the agent wait an additional timestep with the zap off cooldown. This also highlights the need for agents to balance area coverage, where implicit communication becomes crucial, as agents should deduce preferences based on their counterparts’ actions (for example if an agent is moving towards a point where there are limited ghosts, it would be unwise for another agent to do the same). A possible emergent behavior might see agents partitioning the map into zones for each agent, but then if a ghost approaches the path’s end, the nearest agent might have to trail to despawn it, leaving its zone unattended. Such scenarios underscore the significance of cooperative adaptability: it would be a testament to the agents’ collaborative intelligence if a nearby agent adjusts its policy to also cover some of the high ‘path density’ vacated cells until the tailing agent returns. While we’ve outlined potential strategies, the environment’s configurability means more cooperative capabilities could arise and would be interesting to observe in the evaluation.

4.3.3 Considerations when building the environment

We note two important attributes of the terrain of the environment and their consequences. These are referred to as 'bottleneck points' and 'path density'. Bottleneck points refers to an attribute which is common amongst other environments like Overcooked and RWARE and refers to cells which if blocked (by an agent or obstacle) prohibit movement to the other side of that blockage. These can be clearly seen near the centre of the default terrain where there are multiple such bottleneck points. 'Path density' on the other hand is unique to these TD environments and refers to how many path cells an agent could simultaneously target from a single grass cell, for example multiple cells near corners of the path have a higher path density (of 5) than cells on straight parts of the path (of 3). Path density is particularly important to consider in terms of how an environment may be solved as well as for the difficulty of the environment, as a higher path density generally will lead to an easier environment.

Significant consideration must also be placed on the reward function of any environment, as it is solely responsible for distinguish which sort of behaviours we would like to promote in agents. We note that reward structure is more complicated than merely how explicit positive rewards or penalties are given, as rewards can also be determined by the length of an episode (longer episodes allow for greater opportunities for rewards).

It was originally contemplated to have episodes terminate whenever a ghost reached the end of the path (this would have had the benefits of speeding up training time significantly as well as simplifying evaluation). It was considered, however, that interesting dynamics may arise from allowing agents some leeway in purposefully letting some ghosts reach the end in exchange for maintaining a strong territorial position or building a stronger territorial position for the future. This sort of forward planning should also be a valuable capability tested which may result in interesting agent dynamics.

Within the reward function it was considered whether to make the negative reward (penalty) for a ghost reaching the end of the path simply its remaining HP. This would be intuitive for performance of the agents as then a positive overall return would symbolise that agents were able to reduce the total HP of ghosts more than any leftover HP of ghosts who made it to the end. However, the problem with this reward function is that it would not be placing enough emphasis on stopping ghosts from reaching the end, and instead would encourage simply 'farming zaps' (being at high path density positions where agents could zap the most ghosts regularly). We hypothesise that agents would

learn to merely abuse being close to the start of the path or at strategic points as the potential benefit from simply landing more zaps would outweigh the cost of letting a lose ghost (with a low HP) reach the end. And so the penalty for ghosts reaching the end was set to a significantly larger constant value (15 by default) and should always be configured to be higher than the initial ghost HP.

This is also important for two reasons:

It encourages the cooperative capability of zone management and trust. If agents learn how to react to the environment and a good strategy to destroy ghosts then the amount of ghosts that reach the final part of the path should be minimal. A hypothetical agent that learns to watch over over the final part of the path may zap fewer ghosts in an episode than other agents but the presence/abscence of the zaps on a ghost that agent offers has a higher value for the collective than an many individual zaps which never manage to despawn a ghost. Ofcourse, in a cooperative environment both are vital and agents who can learn the importance of balance here are exactly what these environments aim to foster but it is useful from an observational point to perceive differences in tactics and the value each tactic brings to the collective. Trust also plays a significant role here, as agents should trust that peers will target ghosts at different points throughout the ghosts path, discouraging inefficient 'tail' tactics from agents.

The above is interesting to consider in map design more generally. Strategizing the placement of these interesting zones, like the vital role of an agent near the end of the path to despawn escaping ghosts at the expense of not necessarily being in a high density path area, can itself change the emphasis of which cooperative capabilities are most relevant in an environment.

4.4 Passive TD (PTD)

In this environment, the agents are not themselves the towers but instead possess the ability to strategically place or sell towers in the environment. Towers placed deterministically zap nearby ghosts in all adjacent cells and effectively change the terrain as they obstruct agent movement within the cell where the tower is placed, with agents either having to navigate around towers or sell them to get through.

This environment has some additional available configurations beyond the ones specified for the previous environment:

- Place Cooldown Timer: The number of timesteps between an agent placing a tower and being able to place another.

- Max towers: Determines the maximum possible amount of towers simultaneoulsy active belonging to a single agent.
- Zap Cooldown: How often towers will automatically zap nearby cells.
- HP Gain rate: Due to the nature of the environment, agents are much more limited during the beginning of an episode (when there are none or few towers placed) but also much stronger as they fill the environment with their towers. As such HP Gain determines how regularly the ghost's HP increases (ex: every 5 timesteps) to maintain the environment initially solvable but also challenging at later timesteps. This value is by default the same as 'place cooldown timer'.
- HP Gain number: The amount by which HP of ghost's increases (ex: HP +2) every selected number of timesteps. This value is by default the number of agents in the environment (the combination of the above two defaults guarantees the environment is always somewhat challenging).

In order to facilitate the selection of cells for ghosts to place or sell towers, agents now have a 'direction' they face and have a modified action space:{ NOOP, FORWARD, TURN LEFT, TURN RIGHT, PLACE, SELL, LEFT FORWARD, RIGHT FORWARD}. With the last two actions symbolising turning to the Left/Right respectively and then moving forward in that direction all within one timestep. The observation space for each agent here focuses on an ego-centric view where the first 3 rows are the (x,y) coordinates followed by the 'place cooldown timer' for that agent. The subsequent entries (determined by the max towers argument) are the relative (x,y) coordinates of all the towers placed by that agent. Following entries similarly constitute of the (x,y) coordinates of fellow agents followed by their respective towers (with 0s if none are placed). Finally, the last four entries dictate the distance to the closest obstacle, path cell or tower in each of the 4 cardinal directions (North, South, East, West). Finally, rewards are accumulated across all towers 'belonging' to an agent, with the agent receiving the total sum of all succesfull 'zaps' on ghosts from it's towers at any single timestep. Sparse rewards only reward the full ghost's HP (which progressively increases alongside the timesteps) on succesful despawning of a ghost. In both cases again there is a penalty if any ghosts reach the end of the path. Similar to the other two environments, in a Dec-POMDP version rewards are instead summed and shared between all agents.

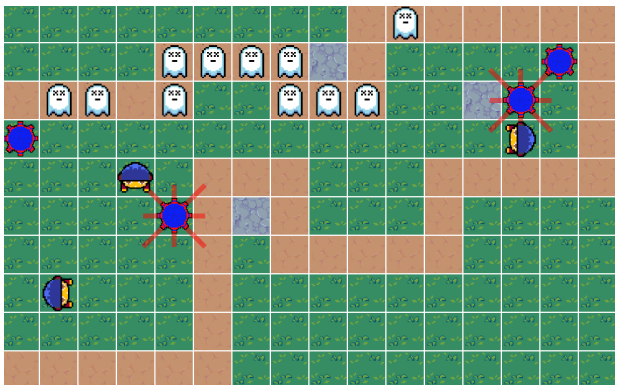

Figure 4.4: Rendering of Passive TD on the standard map with 3 agents taking random actions

4.4.1 The ‘SELL’ action

It is worth further explaining the ‘SELL’ action for this environment; any agent is allowed to sell any tower, irrespective of if the tower is one they have placed or another agent has placed, however selling a tower will only free up that tower space for the agent it originally belonged to. To illustrate, if all agents have maxed out their tower placements and one agent decides to sell another agent’s tower, only the latter (the agent who’s tower was sold) can place a new tower, not the former (the agent who sold the tower). Selling a tower, however, does reduce the remaining cooldown of the selling agent’s ‘PLACE’ action.

This sell mechanic introduces several implications for cooperation. Since the cooldown reduction of selling a tower doesn’t fully compensate for the full cooldown of its placement, agents should not regularly sell towers or they will find they do not have enough towers to deal with high HP ghosts in later waves. However, certain scenarios could justify the selling of a tower; replacing a poorly positioned tower into a high path/cell density area, using the reduced cooldown as a last-resort strategy against a ghost near the end of the path, or clearing a tower that obstructs pivotal paths could all be tactful uses of ‘SELL’. Agents should also be wary of encircling themselves with towers or obstacles, and should internalize this spatial awareness during navigation.

Furthermore, in mixed-motive settings where rewards aren’t uniformly split, the sell mechanic adds another layer of depth. Agents might be incentivized to sell another’s towers to hasten their own ’PLACE’ cooldowns, aiming for an early advantage by placing as many towers as possible. However, this tactic is very individualistic and shortsighted as effective handling of stronger ghost waves requires collective co-

operation, emphasizing the importance of a collaborative approach over individual gain.

The ability to sell and place towers introduces a new unique cooperative capability to be tested which is not present in any of the popular MARL environments through the idea of 'property and ownership' which brings in the complexities of resource management and territoriality. An agent's decision to sell a tower, especially one that doesn't belong to it, can be seen as a breach of another agent's territory or trust. Which adds an interesting layer of social dynamics, where agents must learn not just to cooperate in terms of spatial and defensive strategies but also in terms of respecting the property and territory of others.

In certain situations, one agent selling another agent's tower can be advantageous for both. For instance, imagine an agent has utilized all its tower slots, including one placed earlier in the game, perhaps to capitalize on an available cooldown. As the game progresses, the agent might find a more strategic location for that initial tower on the opposite side of the map. Could a secondary agent discern this strategic need and sell the agent's initial tower? The objective wouldn't be to reduce its own cooldown but rather to free up a tower slot for the first agent, optimizing overall strategy.

Additionally, the ability to place towers brings in a strategic depth where agents must predict and understand the long-term implications of their placements. Agents aren't just reacting to the immediate threat of ghosts but should also be collectively planning for future waves, predicting the behavior of other agents, and understanding the evolving terrain of the environment. Agents may need to forgo short-term advantages, like placing a tower near the start for immediate rewards, in favor of long-term strategic placements that benefit all agents in subsequent rounds.

Chapter 5

Evaluation

This chapter follows the guidelines set out by Gorsane et al. (2022) and follows a standard performance evaluation for Cooperative MARL. This approach ensures ease of comparison and potential enhancements by other researchers to the baselines presented here. Each algorithm was rigorously evaluated using five distinct seeds for each algorithm/environment combination, with the results reflecting the 'average test mean return' across these seeds. The results incorporate the standard deviation from all five runs as an indicator of uncertainty with this measure being particularly important (and recommended for future studies) given that environments like ATD and PTD often exhibit significant variations in returns. This variability arises from the inherent unpredictability of spawn rates where, for instance, a rate of 0.5 doesn't guarantee a consistent number of ghost spawns across episodes and thus can significantly vary potential returns.

The values of key configuration arguments is specified for each of the TD experiments in Table 5.1, this is done to allow researchers to replicate the obtained results and to act as the baseline for future research using these environments. It is worth noting due to the computational and time limitations experiments in this section are not fine tuned and results may improve with a more carefully selected set of parameters. The algorithms are each tested with the predetermined base hyperparameters (learning rate, tau, hidden layers, etc) for each algorithm given in EPyMARL. The purpose of this section is not to obtain optimal results across all algorithms but rather to analyse which algorithms appear to be performing well in which environments and hypothesise as to interesting cooperative capabilities present.

Algorithms will undergo both quantitative and qualitative assessments. Quantitatively, we'll measure total mean returns over the duration of timesteps, contrasting these

results against what a 'good' or 'optimal' return would look like under a solved version of the environment where agents performed as well as theoretically possible. Qualitatively, trained models will be rendered to visually inspect agent behaviors, offering insights into the nature of the policies they've learned. This approach will allow us to pinpoint instances of cooperation or identify potential shortcomings in the tested algorithms. Essentially, this qualitative analysis delves into the agents' behavioral patterns, exploring whether observed behaviors can be attributed to specific components or assumptions of the algorithm being evaluated.

5.1 Algorithms

We give a (very) brief introduction to each algorithm that was tested on these environments and present in the EPyMARL framework:

5.1.1 Independent Learning

These algorithms learn policies independently and simply treat other agents as part of the environment.

IQL (Independent Q-Learning): IQL treats each agent as independent, with each agent learning its own policy while treating other agents as part of the environment. This approach does not consider any explicit coordination among agents.

IA2C (Independent Advantage Actor-Critic): IA2C is an extension of the Actor-Critic method where each agent independently learns a policy and a value function, without central coordination or communication.

IPPO (Independent Proximal Policy Optimization): An adaptation of the Proximal Policy Optimization (PPO) algorithm for multi-agent settings. In IPPO, each agent independently optimizes its own policy with the PPO framework, without explicit coordination.

5.1.2 Centralised Multi-Agent Policy Gradient

These algorithms centralize training using global information but actions are selected based on each individual agent's observations.

COMA (Counterfactual Multi-Agent Policy Gradients): COMA uses a centralized critic to estimate the Q-value for each agent's action while keeping decentralized

policies. The algorithm utilizes counterfactual baselines to address the challenge of multi-agent credit assignment.

MAPPO (Multi-Agent Proximal Policy Optimization): An extension of PPO for multi-agent settings, MAPPO uses a centralized training approach with decentralized execution. Each agent learns based on global information during training but only uses its own local observations during execution.

MAA2C (Multi-Agent Advantage Actor-Critic): Building upon the Advantage Actor-Critic framework, MAA2C centralizes training using information from all agents but decentralizes execution, where each agent acts based on its own policy.

5.1.3 Value Decomposition

These algorithms decompose the joint value function into individual agents, thus actions are coordinated but policies remain agent-specific.

VDN (Value Decomposition Networks): VDN decomposes the global Q-value function into individual agent-wise value functions. By keeping agent policies decentralized, VDN ensures the joint action-value function is a sum of individual agent value functions.

Q-MIX: Q-MIX employs a mixing network to combine individual agent value functions into a joint action-value function. While agents have decentralized policies, the joint action-value function can represent interactions that are not possible with simple addition as in VDN. (Rashid et al. (2020))

EPyMARL supports two additional algorithms: MADDPG and Pareto-AC. MADDPG was excluded from testing due to its prolonged training times, which would have added to computational limitations, and its (generally) suboptimal performance in grid-world environments (Papoudakis et al. (2021)). While tests were conducted with Pareto-AC, it displayed no learning. However this was likely not a problem of the algorithm implementation itself rather of the default hyper parameters given in EPyMARL not being a good fit for these environments. Thus, to maintain clarity, the results of Pareto-AC tests were omitted from diagrams.

5.2 Results for each environment

The following subsections contain the results for each environment as well as a critical analysis into observations found through the results and through observations of the

rendered environment based on the final models after completing training.

5.2.1 Dance Crew

Beginning with Dance Crew, the goal of this environment was to see if agents could learn an optimal dance routine given an egocentric view of the environment and a small gridworld. As the simplest environment it was expected that at least some algorithms would be able to solve this environment and achieve the maximum possible combined reward of 1300 (13 over 100 timesteps).

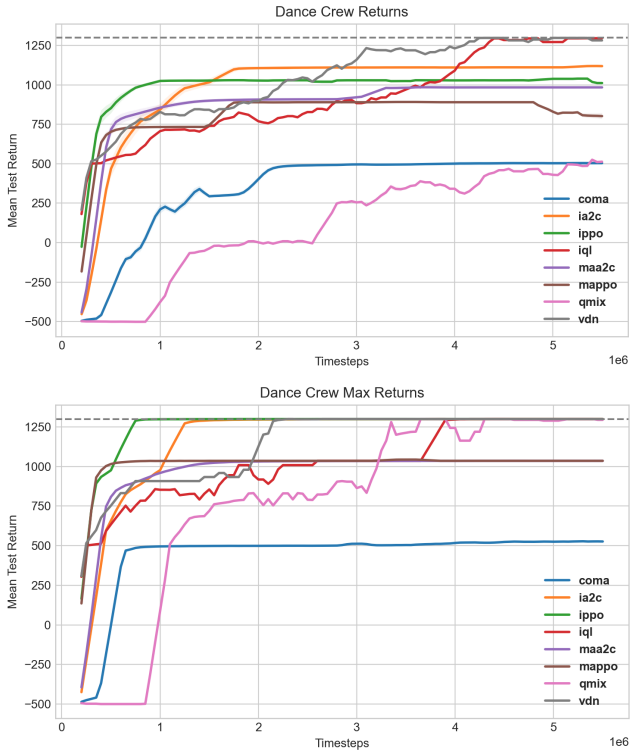

Figure 5.1: (Top): Mean test return over 5.5M timesteps and across 5 seeds, with the shaded lines representing the mean Std. Deviation across runs (Bottom): Max Reward achieved at specified timestep by any of the five seeds

In the results, all algorithms except for COMMA, MAA2C, and MAPPO achieved the maximum reward in at least one seed. COMMA consistently adhered to a suboptimal policy, with agents only performing coordinated standard actions and overlooking the significance of 'flashy' actions. Both MAA2C and MAPPO plateaued, exhibiting two timesteps of peak reward combinations followed by one of standard actions. It would appear that Centralised Multi-Agent Policy Gradient-based algorithms struggled

in this environment, while Independent Learning and Value Decomposition thrived. However, it must be noted that these suboptimal performances may simply be attributed to suboptimal hyperparameter choices, this is one of the key limitations across the evaluation of these environments, as multiple seeds needed to be trained for each algorithm across many timesteps, while it would have been preferred to do additional fine tuning this was not possible given the constraints in computational power as well as time. In rendering, it can be seen that agents demonstrably grasped the essence of evading actions that could jeopardize their peers, and understanding other's limitations. Most algorithms infact learn to minimise the instances of other agents being on the edge of the grid. This behavior suggests that agents developed a holistic understanding of their environment, recognizing the implications of their actions on peers and identifying optimal progression paths.

In a fully cooperative setting where all agents share a combined reward, this game is quite simple as it relies on agents essentially searching for a possible optimal sequence of moves for each agent at each timestep. The game may become increasingly challenging however as a mixed motive game where agents have conflicting interests due to the reward structure, whereby performing a flashy action yields the largest individual reward but requires collective cooperation from other agents. It would be interesting to see if from a mixed motive standpoint agents may for example learn some different leader-follower dynamics than in the fully cooperative case.

Two future research directions have been identified for this environment however which may in themselves be more insightful. The first relates to Dance Crew as a mixed motive game where agents receive their own rewards and must thus balance being competitive as well as cooperative, agents here would have a higher incentive to perform flashy actions themselves and this may lead to stronger conflicts between agents and a bigger test of coopeartion.

The second interesting research direction for Dance Crew relates to it's use within an 'AD-HOC' setting, namely where agents are trained within a certain population of agents to learn a dance routine but then inserted into a different scenario and must learn to cooperate with agents they have not previously encountered.

5.2.2 ActiveTD

We begin by presenting the configurations for all 'Tower Defense' style environments and configurations:

Configuration	ATD	ATD (Sparse)	PTD	PTD (Sparse)	ATD 'Maze'
Map Name	None	None	None	None	'maze'
HP Threshold	70	70	200	200	10
Max Ghost HP	6	6	19	19	3
Spawn Rate	0.5	0.5	0.5	0.5	0.2
Max Num Ghosts	14	14	12	12	3
Max Num Cannons	-	-	5	5	-
Place Timeout Reset	-	-	5	5	-
Zap Timeout Reset	3	3	2	2	2
Sparse	False	True	False	True	False
Max timesteps	100	100	100	100	100

Table 5.1: Table showing configuration of environments with relevant parameters.

Figure 5.2 shows the results of the IA2C, IPPO, MAA2C, MAPPO and QMIX algorithms on the Active Tower Defense (ATD) environment over 10 million timesteps with the configuration described in 5.1. Observing the rendering of the MAPPO agents (who performed surprisingly bad in these tests), the agents barely learn anything and instead simply take themselves to the nearest high path density point (sometimes they simply approach the nearest cell neighbouring a path regardless of density) and repeatedly zap whenever their action is off cooldown, resulting in most of the ghosts making it entirely through the path. Rarely do agents move from these points or try to cooperate further. Meanwhile, QMIX learns to optimise agent's pathing (for example by going towards the 'start' if it does not see ghosts near it that it can shoot).

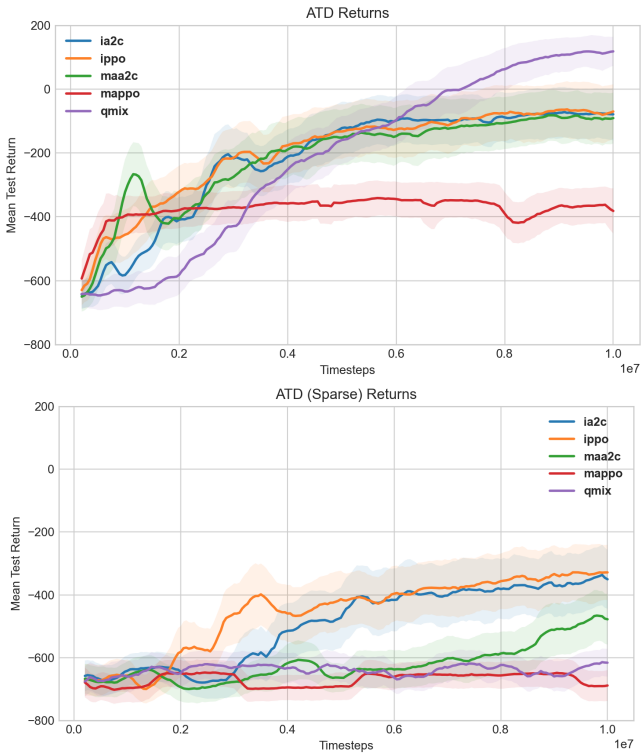

Figure 5.2: (Top): ATD mean test returns and std. averaged across 5 seeds (Bottom): ATD with Sparse rewards mean test returns and std. averaged across 5 seeds

The discrepancy between the performance of QMIX and the other algorithms appears to also be in how cases of coming back from danger are handled. Occasionally multiple ghosts will get close to the end of the path, which would require more than one agent to deal with. All algorithms (except MAPPO) learn this and send multiple agents towards the end to solve this however this is a 'temporary' solution. As we know, high density points are located near the centre of the map (near corners primarily), and so it is in the interest of agents to be close to these points as they offer the highest potential value (by zapping more ghosts). QMIX learns to navigate back to these points quickly and even (ocassionally) is happy to let a ghost through simply to stay close to these points. While other algorithms appear to be less quick to return to optimal places and are more prone to being stuck near the end of the map, at which point they have too little time to despawn oncoming ghosts.

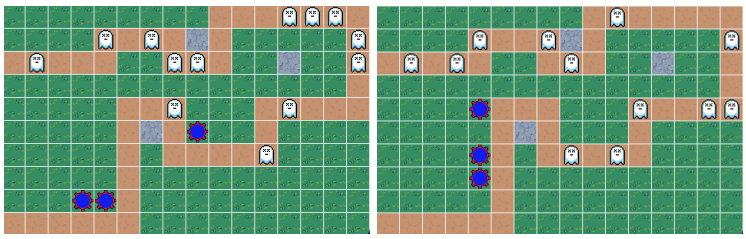

Figure 5.3: (Left): Algorithms not learning to bring back agents from tailing ghosts to the end (Right): QMIX agents succesfully returning to better positions after despawning ghosts, rather than waiting for ghosts to arrive.

Unlike with Dance Crew, defining a maximum achievable return in these tower defense environments is not obvious. As the consistency of ghosts spawning is (to an extent) stochastic and based on the spawn rate, some episodes may offer a higher opportunity for rewards than others, purely based on how regularly ghosts spawn. Some additional constraints such as the maximum number of ghosts, or, more specifically, the maximum possible total HP across all ghosts in the environment at any one timestep, limit the stochasticity of the environment by placing hard limits however the maximum achievable return is difficult to measure. As a guidance, I would recommend that since the penalty should always be double or greater the HP of a ghost (for the reasons described previously refering to reward structure), obtaining a positive return should be considered 'good' progress, as it shows ghosts are able to despawn close to twice as many ghosts as they allow to get through. More so if we calculate the roughly optimal expected returns this would look like the spawn rate $\times$ ghost HP $\times$ number of timesteps, in this case this would be ($0.5 \times 6 \times 100 = 300$). However this number is still high as it would imply ghosts should be constantly despawned instantly (on spawn). Thus for this configuration, I expect total returns in the range of 225-275 should be considered exceptional and close to maximum achievable returns.

Sparse rewards paint a different picture, firstly all algorithms seem to struggle much more with sparse rewards (this is common amongst reinforcement learning problems with sparse rewards generally) but actually we see that independent learning algorithms begin learning much sooner, with MAA2C also learning something eventually. The agent's policies are similar to that of MAPPO in the normal case however, where agents just stay close to the path without moving much and zap ghosts whenever their zap cooldown is at 0.

5.2.3 PassiveTD

We describe how an optimal solution (which never allows ghosts to reach the end) is possible by showing that towers can be strategically placed to cover all points with a high path/cell density such that any agent going through the path will be despawned.

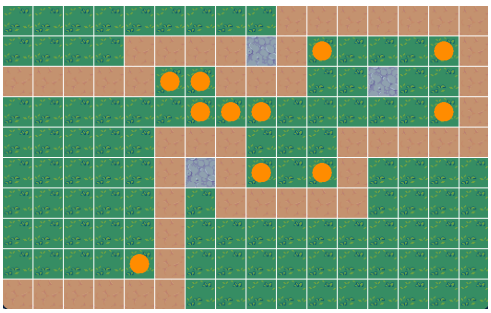

Figure 5.4: Image showing key cells where towers have a path/cell density of 4 or 5

Placing 12 of the available 15 (total) towers at the cells highlighted above and placing the remainder of the 3 towers at any cell with a path/cell density of 3 guarantees all ghosts of 19HP will be despawned. We can calculate the total path area covered by the cannons (including duplicates, since ghosts can get hit by multiple cannons in the same cell) as $(8 \times 5) + (5 \times 4) + (3 \times 3) = 62$ dividing this by 3 (how frequently towers zap) we get that each ghost will be zapped just over 20 times in a single run in expectation. However, one specific timing combination allowed a ghost to only be zapped 19 times and so the max HP of ghosts was capped at 19 in this configuration. Furthermore there are certain points where an agent could sell a placed tower, rotate, place a new tower and repeat to get an additional hit on a ghost if needed.

Figure 5.5 shows that although there exists an optimal solution which allows agents to successfully solve the environment and thus receive no penalties, even after 15 million timesteps all environments are on average still receiving negative returns. With no clear differentiation between algorithms (infact MAPPO performed comparatively well here against other algorithms compared to ATD), we turned to rendering the environment for more insights as to what could be causing agents to fail in cooperating to solve this environment. We also see again the impact sparse rewards can make on learning, limiting the learning of agents significantly with agents barely increasing their rewards after the initial 4 million timesteps.

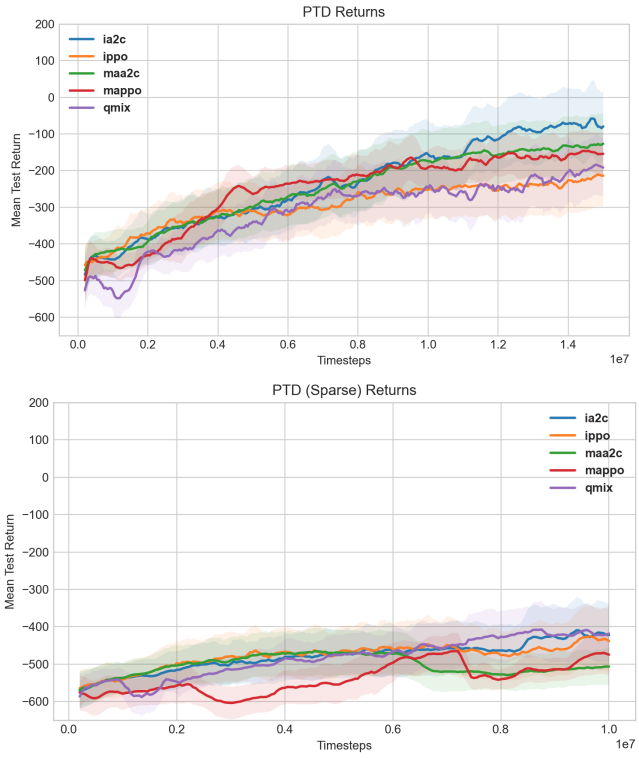

Figure 5.5: (Top): PTD mean test returns and std. averaged across 5 seeds (Bottom): PTD with Sparse rewards mean test returns and std. averaged across 5 seeds

We now focus on the rendering of the algorithms under the default (non-sparse) rewards to evaluate and hypothesise why agents are unable to learn an optimal policy.

We note that, as Figure 5.6 shows, agents predominantly place towers towards the path's beginning, exhibiting only a basic understanding of the environment without recognizing high-density points (with agents even occasionally placing towers where there are no path cells nearby). This strategy falters as higher HP ghost waves advance, and thus agents are greatly penalised at later timesteps leading to low returns.

One potential reason for such behavior could be the inherent delay between the tower placement and the subsequent rewards. As rewards are attributed to zaps from previously placed towers, algorithms might find it challenging to correlate pivotal actions, such as optimal tower placements, with delayed rewards. On the other hand, initial tower placements near the path's start offers (close to) immediate and consistent rewards due to the frequent spawning/despawning of low-HP ghosts at early timesteps, and so this might make it easier for agents to associate these early actions with the reward.

It's also noteworthy that instances of 'free riding' were observed across all algo-

rithms. In the depicted episode (which was nearing completion), not all towers have been placed, this highlights a recurring pattern where an agent would position themselves at the map's bottom left consistently and place 2 or 3 fewer towers than the other agents. Showing a clear lack in cooperation, as the agent in question was rewarded the same as other agents while doing significantly less. This was particularly evident when ghosts would pass along final path cells and the agent would still not place towers .

Moreover the use of the 'SELL' action was extremely infrequent, however, this might stem from the complexity in understanding the value of the function. In a fully cooperative context, selling any tower sacrifices some incoming rewards, and the agent faces the challenge of identifying a more strategic placement, a task which, as demonstrated, is hard for these agents.

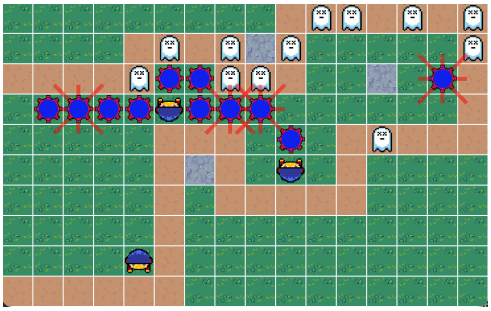

Figure 5.6: Agents after 15 million timesteps only learn to place turrets near the beginning of the path across all algorithms

5.2.4 Alternative configuration: 'Maze'

The 'maze' map illustrates how careful configuration of terrain and clear environment goals can produce diverse scenarios. In this setting, agents struggle even more with their ego-centric partial observability, as they only observe the nearest obstacles in the cardinal directions. The challenge thus becomes not only despawning the ghosts but also navigating the maze. The high density of obstacles makes any tactic of tailing ghosts entirely unfeasible, and so, agents must refine their timing, calculating when to reach cells adjacent to the path to zap nearby ghosts. Arriving too soon to a cell wastes time, but arriving late risks ghosts escaping. Effective cooperation hinges on agents' continuous assessment of their own, their peers', and the ghosts' positions, ensuring efficient task distribution and navigation.

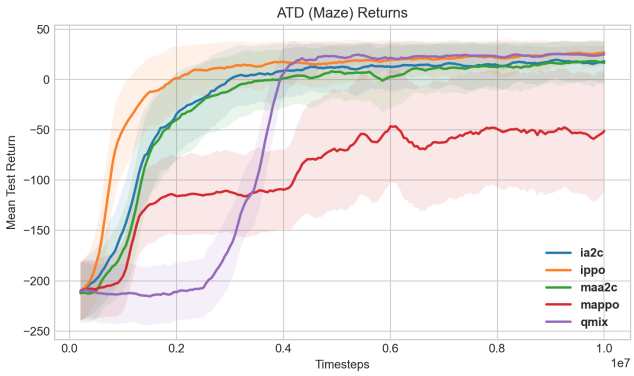

Figure 5.7: Results of running the Active TD environment on the provided ‘maze’ map configuration

This environment configuration was specifically designed to be simple and easy, featuring a low number of ghosts and consistently low ghost HP, and serving more as an illustration of the value and variety of configuration. As indicated by Figure 5.8, most algorithms excel in this task (MAPPO being the exception), with a reward approaching 30 considered notable given the ghosts’ low HP and spawn rate, leading to more limited reward opportunities. Agents typically stay near the bottom, facilitating left-to-right movement and enabling zaps on ghosts below, capitalizing on high path density cells. Moreover, they learn to use their ‘zaps’ much more cautiously to ensure availability when confronting a ghost.

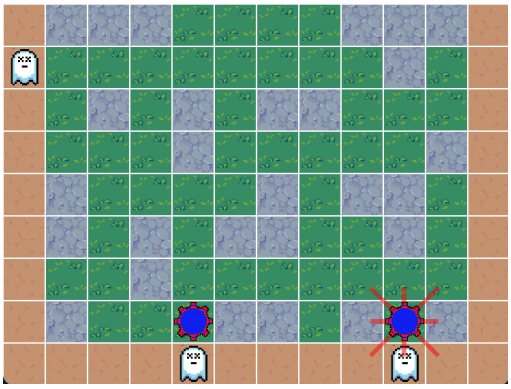

Figure 5.8: Render of a QMIX model after 10Million Timesteps in ’Maze’ configuration

Interestingly there would be a slightly better solution that agents do not learn, and that is to shift more focus towards the left of the map (closer to the beginning) which would allow agents to despawn ghosts quicker, thus allowing more new ghosts to spawn, presenting an opportunity for higher rewards.

Chapter 6

Conclusion

6.1 Summary of the project

The focus of this project was on the design, implementation and testing of novel MARL environments focused on cooperation, and is a direct response to the limited number of current MARL environments available for cooperative MARL research. This project began by introducing the problem statement, subsequently diving deep into the current landscape of MARL environments and providing analysis on how environments are built to test cooperative capabilities. This analysis also discussed the cooperative capabilities typically assessed within these environments and identified opportunities for new cooperative capabilities to be tested.

Following this three distinct environments were proposed, beginning with Dance Crew, an environment where agents must learn to coordinate their actions and learn how the consequences of their actions may limit the future actions of other agents. While Dance Crew is a simple environment and was comfortably solved by many of the algorithms tested on it, there may be value in the environment as a mixed motive game due to the potential competitive conflicts caused by the presence of flashy actions. Although simple, it serves as a valuable addition to the collection of available cooperative MARL environments.

The next two environments were built based on the Tower Defense genre, and can be thought of as an active and passive version of the game, where the agents either embody the towers directly or must strategically place them themselves. These two environments were significantly more complex than Dance Crew and introduced new cooperative capabilities to be tested. Active TD is a dynamic environment where agents must have precise spatial coordination as well as a strong understanding of their action timing. As

was observed in the evaluation, this environment provided various additional challenges such as recovering from dangerous situations which caused distinct differences in performances between algorithms.

Passive TD on the other hand introduces an entirely new aspect of ownership to cooperation, where agents can place towers strategically and should cooperate in the territory they cover to despawn ghosts effectively. When evaluating this environment we saw a surprisingly low return across all environments, with very little differences in performances between them. We speculated on potential reasoning behind these low results, emphasising the struggle of algorithms in associating actions and rewards due to the temporal delay as well as the difficulty agents faced in understanding the utility of the sell action. We then showed a new configuration of the Active TD environment, showing how modifying the environment can be used to test diverse cooperative capabilities and the importance of thoughtful configuration in environment creation.

6.2 Future Work

This project, which has a primary focus on providing new avenues for future research through the development of innovative environments, placed a strong emphasis on the future applications and longevity of it's environments at each step of the process. Environments were designed with the researcher in mind from the foundations of making environments intuitive through to laying out a structured mode for analysis of results. Future research should make use of the large configuration freedom of these environments and develop their own versions, always adhering to correct evaluation procedures and performing abalation studies whenever things are changed.

We note that although the evaluation of these environments provides an initial benchmarking of these environments, due to computational limitations hyperparameter tuning of the algorithms was not possible, thus a first step for future research should be on trying to optimise the rewards obtained by multiple algorithms on these environments.

Furthermore, and perhaps the most exciting application for future research (alongside with the customisation of new scenarios), the environments presented in this project have significant additional value as partially observable stochastic games, where agents no longer share a single reward function but instead receive their own rewards. Research into how agents balance maximising their own individual rewards against pursuing less obvious but vastly more ambitious 'optimal' rewards can provide valuable insights on algorithmic capabilities and the broader, intricate and fascinating, area of cooperation.

Bibliography

John P. Agapiou, Alexander Sasha Vezhnevets, Edgar A. Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, DJ Strouse, Michael B. Johanson, Sukhdeep Singh, Julia Haas, Igor Mordatch, Dean Mobbs, and Joel Z. Leibo. Melting pot 2.0, 2023.

Stefano Albrecht. Common inaccuracies in multi-agent rl research, 2021. URL `https://agents.inf.ed.ac.uk/blog/multiagent-rl-inaccuracies/`.

Anton Bakhtin, David J Wu, Adam Lerer, Jonathan Gray, Athul Paul Jacob, Gabriele Farina, Alexander H Miller, and Noam Brown. Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning. *arXiv preprint arXiv:2210.05492*, 2022.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280: 103216, 2020.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Alan Bundy and Lincoln Wallen. Breadth-first search. *Catalogue of artificial intelligence tools*, pages 13–13, 1984.

Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele

Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11):4948, 2021.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10707–10717, 2020.

Vincent Conitzer. Designing preferences, beliefs, and identities for artificial intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9755–9759, 2019.

Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R McKee, Joel Z Leibo, Kate Larson, and Thore Graepel. Open problems in cooperative ai. *arXiv preprint arXiv:2012.08630*, 2020.

Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground. *Nature*, 593 (7857):33–36, 2021.

Yuan Deng and Vincent Conitzer. Disarmament games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Joris Dinneweth, Abderrahmane Boubezoul, René Mandiau, and Stéphane Espié. Multi-agent reinforcement learning for autonomous vehicles: A survey. *Autonomous Intelligent Systems*, 2(1):27, 2022.

Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning, 2022.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative marl. *Advances in Neural Information Processing Systems*, 35:5510–5521, 2022.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

Aleksandar Krnjaic, Jonathan D Thomas, Georgios Papoudakis, Lukas Schäfer, Peter Börsting, and Stefano V Albrecht. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. *arXiv preprint arXiv:2212.11498*, 2022.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4501–4510, 2020.

Zhihan Liu, Miao Lu, Zhaoran Wang, Michael Jordan, and Zhuoran Yang. Welfare maximization in competitive equilibrium: Reinforcement learning for markov exchange economy. In *International Conference on Machine Learning*, pages 13870–13911. PMLR, 2022.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Johann Lussange, Ivan Lazarevich, Sacha Bourgeois-Gironde, Stefano Palminteri, and Boris Gutkin. Modelling stock markets by multi-agent reinforcement learning. *Computational Economics*, 57:113–147, 2021.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

John F Nash Jr. The bargaining problem. *Econometrica: Journal of the econometric society*, pages 155–162, 1950.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL http://arxiv.org/abs/2006.07869.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Brian Skyrms. The stag hunt. In *Proceedings and Addresses of the American Philosophical Association*, volume 75, pages 31–41. JSTOR, 2001.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Sarah A. Wu, Rose E. Wang, James A. Evans, Joshua B. Tenenbaum, David C. Parkes, and Max Kleiman-Weiner. Too many cooks: Coordinating multi-agent collaboration through inverse planning. *Topics in Cognitive Science*, n/a(n/a), 2021. doi: https://doi.org/10.1111/tops.12525. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12525.

Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert,

Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Wei Zhou, Dong Chen, Jun Yan, Zhaojian Li, Huilin Yin, and Wanchen Ge. Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Autonomous Intelligent Systems*, 2(1):5, 2022.