Role Specialisation in Heterogeneous Agents Using Reinforcement Learning

Geoffrey Nightingale

MSc Computer Science in Artificial Intelligence
The University of Bath
2023

Role Specialisation in Heterogeneous Agents Using Reinforcement Learning

Submitted by: Geoffrey Nightingale

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

Over the past decade, reinforcement learning methods have achieved human level results in complex environments such as Atari video games, the board game Go, and modern video games including Starcraft II and Dota 2. Specialisation is a universal phenomena in multi-agent systems, and has been studied extensively in animal and human settings. Reinforcement learning research in multi-agent environments has gained increased interest in recent years, however there have been few studies which analyse role specialisation within teams of heterogeneous agents.

In this study, I examine how heterogeneous agents learn specialised roles in the game of capture the flag using reinforcement learning methods. I develop a bespoke capture the flag environment and introduce four agent types with distinct attributes. I implement Proximal Policy Optimisation to train agents to play capture the flag using self-play over nine unique experiments. I then analyse whether specialisation arises from these unique agent attributes using bespoke evaluation metrics and both quantitative and qualitative methods.

I find that heterogeneous agents consistently learn specialised roles when trained using reinforcement learning and self-play. Role specialisations are consistent as the number of agents and environment sizes are increased. Furthermore, distinct adaptations in behaviour are observed in response to strategies of the opposing team, which change the nature of specialised roles over the evolution of training.

These results imply that teams of heterogeneous agents can be trained to leverage the unique capabilities of different agent types using reinforcement learning methods, without the need for need for hand-crafted behaviours or intricate reward design. This is especially important where useful behaviours for solving a problem are not known a priori.

Contents

List of Figures

List of Tables

Acknowledgements

Thanks to my supervisors Joshua Evans and Akshil Patel for their time, continual encouragement and valuable feedback.

Thanks to Dana and Maia for their love and support. I could not have done this without you. Each timestep with you is +1000 reward.

Unreal Tournament is a game I remember fondly, and was a big inspiration for this study. Thank you Epic Games for making it.

Special thanks to my employer, Contino, for funding my studies since I joined them in early 2022.

Thanks Mum and Dad for everything.

Chapter 1

Introduction

Reinforcement learning (RL) methods have achieved human-level performance in a variety of domains, such as Atari video games (Mnih et al., 2013), the board game Go (Silver et al., 2017) and complex modern video games including Starcraft II (Vinyals et al., 2019) and Dota 2 (OpenAI et al., 2019). In recent years, there has been increasing interest in applying RL methods to multi-agent settings. Specialisation is pervasive in multi-agent systems. In the natural world, species are forced to specialise due to evolutionary pressures. In human society, specialisation is a fundamental principle of modern society and the global economy. Additionally, real-world agents are often heterogeneous, with unique attributes, preferences, and beliefs. Given the importance of specialisation in multi-agent systems, it is surprising that it has been scarcely studied in the context of multi-agent RL.

In this dissertation, I investigate how heterogeneous agents learn specialised behaviours in multi-agent systems using reinforcement learning and self-play. I use the game capture the flag (CTF) as the setting from which to study specialisation.

In the remainder of this chapter, I provide background to capture the flag and theories of specialisation, research objectives, contributions of this work and the overall structure of this dissertation.

1.1 Background

Reinforcement learning (RL) can be defined as a computational approach to goal-directed learning through interaction with the environment (Evans, 2021). RL draws influence from several domains including animal psychology, neuroscience, operations research and optimal control theory (Russell, Russell, and Norvig, 2020). RL is a distinct branch of machine learning, separate from supervised and unsupervised learning methods, which respectively try to minimise error in predicting known outcomes, and find underlying structure in data sets. Multi-agent reinforcement learning (MARL) considers the interaction of multiple agents, which increases problem complexity through increased dimensionality and non-stationarity, issues which are explored further in chapter 2.

CTF is a team-based multiplayer game with two teams. The precise origins of CTF are unknown, however variations of the game appear in 19^{th} century texts such as Lehr- und Handbuch der deutschen Turnkunst (Lübeck, 1843) as 'fahnenbarlauf' and in Scouting for Boys (Baden-Powell, 1908) as 'flag raiding'. More recently, CTF gained popularity in the early 2000s through computer games such as Unreal Tournament and Quake III Arena. In CTF, each team is assigned a section of the playing field as their territory with a flag located in this area. The objective of the game is for players to acquire the opposing teams flag and return it to their territory, whilst avoiding

being tagged. Each time an agent successfully returns the opponent's flag to their area, they earn a point for their team. The game is finished when either some time limit has been reached or a team reaches a given number of points. At termination the team with the most points wins the match. Agents can 'tag' agents from the opposing team, causing them to 'respawn' in a predetermined area in their territory. Respawning generally puts agents at a disadvantage, as they are either temporarily frozen from play or need to travel extra distance to resume play. If an agent is tagged whilst carrying the flag, the flag is dropped and the tagged agent is returned to their spawning zone. In the context of MARL, CTF is an example of mixed multi-agent learning problem, as it involves both cooperative and competitive elements. Teams compete against each other whilst individual team members work together in order to gain an advantage against the opposition and maximize both individual and team rewards. Whilst games such as CTF are a simplification of reality they provide a basis from which we can improve our understanding of natural systems and develop better artificial systems. Through study of simplified game environments, we can abstract away higher level complexities to more clearly understand how agents make decisions, which actions and strategies are optimal and how strategies evolve over time.

Specialisation is a hallmark of evolution and a central pillar of modern society. In the animal world, species have evolved to learn specialised roles, such as ants taking on different tasks to improve the function of the overall colony (Bourke and Franks, 1996). Humans also have jobs that are highly specialised, and we trade with each other to obtain goods and services that we do not produce ourselves. The necessity and benefits of specialisation have been recognized since the time of Plato, who stated that society is formed by natural inequalities that allow division of labor to meet societal needs (Lee and Lane, 2007). Sociological theories also support this view, noting that division of labor in early civilizations was based on factors such as sex, age, and social status. Specialisation is also foundational in economics. Smith (1776) observed that division of labor enables teams of workers to produce more output than if each were responsible for all tasks in a manufacturing process. Ricardo (1817) further developed this idea with his theory of comparative advantage, which proposes that nations should specialise in producing goods for which they have the lowest opportunity cost, and trade for other goods where they have a higher opportunity cost. This arrangement increases welfare for all parties. Specialisation also generates interdependence between parties. Durkheim (1984) introduced the concept of organic solidarity to describe how advanced societies are held together by the interdependence of individuals who perform different tasks and have different values. In this way, specialisation helps to maintain societal stability.

1.2 Research Objectives

Given the importance of specialisation in real-world systems involving multiple agents, I find few studies which explore how heterogeneous agents learn specialised roles using MARL. This study explores the emergence of specialised roles in heterogeneous agents through the setting of CTF, a complex multi-agent environment that is also relatively unexplored in the literature. Specifically, this study aims to address the following research question:

Do heterogeneous agents learn specialised roles when trained using RL and self-play in a CTF setting?

Three main research objectives are defined to answer this research question:

1. **Develop a bespoke 2D multi-agent capture the flag environment**. I build a bespoke, 2D Capture the Flag environment, with four distinct agent types each with unique characteristics. Using a customised environment enables formulation and exploration of specific hypotheses related to specialisation. Additionally, it addresses some of the practical

challenges in MARL highlighted by Hernandez-Leal, Kartal, and Taylor (2019) such as financial and compute limitations, and reproducibility of results.

2. **Apply reinforcement learning methodologies to train agents to play capture the flag**. I implement Proximal Policy Optimisation (PPO), a powerful reinforcement learning algorithm, combined with basic self-play to train agents to play CTF. Teams of agents start off knowing nothing of how to play CTF and begin by acting randomly. Through many generations of playing each other using trial and error, agents learn effective CTF strategies.
3. **Evaluate the emergence of specialised behaviours over the course of training**. I design nine experiments and twelve evaluations metrics to analyse the emergence and evolution of agent behaviours in a range of settings.

1.3 Contributions

This study fills a gap in the literature, by examining in detail if heterogeneous agents can learn specialised roles using RL and self-play. Research in this area is limited, and unlike other RL studies on specialisation, this study uses heterogeneous agents and examines learned behaviours in detail. This is important as specialisation is fundamental to multi-agent systems and most real-world multi-agent systems feature heterogeneous agents with diverse attributes. If strong evidence is found that diverse agents can learn specialised roles using RL methods, this relieves the need for system designers to hand-craft specialised behaviours or intricate reward signals when developing multi-agent systems with heterogeneous agents. Agents can learn optimal behaviours through RL and self-play alone. This is especially important where useful behaviours for solving a problem are not known a priori.

1.4 Dissertation Structure

This dissertation is structured as follows:

1. **Introduction**
   Key background material is presented, research objectives are defined, main contributions of this study are outlined and dissertation structure explained.
2. **Background and Literature Review**
   Fundamentals of RL are examined and related studies in specialisation and CTF are reviewed.
3. **Methods**
   The CTF environment developed for this study is presented. Evaluation metrics and analytical methodologies are described.
4. **Results and Analysis**
   Nine experiments are presented and results are analysed.
5. **Discussion and Conclusion**
   Experiment results are interpreted within the context of the research objectives and related research. Limitations of this study are identified. Areas for further research are highlighted and conclusions drawn.

Chapter 2

Background and Literature Review

In this chapter further RL background is provided and related studies from the literature are reviewed. Firstly, necessary theoretical foundations of RL theory are introduced, building from fundamentals up to multi-agent methods. Following this, related studies from the literature in multi-agent specialisation and CTF are reviewed.

2.1 Reinforcement Learning

RL is concerned with understanding how actions map to numerical rewards in given situations, learnt through repeated trial and error. Actions taken in the present may produce not only immediate rewards, but also rewards in the longer term. Trial and error search and delayed rewards and defining features of RL (Sutton and Barto, 2018). Another prominent aspect of RL is the ongoing trade-off between exploration and exploitation. Agents must balance trying new actions to gain new knowledge, versus leveraging what they have already know works well from prior experience.

2.1.1 Markov Decision Processes

RL seeks to solve sequential decision problems, where an agent takes actions in response to observed states in an environment across multiple time-steps, with the goal of maximising expected cumulative rewards.

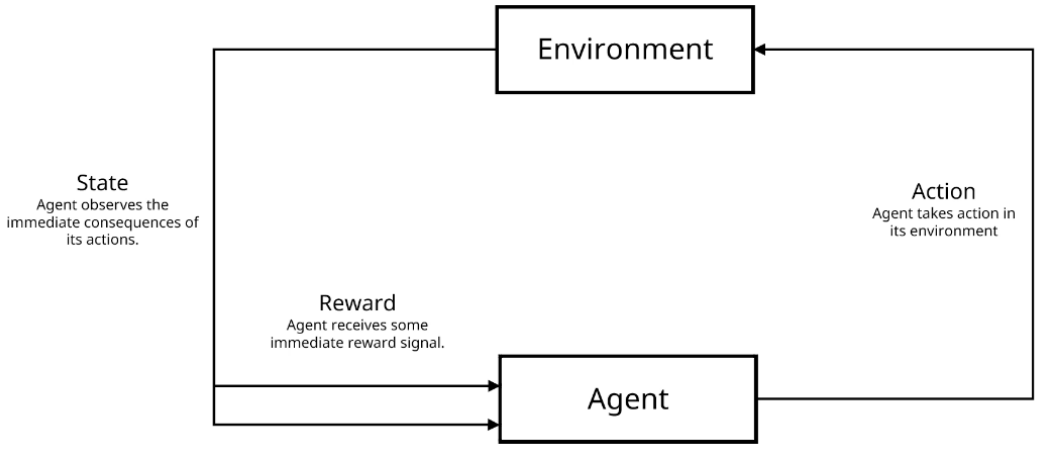

Figure 2.1: Agent-Environment Interaction Cycle (Evans, 2021)

RL problems are commonly framed in terms of an agent-environment interaction cycle, where an agent percepts the current of state of the environment, uses this information to take some action, which then leads to some new state of the environment and a reward signal received by the agent. Some of the core terminology used in RL includes:

- Agent: An actor in the environment, who can alter the state of the environment via actions. In CTF, each player in the game is an agent.
- Environment: The world in which the agents interacts. In CTF, the environment is composed of the players, objects in the environment, the game configuration and the overarching rules that govern the game.
- State: The current snapshot of the environment. In CTF, this would include the location of all the players at a given point in time, the flag locations and the current score.
- Reward: A numerical reward received by the agent after taking an action. In CTF, agents should receive a positive reward for capturing the flag. Rewards are often subjective, requiring carefule design.

Sequential decision problems can be mathematically formalised as a Markov Decision Process (MDP), defined as the tuple (S, A, P, R, γ) (Sutton and Barto, 2018). S is the set of environment states. $A(s), s \in S$ is the set of actions available in each state. $P : S \times A \times S \rightarrow [0, 1]$ is the transition function, which determines the probability of moving from state $s \in S$ to state $s' \in S$ after taking action $a \in A(s)$. $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function, which defines the reward an agent receives from taking action a in state s and moving to state s'. γ is a discount factor used to weight immediate rewards against future rewards.

Central to the MDP is the concept of the Markov property, which states that all information relevant to decision making is conveyed in the current state representation. If the Markov property holds, historic state information has no impact on decision making. This is formalised in equation 2.1, which states that the probability of the next state and reward, given knowledge of all previous states, actions and rewards is equivalent to the probability of the next state and reward, given the current state and action.

$$p(S_{t+1}, R_{t+1}|S_0, A_0, R_1, ..., R_t, S_t, A_t) = p(S_{t+1}, R_{t+1}|S_t, A_t) \tag{2.1}$$

Another central concept in RL is the notion of the policy π. A policy is a mapping from states to actions $\pi : S \rightarrow A$. The main goal in RL is to find the optimal policy π_* that maximizes the expected discounted sum of rewards. In CTF, if an agent is near the opposing team's flag (the state), the agent should pick up the flag (the action). Policies are learnt through interaction with the environment, and can be deterministic or stochastic depending on the methods used. The action prescribed by a policy is generally the action with the highest expected value for a given state. When perfect information about the environment is known, including the transition and reward functions, dynamic programming methods such as value iteration and policy iteration can be used to find the optimal policies. In practice, the transition function is generally unknown, especially in complex environments such as CTF. In these cases RL methods are used to learn policies through interaction with the environment.

Value functions quantify the expected value of returns from following a policy. There are broadly three types of values functions (Morales, 2020):

1. **State value function**: The state value function is the expected return for an agent that

begins in state s and follows policy π thereafter.

$$v_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right] \tag{2.2}$$

2. **Action value function**: The action value function is the expected return for an agent that begins in state s, takes action a and follows policy π thereafter. The action value is commonly known as the Q-function, and its estimates referred are called q-values.

$$q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \tag{2.3}$$

3. **Action advantage function**: The action advantage function quantifies how much better it is to take an action a in state s compared with the average action taken in that state under policy π. The advantage function can improve stability in training by centering updates around zero, which aids in reducing variance.

$$a_\pi(s, a) = q_\pi(s, a) - v_\pi(s) \tag{2.4}$$

2.1.2 Value-based, Policy-based and Actor Critic methods

Most RL algorithms can be categorised as either value-based, policy-based or actor-critic methods.

Value-based methods aim to find good estimates of the action-value function $Q(S_t, A_t)$. Q-learning is a popular value-based method, whereby agents maintain an estimate of the Q-value function and update it iteratively using rewards received the the highest Q-value of the new state (Sutton and Barto, 2018), illustrated in equation 2.5. Q-learning is an example of an off-policy algorithm, as it learns from data generated by a different policy than the one being used to take actions. In contrast, on-policy algorithms learn from the same policy that is used to take actions.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a) - Q(S_t, A_t)\big] \tag{2.5}$$

Policy-based methods search over the policy space using function approximation and return a policy directly by outputting a probability distribution over actions. This removes the need for learning value functions and knowledge of environment dynamics (Morales, 2020). Central to policy based methods is the policy gradient, given in equation 2.6.

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta log \pi_\theta(A_t|S_t) G(\tau)\right] \tag{2.6}$$

The policy gradient increases the log probability of actions in a trajectory proportional to the total return of that trajectory. In other words, actions which generate higher rewards in a given state are more likely to be selected. Policy-based methods can learn stochastic policies, enabling inherent exploration without the need for strategies such as the ϵ-greedy approach used in Q-learning. Other advantages of policy-based methods include better handling of continuous action spaces and improved learning stability. Policy-based methods are typically less data efficient compared to value-based methods due to their on-policy nature.

Actor critic methods lie at the intersection of value-based and policy-based methods and leverage aspects of both. An actor directly produces a policy as a probability distribution over actions, whilst a separate critic produces state values to inform the actor about the quality of chosen

actions. During training, the actor attempts to maximize expected return using information from the critic. If a chosen action is better than expected, the probability of choosing that action in the future is increased. During training, the critic attempts to reduce the difference between its estimated value for a given state and the actual rewards.

Simple RL methods such as Q-learning use bootstrapping methods to produce value estimates which are then stored in a table for retrieval when a given state is next visited. These estimates are updated in an iterative fashion through each trajectory of experience. Given enough episodes of play, values converge to their true values via the law of large numbers. In a simple game such as noughts and crosses, there are naively $3^9 = 19,683$ possible board states. Although a seemingly large number, modern computers can easily compute and store Q-values for each possible board state.

2.1.3 Deep Reinforcement Learning

In many cases, state spaces will be too large to use tabular methods such as Q-learning. The number of possible states might make it infeasible to store the values of each state in a lookup table. For instance, consider a 2D CTF environment consisting of an 11×11 grid, with 14 unique object types, such as agents, flags and different kinds of obstacles. In this environment there are 14^{121} possible game states, which is an unfathomably large number, exceeding storage capabilities of the most powerful computers. Furthermore, many states will only be visited a small number of times, leading to poor value estimates. Function approximation methods provide a solution, allowing us to assign similar states with similar value estimates.

Deep reinforcement learning (DRL) uses artificial neural networks (ANNs) as function approximators. ANNs are biologically inspired machine learning architectures, which process inputs through successive layers of weights to produce an output. Feed forward ANNs use supervised learning, where outcomes are known and backpropagation is used to iteratively adjust model weights to minimise error. Value functions such as $V(s;\theta)$ and $Q(s,a;\theta)$ can be estimated by an ANN parameterised by weights θ. The use of ANNs improves sample efficiency for large state spaces through generalisation, as similar states should produce similar value estimates. Additionally, ANNs can reduce the need for manually designed features (Hernandez-Leal, Kartal, and Taylor, 2019).

In the seminal paper 'Playing Atari with Deep Reinforcement Learning', Mnih et al. (2013) use DRL to train agents to play Atari video games through pixel input from the game screen. The authors introduce Deep Q-Networks, which combine ANNs with Q-learning, and make several novel introductions to improve stability of training via the use of separate target networks and experience replay. DQN is an example of a value-based DRL method, as it produces estimates of Q-values corresponding to each action for a given state. DQN makes use of two ANNs. The first is a primary Q-network to estimate Q-values for all possible actions in a given state, and the second is a target network to generate target Q-values during training to mitigate the moving target problem and stabilise learning. The target network is updated at a lower frequency than the Q-network, and takes copies of the weights from the Q-network periodically. The authors demonstrated that DQN learned to play Atari games at human comparable levels, a landmark achievement for the time paving the way for future DRL methods.

2.1.4 Proximal Policy Optimisation

Proximal Policy Optimisation (PPO) is an actor critic DRL algorithm, originally introduced by Schulman et al. (2017). PPO has gained widespread popularity due to its relative simplicity and excellent performance on a range of benchmarks as reported in numerous studies (Schulman et al., 2017; Andrychowicz et al., 2020; Papoudakis et al., 2021; Yu et al., 2022). PPO is used in

this study, therefore its core premise and features are briefly discussed here. Originally developed as an algorithm for individual agents, it has since been modified for multi-agent scenarios (Yu et al., 2022). Whilst value-based methods such as DQN can reuse data collected in the memory buffer over many gradient updates, policy-based based methods can typically only perform one gradient update per data sample due to their on-policy nature. PPO is able to reuse data samples multiple times through a clipped policy gradient. PPO samples data through interaction with the environment, and uses gradient ascent to optimise a surrogate objective function.

The overall objective function for PPO is given by

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t\Big[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\Big] \tag{2.7}$$

Where:

- $L_t^{CLIP}(\theta)$ is the clipped policy objective loss. This is discussed in more detail shortly.
- $L_t^{VF}(\theta)$ is the value function loss, given by $L_t^{VF}(\theta) = E_t[(V^{pred}(s_t) - V^{target}(s_t))^2]$
- $S[\pi_\theta](s_t)$ is an entropy bonus given by $S[\pi_\theta](s_t) = E_t[entropy(\pi_\theta(a_t|s_t)]$. The entropy bonus encourages exploration and prevents policies from becoming deterministic early in training.
- $c1$ and $c2$ are scalar weights used to control the importance of the value function loss and entropy terms.

One of the core innovations of PPO is a clipped objective function that limits the change in action probabilities for each gradient update. This surrogate objective function is derived from the policy gradient objective but is bounded to ensure that the new policy remains 'proximal' to the previous policy. This mechanism prevents catastrophic collapses in performance observed in earlier policy gradient methods such as A2C, and also enables PPO to reuse experiences over multiple gradient updates, which is atypical for an on-policy RL algorithm. The clipped policy is defined as:

$$L^{CLIP}(\theta) = \hat{E}_t\Big[min(r_t(\theta))\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big] \tag{2.8}$$

Central to this clipped objective function is the ratio function, given by:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2.9}$$

The ratio function describes divergence between the current and old policies. When $r_t(\theta) > 1$, the action a_t in state s_t is more likely in the current policy than the old policy. Conversely, when $0 \leq r_t(\theta) \leq 1$, the action a_t in state s_t is less likely in the current policy than the old policy. The clipping function serves to limit divergence from the current policy by taking the minimum of the bounded and unbounded ratio functions multiplied by the advantage estimate $\hat{A}_t$.

2.1.5 Multi-agent Reinforcement Learning

Despite the successes of RL in single-agent and 1v1 environments, real life situations typically involve the interactions of many agents. Furthermore, these interactions are not always adversarial and the ability for multiple actors to cooperate to achieve common goals is often critical.

In recent years multi-agent problems have gained increased exposure and research focus. The domain of multi-agent reinforcement learning (MARL) is concerned with problems where multiple

agents operate in a common environment and agents aim to optimise both individual and team rewards (Zhang, Yang, and Başar, 2021). Multi-agent problems fall under three distinct categories:

1. **Fully cooperative**: Agents work together to maximise a shared long-term rewards. These scenarios are typically modelled using a common rewards model where $R_1 = R_2 = \cdots = R_N = R$.
2. **Fully competitive**: Agents work against each other in order to compete for rewards. Such scenarios are often characterised as zero-sum games where $\sum_{i \in N} R_i(s, a, s') = 0$ for any (s, a, s').
3. **Mixed**: In mixed settings, both cooperative and competitive elements are present, and no restrictions are imposed on goals and relationships among agents. CTF is an example of a mixed setting as agents must work together to score more points than the opposing team.

Multi-agent problems are commonly formalised as stochastic games or decentralized partially observable Markov decision processes (Dec-POMDPs) when states are partially observable. The former will be explored, as full observability is assumed in the CTF environment developed in this study. A stochastic game extends the MDP framework to multiple agents and is defined as the tuple (S, N, A, P, R) (Littman, 1994). Transition P and reward functions R, depend on the joint actions $A = A_1 \times \cdots \times A_N$ of all N agents, giving $R = R_1 \times \cdots \times R_N$ and $P = S \times A_1 \times \cdots \times A_N$.

Learning in multi-agent settings is more complex compared to single agent settings and additional challenges arise in the form of non-stationarity, high dimensionality, multi-agent credit assignment, global exploration and over-generalisation (Hernandez-Leal, Kartal, and Taylor, 2019). Due to these complexities, multi-agent environments are considered an open problem in RL (Dafoe et al., 2020). One of the core challenges in MARL is non-stationarity. For an agent i, and where all other agents are represented as i^{-1}, the value function now depends on both joint action $a = (a_i, a_{-i})$ and joint policy $\pi(s, a) = \Pi_j \pi_j(s, a_j)$ (Zhang, Yang, and Başar, 2021):

$$V^i_{\pi^i, \pi^{-i}}(s) := \mathbb{E}\left[\sum_{t \geq 0} \gamma^t R^i(s_t, a_t, s_{t+1}) \middle| a^i_t \sim \pi^i(.|s_t), s_0 = s \right] \tag{2.10}$$

This implies that optimal policies are dependent not only the agents policy, but also the policies of all other agents. When multiple agents are learning concurrently, the Markov property is compromised, as the environment is no longer stationary, thus eroding convergence guarantees. To mitigate this, many MARL actor critic implementations use centralised training, decentralised execution (CTDE) methods, whereby additional global information, such as policies of other agents, is fed into the critic network during training to uphold the Markov property and produce more accurate value estimates.

The technical challenges of MARL are accompanied by practical challenges including reproducibility of results, implementation challenges and access to computational resources (Hernandez-Leal, Kartal, and Taylor, 2019). Reproducibility of studies can be difficult due to the complexity of these systems and multiple sources of stochasticity including baselines, hyper-parameters, architectures and the selection of random seeds. Implementation of algorithms often contain additional non-trivial optimisations and parameter tuning which can conflate whether results are driven from tuning decisions or the methods being studied. Computational infrastructure can be a major obstacle for those interested in conducting research in MARL. For example, Jaderberg et al. (2019) trained agents on more than four years of game time using state-of-the-art computer hardware. Such training regimes will not be feasible for most researchers due to limited access to compute resources.

Despite the increased complexity of MARL, there have been numerous well publicised successes in challenging domains such as 3D soccer (Liu et al., 2019), Starcraft II (Vinyals et al., 2019), Dota II (OpenAI et al., 2019), Quake III Arena (Jaderberg et al., 2019) and hide and seek (Baker et al., 2020). Some studies aim to achieve state of the art performance, such as Vinyals et al. (2019) and OpenAI et al. (2019), which both achieve rankings in the top 1% of human players. Other studies, such as Liu et al. (2019) and Baker et al. (2020) seek to understand and analyse how agent behaviours evolve and adapt over generations of self play. Leibo et al. (2019) describe this phenomena as auto-curricula, whereby intrinsic dynamics arising from competition and cooperation force agents to adapt their behavioural policies without explicit environmental engineering.

2.2 Related Studies

Given the importance of specialisation in multi-agent systems (established in chapter 1), and recent successes in MARL, there are surprisingly few studies which examine agent specialisation in detail. Previous MARL studies on specialisation have generally focused on scenarios involving homogeneous agents and commonly lack detailed and robust analysis of learned behaviours.

Murciano, R. Millán, and Zamora (1997) use RL methods to study specialisation in a multi agent system with homogeneous agents. Agents must assemble complex objects by collecting and assembling sub-parts scattered throughout the environment. The authors use six agents and the availability of objects is non-uniform across experiments. The authors introduce a suite of metrics to measure learning efficiency and use statistical tests to analyse if learned performance is significantly different from initial performance. The authors find that agents develop affinities for certain object types, and the rarer the object type, the greater number of agents seek that object type out. By specialising in specific object types, agents are able to assemble more complex parts per simulation. The authors also show that specialising in a single object produces better outcomes and having affinities for several objects and that individual rewards produce better results than shared rewards in this environment.

Gasparrini, Solé, and Sánchez-Fibla (2019) study task specialisation using DRL methods in an environment where homogeneous agents must move objects sequentially through three areas. Agents must transition objects from area 1 to area 2 (task 1), and then from area 2 to area 3 (task 2). Agents are rewarded for transferring objects between areas. Agents and objects spawn in random locations at the beginning of each episode, adding an element of non-determinism to the environment. The authors use DDDQN and A2C algorithms with convolutional neural networks as function approximators. Specialisation is measured by the proportion of time spent each agent spends on each task. The authors find that increasing the number of agents in the environment increases specialisation. In a two agent environment, the tasks are shared equally between agents. With six agents, clear division of labour emerges, with each agent learning to favour a specific task. The authors surmise that specialisation arises as population size increases, as individuals sacrifice general cognitive abilities for communal efficiency as observed in natural systems such as ant colonies. The authors do not report any results from statistical testing to assert whether the observed differences are statistically significant.

Wang et al. (2020) combine role-based and MARL frameworks to enable agents to specialise in tasks in the game of Starcraft II. The authors leverage QMIX network architecture combined with a latent embedding space for role representation to enable agents to learn sub-task specialisation, and share learning with other agents who are responsible for similar sub-tasks. Experiments match different combinations of unit types against each other. They authors note the emergence of agent specialisation based on characteristics such as agent location and health and also find that agents learn to specialise in tasks based on the current game environment. The authors do

not deeply explore exactly how these learnt roles are different nor do they present any statistical testing to confirm that observed differences are significant. I dispute whether the authors have tested for true specialisation in this study. The presented results suggest agents have learnt something akin to context specific policies, where agents actions are dictated by circumstance such as health and location, as opposed to attributes inherent to a given class of agents.

There are few previous studies on CTF in the literature using RL, and all have only considered homogeneous agents. Studies using 2D environments use intricate reward signals and have generally achieved poor learning outcomes.

Hefny et al. (2008) study capture the flag in a fully observable 2D environment training individual agents separately. In an era predating modern DRL algorithms, the authors use a 3-layer AI system which fuses reinforcement learning, path search and supervised learning methods. The authors implement a complex reward system, which rewards agents for specific actions. Conclusions are vague, as the authors note that learned behaviours are adequate, however these claims are not supported with any evidence.

Ivanovic et al. (2014) extend the work of Hefny et al. (2008) and use Q-learning to train a centralised 'commander' to control an entire team as opposed to training agents individually. Agents are trained against three pre-programmed opponent types using detailed reward systems. The authors find that the trained agents performed poorly against the pre-programmed opponents, however results improve when map complexity is reduced.

Jaderberg et al. (2019) demonstrate that agents can achieve human-level performance in a 3D multi-agent capture the flag environment, using only pixels and points scored as input. They propose a complex agent architecture utilising population based training, a paradigm which combines traditional neural network training with genetic algorithms. The authors find that trained agents achieve superior performance to strong human players. Additionally, the trained agents were considered by human players to be more collaborative when compared with human teammates. Although the authors achieve super-human-level performance, they only use homogeneous agents in teams of two agents.

Chapter 3

Methods

This chapter outlines the CTF environment and the methods used to meet the research objectives of this study. Specifically, I describe the details of the CTF environment, the implementation of Proximal Policy Optimisation, the self-play framework used and evaluation metrics and methods.

3.1 Environment Design

To conduct this study, I have developed a bespoke 2D capture the flag environment. The following sections describe the environment design in detail.

3.1.1 Environment Mechanics and Rules

Each CTF game consists of a finite number of time-steps. The objective for each team is to score the highest number of points, through capturing the opponents flag. One point is awarded for each successful flag capture. The game environment consists of a square grid, populated with agents and environmental objects.

Move order: At each time-step, each agent selects an action. The order of which agents take actions is randomised based on an in-game dice roll. This mechanism serves to remove any first-mover advantages and also enables simple resolution of any contests where two agents might attempt to move into the same square in the grid. Additionally, this dice-roll mechanic introduces an element of randomness into the environment, leading to uncertain gameplay outcomes.

Agent actions: Once the agent order is established, each agent takes turns to make an action. The set of actions for each agent is dependent on the agent class (see table 3.4 for details), however all agents have a shared action space consisting of moving one square up, down, left, right or remaining stationary.

Flag Capture: An agent picks up the opponents flag when they move to a square adjacent to the opponent flag. Similarly, an agent successfully captures the opponents flag by moving to a square adjacent to their own flag whilst holding the opponents flag.

Tagging: An agent tags an opponent by moving into a square adjacent to an opponent. Tags reduce agent health points by a predetermined quantity dependent on the agent class. When an agent's health points reach zero, they 'respawn' at their team spawning zone, which is predefined for each map. Tags occur with 75% probability, adding an additional element of uncertainty to the game.

Health regeneration: At the end of each round of actions, all agents regenerate 0.25 HP, up to the maximum for their class type. This enables an element of self-preservation as agents can

mitigate the cost of re-spawning by avoiding opponents if it makes sense to do so.

Game termination: Each game terminates after 500 time-steps. The team with the highest score at termination wins.

Deviation from standard CTF rules: To simplify gameplay and expedite learning, two material modifications to the standard rules of capture the flag have been made. Firstly, flag captures occur regardless of the whereabouts of the agent's own flag. This deviates from standard rules, where a teams flag must be in its home position for successful capture. Secondly, when a flag-bearing agent is tagged and respawns, the opposing team's flag is returned to its home position. This deviates from standard rules, where the flag would be dropped in place, and would need to be retrieved by an agent from either team.

3.1.2 Agent Classes

I have designed four distinct agent classes, each with differing traits and abilities. Agent classes are characterised based on their mobility, strength, damage and ability to transform the environment.

Scout: The Scout is the generic agent class in our study. The scout has more health-points (HP) than other classes, and is therefore able to withstand more tags.

Guardian: The Guardian class possesses superior defensive capabilities, dealing significant tagging points when within the vicinity of its own flag. For balance, the guardian deals less tagging damage when outside its defensive zone.

Vaulter: The Vaulter class has an expanded action set and is able to move one or two squares in any direction. The vaulter can use this ability to traverse the board more quickly, and to 'jump' over obstacles. This ability does not come for free however, and the vaulter incurs a HP penalty each time it uses this ability. This makes the vaulter more vulnerable to enemy attacks as it uses this ability repeatedly. The vaulter must have a minimum level of HP in order to execute the vault ability. Additionally, the vaulter deals less tagging damage than other agent types.

Miner: The miner has an expanded action set and is able to manipulate certain tiles in the environment by picking them up and placing them. To mine a tile, the miner agent moves into a destructible tile twice in order to remove it from the board and add it to its inventory. The miner can use these abilities to create shortcuts and obstacles in the environment.

All agent classes are able to move up, down, left and right or stay stationary. All agent classes can also capture the flag, and tag opponents. These shared abilities ensure that the basic game mechanics can be fulfilled with any combination of agent types. Details of key characteristics for each agent class are outlined in table 3.1. Complete action sets for each agent class are given in table 3.4. For presentation throughout this study, each agent class is represented by a unique avatar as depicted in figure 3.1.

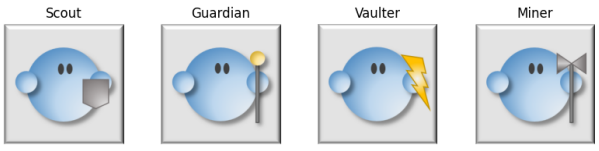

Figure 3.1: Agent type tileset

3.1.3 Environment Objects

Besides the agents, a variety of other objects populate the environment including:

Free tiles: An open space in the grid, where an agent can freely move into.

Table 3.1: Agent Class Characteristics

	Scout	Guardian	Vaulter	Miner
Number of actions	5	5	9	9
Health points	10	8	8	7
Tagging points	1	0.5 or 2.5 (def. zone)	0.5	1
Movement range	1	1	1 or 2	1
Vault cost	-	-	1.25	-
Mining ability	N	N	N	Y

Block tiles: These tiles pose obstacles to all agent classes and cannot be be moved or moved into.

Destructible tiles: These tiles can be mined and accumulated by the miner class. For added challenge, these tiles have two states - an undamaged state and a damaged state. A destructible tile is mined when a miner agents attempts to move into it twice, the first time renders the tile damaged, and the second time removes the tile from the board and adds it to the miner's inventory.

Flags: Each team has its own flag. The goal for all agents is to capture the opposing teams flag, and return it to their own flag tile.

Graphical depictions for each object are provided in 3.2. These depictions are used throughout this study.

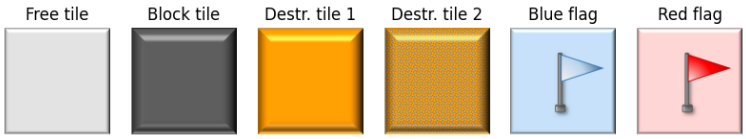

Figure 3.2: Environment objects

3.2 Stochastic Game Formulation

Given the core mechanics of the environment described thus far, the environment can be defined as a stochastic game (S, N, A, P, R) as introduced in chapter 2. Additionally, each game of CTF consists of T timesteps where $t \in \{1, 2, \ldots, T\}$ defines a specific point in time within an episode of play.

3.2.1 State Representation

The state at time step t is defined as $s_t \in S$, where S is the total game state space. S is composed of two distinct components:

1. A grid state representation of elements on the current game-screen, S_{grid}
2. A metadata representation of information that is not available through the game screen representation, but that is required to uphold Markov conditions, $S_{metadata}$

The combined state is given by $S = S_{grid} \times S_{metadata} = \{(g, m) | g \in S_{grid} \wedge m \in S_{metadata}\}$.

Grid state representation
The current game screen is represented as an $h \times h \times c$ dimensional matrix, where h is the grid height and width and c is the number of channels. The grid state is fully observable, and each agent has sight of where all objects are positioned at a given point in time. Each channel in the

grid is binary encoded to represent a different object in the environment. A value of 1 indicates that the indexed object is present in that grid cell, whereas a value of 0 indicates that the indexed object is not present in the grid cell. This representation scales well as the number of agents increases, as agents of the same type are represented using the same channel. Details of the contents of each channel are given in table 3.2.

Table 3.2: Grid State Channels

Channel	**Element**
1	Current agent
2	Block tile
3	Destructible tile 1
4	Destructible tile 2
5	Scout agents - own team
6	Guardian agents - own team
7	Vaulter agents - own team
8	Miner agents - own team
9	Scout agents - opposing team
10	Guardian agents - opposing team
11	Vaulter agents - opposing team
12	Miner agents - opposing team
13	Flag position - own team
14	Flag position - opposing team

Metadata state representation

The metadata state representation contains pertinent game state information not represented by the grid state, but which contributes to satisfying the Markov property. Such information includes game completion, which team is currently winning, agent HP and which agents currently hold the flag, if any.

This information is represented as a $1 \times k$ vector, where $k = 8 + 2(N - 1)$ and N is the total number of agents. The first eight elements of the vector contain information of the game state and the current agent, and are present in all team configurations. Vector elements thereafter correspond to the other agents in the game. The size of this vector scales linearly with the number of agents and two additional elements are added for each agent after the first. Details of each element in the metadata state vector are given in 3.3.

Table 3.3: Metadata State Vector Contents

Category	**Index**	**Description**
Game State	1	Game completion % $\in [0, 1]$
	2	Agent team % of max flag captures $\in [0, 1]$
Current Agent	3	Scout Type $\in \{0, 1\}$
	4	Guardian Type $\in \{0, 1\}$
	5	Vaulter Type $\in \{0, 1\}$
	6	Miner Type $\in \{0, 1\}$
	7	Agent HP % $\in [0, 1]$
	8	Has Flag $\in \{0, 1\}$
Other agents	9	Agent HP % $\in [0, 1]$
	10	Has Flag $\in \{0, 1\}$
	...	...
	k - 1	Agent HP % $\in [0, 1]$
	k	Has Flag $\in \{0, 1\}$

3.2.2 Number of Agents

The number of agents in this study varies depending on the experiment, with total agents ranging between four and eight: $N \in \{z \in \mathbb{Z} \mid 4 \leq z \leq 8\}$.

3.2.3 Action Space

The joint action space of all agents is given by $A = A_1 \times \cdots \times A_N$. Each agent $i \in N$, selects an action $A_i \in \{u \in \mathbb{Z} \mid 1 \leq u \leq 9\}$. At timestep t given a state s, agent i selects an action $a_{i,t} \in A_i$. Actions sets are different based on agent class as detailed in 3.4.

Table 3.4: Agent Class Action Sets

Action Index	Scout	Guardian	Vaulter	Miner
1	Up	Up	Up	Up
2	Down	Down	Down	Down
3	Left	Left	Left	Left
4	Right	Right	Right	Right
5	Stay	Stay	Stay	Stay
6	-	-	Up 2x	Place Block Up
7	-	-	Down 2x	Place Block Down
8	-	-	Left 2x	Place Block Left
9	-	-	Right 2x	Place Block Right

In this study, the Scout and Guardian agent classes have five total actions, whilst the Vaulter and Miner agent classes have nine total actions. I have implement action-masking, a technique that forces probabilities of invalid actions to zero (Huang and Ontañón, 2022). Concretely, the probability of selecting actions 6 - 9 are set to zero for Scout and Guardian classes. I experimented replacing these actions with the 'stationary' action, however I found that the use of action masking greatly improves the speed of learning. This makes intuitive sense as agents spend less time trialling redundant actions. Results of this experiment can be found in appendix B.

3.2.4 Reward Function

Rewards are defined as the joint of the individual agent rewards: $R = R_1 \times \cdots \times R_N$. At timestep t the rewards for agent i are given as $r_{i,t} = R_i(s_t, a_t, s_{t+1})$.

Reward design is an integral component of a reinforcement learning system and can significantly impact learning outcomes. I have designed a sparse reward framework in order to allow agents to organically learn strategies with minimal guidance. Agents are incentivised to win the game by as large a margin as possible, with no further instruction of how to behave. This is a point of difference from previous 2D CTF RL studies (Hefny et al., 2008; Ivanovic et al., 2014), that implement detailed reward structures for specific actions.

A reward of +1 is received by an agent when they capture the opponents flag. Conversely, agents incur a -0.5 point penalty when their own flag is captured. This schema leads to balanced gameplay and discourages overly defensive strategies. Upon game termination, each agent in the winning team receives a score bonus equal to the difference between their total team score and the opponents team score. This reward is applied globally. For example, if the winning team has a score of 5, and the losing team a score of 3, then each agent in the winning team receives a bonus of 2 points. No adjustment is made to the losing team. In practice I found penalising the losing team would often impede effective learning as flag captures could be disincentivised if associated with negative rewards from losing the match. The winning bonus has the added effect of disincentivizing collusion, whereby teams avoid conflict to maximise total scores.

From this information, the reward function can be further formalised as follows:

$$R_i(s_t, a_t, s_{t+1}) = \begin{cases} +1 & \text{if opponent flag captured by } i \\ -0.5 & \text{if own flag captured} \\ max([points_{own} - points_{opp}], 0) & \text{if terminal is true} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

3.2.5 Transition Function

The transition function is defined as $s' \sim P(s, a)$, where P provides the probability distribution of next possible states, given the current state s and a joint action a.

Individual agent actions a_i are estimated from their PPO-approximated policies π^i_θ, where θ represents the policy network weights. Further details on PPO implementation specifics are covered in the following section.

3.3 PPO Implementation

PPO has been chosen to train agents, as numerous studies show that PPO achieves competitive or superior results to other algorithms in a range of single and multi-agent environments (Schulman et al., 2017; Andrychowicz et al., 2020; Witt et al., 2020; Papoudakis et al., 2021; Yu et al., 2022). In the following sections specific PPO implementation details are given.

3.3.1 Network Design

The policy and value networks use a similar architecture to the original DQN network presented by Mnih et al. (2013). Two points of difference are:

1. Tanh activation functions are used instead of ReLU. Andrychowicz et al. (2020) find that tanh outperforms ReLU for simpler, shallow networks such as the one used in this study. One explanation is that for simple networks, the presence of 'dead' units from ReLU can be detrimental to learning.
2. An additional fully connected layer is added to allow for another level of abstraction, given additional data is added to the network through the metadata state vector.

There are two inputs to the neural network, the grid state and the metadata state vector. Before passing the grid state into the network, it goes through a pre-processing step to make the representation egocentric to the current agent.

The first hidden layer in the network uses a 2D convolution with 16 3×3 filters with stride 1 and applies a tanh activation function to the output. The second hidden layer convolves 32 3×3 filters with stride 1, again followed by a tanh activation function. The third hidden layer is a fully connected layer which concatenates the unrolled output from the second hidden layer with the metadata vector, before applying a tanh activation function creating an output size of 256 neurons. The size of the unrolled output from the second layer, denoted ϕ, varies depending on how many game objects are used in a given map. The fourth hidden layer is another fully connected layer which again applies the tanh activation function and outputs 128 neurons. At this point, the network splits into two heads:

1. An action head, which is a fully connected layer, applying a linear activation with an output for each of the 9 actions.

2. A value head, which is a fully connected layer, applying a linear activation to produce a single scalar for the state-value estimate.

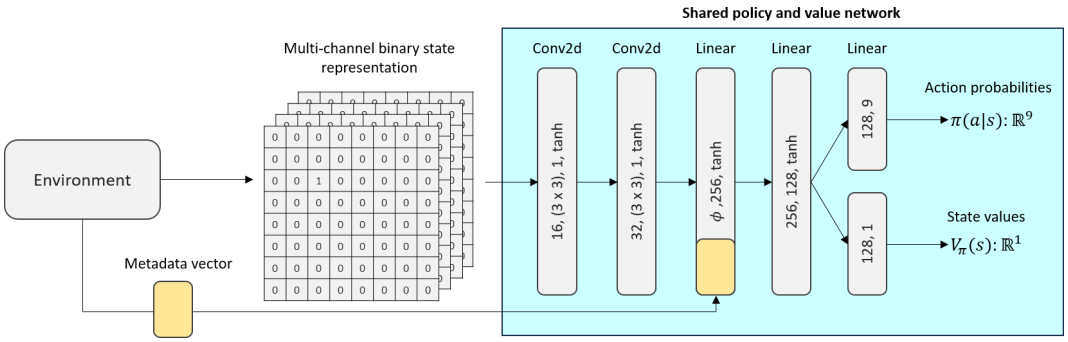

Figure 3.3: Policy and Value network

3.3.2 Parameter Sharing

I have implemented parameter sharing, a technique where network weights are shared across agents in a team. This enables training of separate policies for different agent classes using the same action and value networks. Using parameter sharing allows reuse of representations, improved sample efficiency and also reduces training time, as fewer networks need to be trained (Gupta, Egorov, and Kochenderfer, 2017). Papoudakis et al. (2021) find that parameter sharing improves performance across a range of MARL environments.

Information from the metadata vector indicating the current agent class enables the network to distinguish between agent classes during training to learn distinct policies. All studies reviewed in the literature have only used parameter sharing in environments with homogeneous agents, so this is potentially a novel implementation.

Given $C \in \mathbb{N}$ agent classes and policy π with shared parameters θ, the action for agent class $c \in C$ in state $s \in S$ is given as $a_c = \pi_\theta(c, s)$

3.3.3 Shared Memory Buffer

In addition to parameter sharing, the memory buffer is shared between agents from the same team. Rollouts are executed at the team level, as are gradient updates. This means that for each rollout, experiences are collected for all agents in the team into the memory buffer. In the optimisation step, gradient updates are made through the entire memory buffer containing all agents. At each gradient update, the policies for all agents in a team are updated in tandem. This helps to better satisfy the Markov condition, as during training each agent has access to all of the actions taken by their teammates. I hypothesise that the shared memory buffer better enables teams to learn coordinated strategies than if each agent were trained exclusively using their own experiences. I have not substantiated this claim through empirical testing, and leave this as a future area for research.

3.4 Learning

3.4.1 Self-play

Self-play is used to train agents, utilising a framework similar to that of Bansal et al. (2018). When training teams with different team compositions, two sets of networks are initialised at the beginning of training, one for each team. At each training iteration, each team's network plays against the previous iteration of the opponent. At the end of each training iteration, the

most recent policies learnt by both teams are pitted against each other over 30 duels in order to produce evaluation metrics for that training iteration (see section 3.5).

When teams are symmetric in composition, pure self-play is used, and the policy is trained against its previous self at each training iteration.

The general self-play process is detailed in algorithm 1.

Algorithm 1 Self-Play Algorithm

Initialise $\pi_{\text{team1 start}}$ and $\pi_{\text{team2 start}}$ to some starting policy
$\pi^{(1)}_{\text{team1}} \leftarrow \pi_{\text{team1 start}}$ ▷ Initial policy for Team 1
$\pi^{(1)}_{\text{team2}} \leftarrow \pi_{\text{team2 start}}$ ▷ Initial policy for Team 2
$\mathcal{R} \leftarrow \emptyset$ ▷ Initial empty rollouts
Initialize $\mathcal{M}$ as an empty collection of meterics
for $t = 1$ to $N_{training\ generations}$ **do** ▷ Training generations
    $\pi^{(t)}_{\text{team1}'} \leftarrow \pi^{(t)}_{\text{team1}}$ ▷ Store current policies
    $\pi^{(t)}_{\text{team2}'} \leftarrow \pi^{(t)}_{\text{team2}}$
    for $i = 1$ to 2 **do** ▷ Train teams against opponents previous policy
        for $j = 1$ to $N_{training\ steps}$ **do**
            $\mathcal{R} \leftarrow$ Generate rollouts using $\pi^{(t)}_{\text{team i}}$ against $\pi^{(t)}_{\text{team (3−i)}'}$
            Update $\pi^{(t)}_{\text{team i}}$ using gradient $\nabla J(\pi^{(t)}_{team\ i})$ from $\mathcal{R}$
        end for
    end for
    for $k = 1$ to N_{duels} **do** ▷ Dueling for metric generation
        $\mathcal{M}_k \leftarrow$ Duel: $\pi^{(t)}_{\text{team1}}$ vs $\pi^{(t)}_{\text{team2}}$
        Store $\mathcal{M}_k$
    end for
end for

3.4.2 Hyperparameters

One of the practical challenges in RL is finding a set of hyperparameters conducive of effective learning. Due to the sparse nature of rewards this environment, established defaults do not work well. I perform a grid search on key hyperparameters to identify a combination that produces effective learning. Results are provided in appendix C.

3.5 Evaluation Metrics and Analytical Methods

To evaluate the emergence of specialisation, I have devised a range of evaluation metrics and use both quantitative and qualitative methods for analysis.

3.5.1 Evaluation Metrics

I have designed a range of evaluation metrics that enable effective analysis of learned behaviours and the emergence of specialisation over the course of training. Many of these metrics take inspiration from those used by Baker et al. (2020), who monitor metrics such as agent movements and object interactions over training episodes. I build on this approach by introducing statistical testing to the results (see 3.5.2).

I have devised twelve metrics covering flag interactions, tagging, positional frequency, co-ordination with other agents, and interaction with environmental objects. These metrics enable identification of specialised behaviours. Agents who have disproportionate flag captures are deemed to specialise in flag capturing. Likewise agents who make disproportionate tags, and spend a significant amount of their time in proximity to their own flag are deemed to have specialised in defensive behaviours. Agents who spend relatively more time adjacent to a teammate have learnt some form of cooperative behaviour, and those agents who mine and place blocks have learnt how to

manipulate the environment. Metric definitions are given in table 3.5. Metrics are gathered by duelling the learned policies of both teams over 30 matches and averages are reported at the end of each training generation. All metrics are reported at both agent and team levels.

Table 3.5: Evaluation Metrics

Category	**Metric**	**Description**
Flag Capture	Flag pickups	Count of flag pickups
	Flag captures	Count of successful flag captures
	Flag dispossessions	Count of tags causing a flag-bearing opponent to drop the flag.
Tagging	Tag count	Count of tags made
	Tag respawn count	Count of tags that lead to the opponent respawning.
Positional	Steps in defensive zone	Number of time-steps spent in the agent's defensive zone. The defensive zone is defined as within three squares of the agent's own flag.
	Steps in attacking zone	Number of time-steps spent in the agent's attacking zone. The attacking zone is defined as within three squares of the opponent's flag.
	Steps adjacent to teammate	Number of time-steps spent adjacent to a teammate. Indicative of co-operation and coordinated movements.
	Steps adjacent to opponent	Number of time-steps spent adjacent to an opponent.
Blocks	Blocks mined	Count of blocks mined from the map. Only applies to the miner agent.
	Blocks laid	Count of blocks placed on the map. Only applies to the miner agent.
	Blocks laid distance from own flag	The average distance of blocks laid from the agents own flag. Blocks laid closer to an agent's own flag imply defensive intent

3.5.2 Quantitative Analysis

For quantitative analysis, the metrics described above are generated over five runs for each experiment. Across teams and agents within teams, p-values between metric means are computed using Welch's t-test to discern if differences in behaviours are statistically significant. Welch's-t test is chosen over a standard two-sample t-test, as there is no strong basis to assume that variances are equal between the distributions of metrics generated across different teams and agents. Empirically, the variances are found to be quite different. Results are reported at two levels:

1. Agent level - to determine if different behaviours have been learnt between agent class types within a team.
2. Team level - to analyse if different behaviours have been learnt between teams.

Results are analysed in the context of agent classes and their attributes, to determine whether any specialised behaviours have been learnt and if they align with expectations.

3.5.3 Qualitative Analysis

Quantitative analysis is supported with a range of qualitative approaches.

Firstly, trends in metrics over the course of training are analysed. As before, both team and agent level trends are analysed to discern how behaviours have evolved over the course of training.

Secondly, agent visitation patterns are examined at several specific points throughout training. These metrics are generated by running several duels using fixed policies at certain points during training, and then collecting the aggregate number of time-steps spent by each agent in each cell in the environment grid. This analysis provides insight into the navigation patterns of the agent classes, and how these evolve over time, allowing depiction of both agent movement and strategy.

Finally, actual gameplay using the learnt policies at the end of training are observed to analyse in-play behaviours which might be too nuanced to be captured by the other methods.

3.6 Software Design and Hardware

Details of software design, including links to the code repository are provided in appendix A.

All experiments in this study were run on a single Apple Macbook M1 Pro, with 8 cores and 32 GB of RAM.

Chapter 4

Results and Analysis

In this chapter I introduce nine experiments that test for specialisation. I describe the hypotheses for each experiment, followed by presentation and analysis of results.

4.1 Experiment Overview

This study examines nine distinct experiments, each testing for agent specialisation under different agent class combinations and environment configurations. Experiments are divided into three categories:

1. **Targeted Specialisation**: These experiments test the Guardian, Vaulter and Miner agent classes alongside a Scout teammate against two Scouts. These simple experiments allow analysis of specialisation in a mixed team compared with a team of homogeneous agents.
2. **Mixed Agent Classes**: These experiments test different mixtures of agent types in 2v2 settings to understand how mixed agent types across both teams impacts on specialisation and if agent class behaviours align with findings from earlier experiments.
3. **Complex Environments**: These experiments increase the number of agents and environment size to test weather earlier findings hold under increased complexity.

Summary details of each experiment are given in table 4.1.

Table 4.1: Experiments

#	Category	Name	Agent Config	Grid Size
1	Targeted Specialisation	The Split	2v2	11x11
2	Targeted Specialisation	The Fence	2v2	11x11
3	Targeted Specialisation	Jailbreak	2v2	11x11
4	Targeted Specialisation	One Way Out	2v2	11x11
5	Mixed Agent Classes	Keyhole	2v2	11x11
6	Mixed Agent Classes	Skittles	2v2	11x11
7	Complex Environments	Gridlocked	3v3	13x13
8	Complex Environments	The Wall	3v3	13x13
9	Complex Environments	Arena	4v4	15x15

Each experiment is run over five trials with randomised seeds to generate metrics for statistical testing. Each trial consists of 50 generations of training and each generation uses 100,000 time steps of gameplay. Therefore, each experiment uses 50M time-steps of gameplay (each team is trained on 25M time-steps). Exact details of training configurations are provided in appendix D.

For brevity and to avoid ambiguity the following notation is used throughout this section: T1 and T2 refer to Team 1 and Team 2 respectively. Agents are denoted as T1-S, T2-G, T1-M and so forth, which refer to T1 Scout, T2 Guardian and T1 Miner. T1 are red and T2 are blue in all experiments. The following notation is used to indicate statistical significance: ** $p < 0.05$; *** $p < 0.01$.

For economy, only select tables and figures presented in this chapter. The reader is directed to artefacts in appendix E as required.

4.2 Targeted Specialisation Experiments

4.2.1 Experiment Descriptions and Hypotheses

Experiment 1 - The Split. This experiment tests whether a mixed team of a Guardian and a Scout learn specialised roles. T1 comprises of a Guardian and a Scout, while T2 consists of two Scouts. A single diagonal row of block tiles separates the two teams, forcing a single area of movement in the top right quadrant of the game grid, forcing confrontation between teams. I hypothesise that the T1 Guardian will learn to defend its flag whilst the T1 Scout will assume a more attacking role.

Experiment 2 - The Fence. This experiment tests whether a mixed team of a Vaulter and a Scout learn specialised roles. T1 consists of a Vaulter and a Scout, and T2 consists of two Scouts. A row of four block tiles separates the flags of the two teams, impeding a direct path for capture and forcing the agents to take a longer route around this structure in for flag captures. I hypothesise that the T1 Vaulter will learn to use its vaulting ability to vault the block tiles to create a shorter route for flag capture and become a specialist in flag captures.

Experiment 3 - Jailbreak. This experiment tests the Miner agent class. Specifically, it tests whether a Miner agent can learn to remove blocks from the environment in order to free a teammate. T1 consists of a Miner and a Scout, and T2 consists of two Scouts. One Scout from each team is trapped by destructible tiles, which can only be removed by a Miner agent. I hypothesise that the T1 Miner will learn to remove the destructible tiles surrounding the T1 Scout, gaining a numerical advantage over the opposing team and creating more flag captures.

Experiment 4 - One Way Out. This experiment examines the Miner agent class again. In contrast to the previous experiment, this scenario is designed to test if the Miner agent can learn to mine and place blocks in order trap the opposing team. T1 consists of a Miner and a Scout, and T2 team consists of two Scouts. The spawning zones for each team have a narrow opening from which agents can enter and exit. Several destructible tiles are placed throughout the map, including a row of tiles between the two flags, forcing a longer route for capture. I hypothesise that the T1 Miner will learn to mine tiles and block the T2 spawning zone, preventing T2 agents from exiting their spawning zone when tagged and effectively nullifying their threat in the game.

Starting game states for experiments 1 - 4 are provided in figure 4.1.

4.2.2 Results

Experiment 1 - The Split. Overall results indicate that T1-G learns to defend its flag whilst T1-S assumes a more attacking role, consistent with my hypothesis. Complex behaviours in how T2 agents adapt to this specialism are also observed.

Examining agent results at training generation 50 (see E.2), T1-G agent spends more steps in its defensive zone (231 vs 44)*** compared with T1-S. In contrast, T1-S spends more time in the attacking zone (66 vs 3)*** and has more flag captures (1.75 vs 0.05)** in comparison with

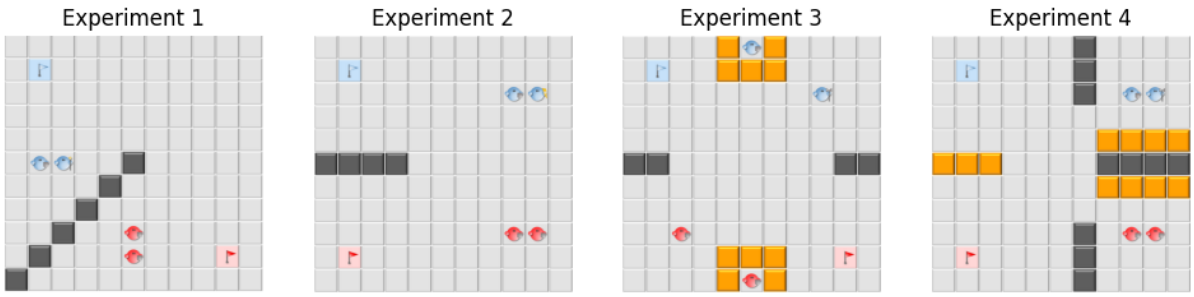

Figure 4.1: Experiment Starting Positions

T1-G. Across tagging metrics, the T1-S makes more tags (79 vs 28), however T1-G makes more tags which result in respawns (6.35 vs 1.72).

Agent level results for T2 are not reported for this set of experiments, as both T2-S follow the same policy as a result of parameter sharing (see 3.3.2), and there are no material differences in behaviours for each agent.

At the team level (see E.1) T1 spends more time in their defensive zone (275 vs 135 steps)***. It is also observed that T2 spends more time next to a teammate (656 vs 561 steps)***.

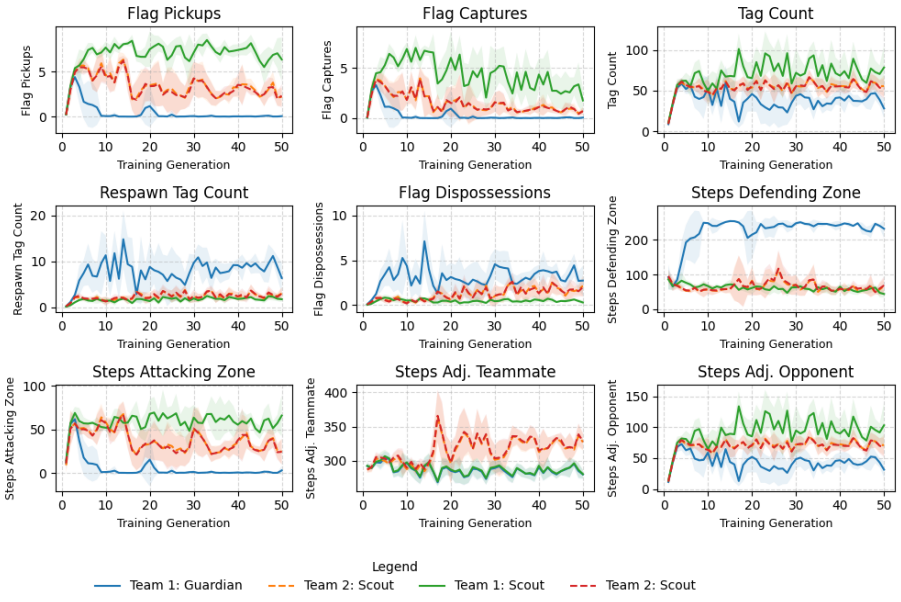

Figure 4.2: Experiment 1 Agent Metrics

From observing several rounds of gameplay, both T2-S coordinate to team-up on the T1-S. They effectively deal double tagging points by moving in tandem against T1-S. This behaviour is reflected in the steps adjacent to teammate metric.

These findings are reinforced by analysis of agent visitation maps in figure 4.3. This plot illustrates the movement of agents averaged over 10 matches, collected at the 10^{th}, 30^{th} and 50^{th} training generations. Darker shades indicate more frequent visits to a grid cell. T1-G spends most of its time in the upper left quadrant of the grid, near its flag throughout training. T1-S learns a direct path between flags. In contrast, both T2-S initially learn a direct route between flags, however by the 50^{th} training generation they have learnt a coordinated attack pattern, where one Scout takes a route at the top of the board while the other targets the center at the same time. In this

way, The Guardian must choose which opponent to pursue, giving the free opponent a better chance of survival and flag capture.

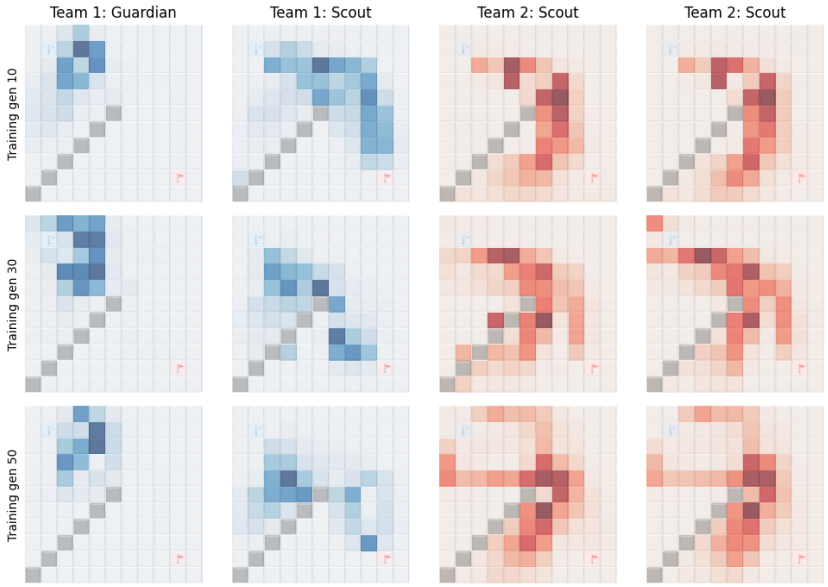

Figure 4.3: Experiment 1 Agent Visitation Maps

Experiment 2 - The Fence. Overall results indicate that T1-V learns to traverse the barrier in the middle of the grid. T1-V does not become a flag capturing specialist as hypothesised as T2 learn to mitigate the superior mobility of T1-V over the course of training.

From figure 4.4, T1-V dominates flag pickups, captures and steps in the attacking zone in early training generations. However, by training generation 50 (see E.4), there are no statistically significant differences in flag metrics or positional metrics between T1-S and T1-V. T1-S engages opponents more, with more tags (97 vs 68)*** and more tags leading to respawns (5.15 vs 3.09)**. T1-S is more inclined to engage opponents whilst T1-V is more evasive. This makes intuitive sense as the Vaulter incurs HP penalties from using its vaulting ability, and is more vulnerable to tags. This emergence and disappearance of specialisation is likely driven by changes in T2 strategies in response to early dominance of T1.

At the team level (see E.3), T2 have more tags leading to respawns (15.82 vs 8.24)*** and more flag dispossessions (5.85 vs 2.41)***. Observing positional metrics, T2 spend more time in their defensive zone (237 vs 157 steps)**.

From the agent visitation map in figure 4.5, T1-S initially takes on a more defensive role in early training before learning a more direct route between flags. T1-V learns to jump over the fence in the middle of the grid by training generation 30, however by training generation 50 it learns to hug the left wall when near the opponents flag, and also learns a much longer route between flags in response to improved defensive strategies learnt by T2. In early training generations, T2 learn extremely defensive strategies, almost never moving outside of their defensive quadrant. By training generation 30, the T2 Scouts have learnt to strike an effective balance between attacking and defending, taking a fairly direct route between flags, but also spending ample time defending key squares around their flag to thwart attacks. These observations support the earlier hypothesis that the emergence and disappearance of specialisms in Team 1 are driven by strategic responses from Team 2.

Experiment 3 - Jailbreak. Overall results indicate that T1-M learns to mine blocks to effectively free its T1-S teammate, creating a numerical advantage over T2. This results in total

Figure 4.4: Experiment 2 Agent Metrics

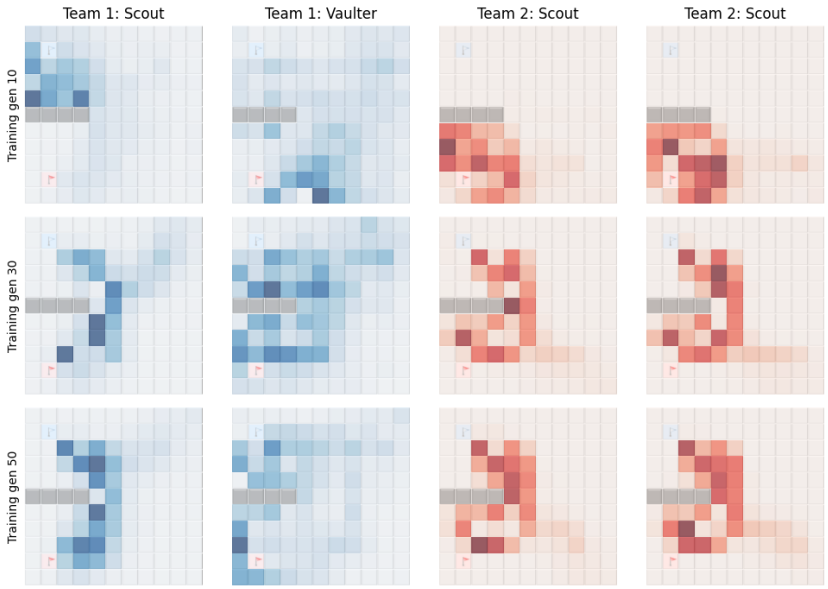

Figure 4.5: Experiment 2 Agent Visitation Maps

game dominance and is consistent with my hypothesis. Once freed, T1-S and T1-M share similar behaviours.

Observing figure 4.6, T2-S dominates flag pickups and captures until later training generations when the T1-M has learned to free its teammate and the T1-S has subsequently become proficient at the game. By training generation 50 (see E.6), behaviours of the T1 agents are strikingly similar, except for flag metrics where T1-M makes more flag pickups (8.01 vs 7.51)***, but T1-S makes more flag captures. This might be driven by differences in HP for the two agent classes.

At the team level, T1 flag pickups and captures consistently increase over the course of training, whilst these metrics are slightly decreasing for T2. These differences are significant at the 1% level at training generation 50 (see E.5), and are driven by the Miner agent learning to free its Scout teammate, and the Scout subsequently learning to play the game.

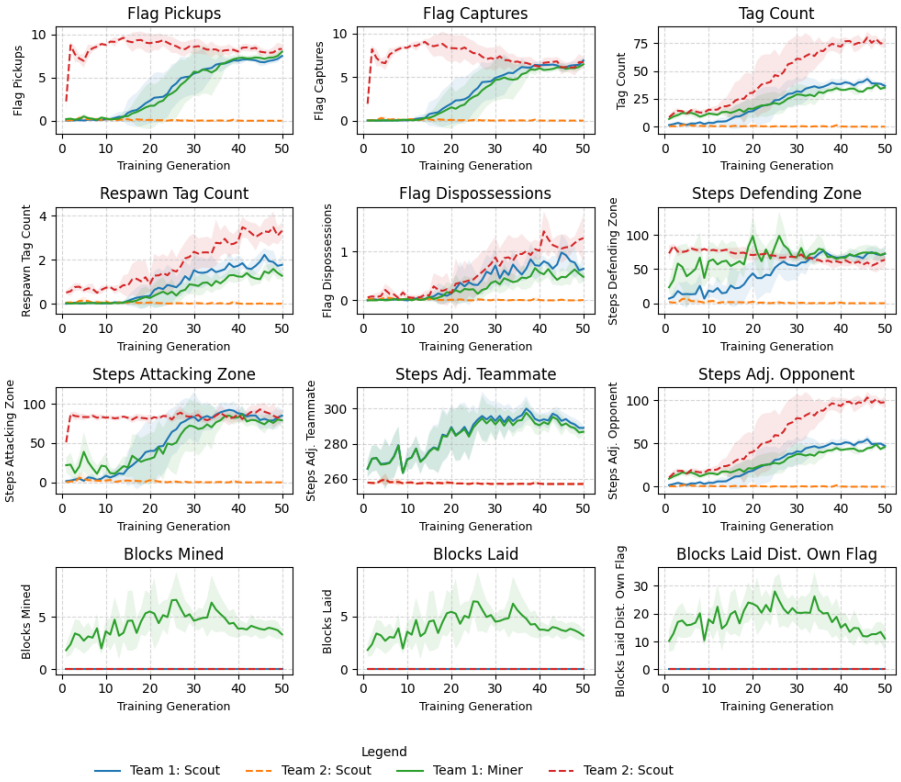

Figure 4.6: Experiment 3 Agent Metrics

Agent level findings are supported by observations from the agent visitation maps in figure 4.7. From generation 30, the T1-M has learnt to reliably free its teammate and once freed, the T1-M and T1-S have remarkably similar routes of travel. T1-M actively places tiles to obstruct T2-S when it is in close proximity. T2-S quickly establishes a direct route between flags in early training generations, later following a more variable path, required to avoid tiles placed by T2-M. One peculiar observation is that in early and later training generations, T1-M occasionally frees T2-S. This is probably due to stochasticity in the policy causing T1-M to occasionally move into the destructible tiles encompassing T2-S.

Experiment 4 - One Way Out. Overall results indicate that T1-M learns to mine and place blocks to trap T2 agents, consistent with my hypothesis. Adaptations made by T2 to mitigate the T1 strategy are observed in later training generations.

Observing figure 4.8, T2 dominates flag pickups and captures until about training generation 20. At training generation 50, there are no statistically different behaviours between T1 agents, with the exception of block interaction metrics that are exclusive to T1-M. This mirrors findings from experiment 3, where T1-M and T1-S demonstrated similar behaviours.

Team level results are consistent with agent level findings. T2 dominates early matches, however from training generation 20 onward T1 consistently produces more flag pickups and captures. At training generation 50 (see E.7), T1 have more flag captures (23 vs 3)***, spend more time in the attacking zone (218 vs 86 steps)*** and more time in the defensive zone (179 vs 61 steps)***. These results imply decreased mobility of T2 agents.

From the agent visitation maps in figure 4.9, in early training generations T1-M learns to remove a tile in the far left side of the board, creating a shortcut for flag capture. Both T2-S agents adapt to this behaviour and learn to target this area of the board. By training generation 30, T1-M has learnt to mine a block and seal the entrance to the T2 spawning area. Both T2-S

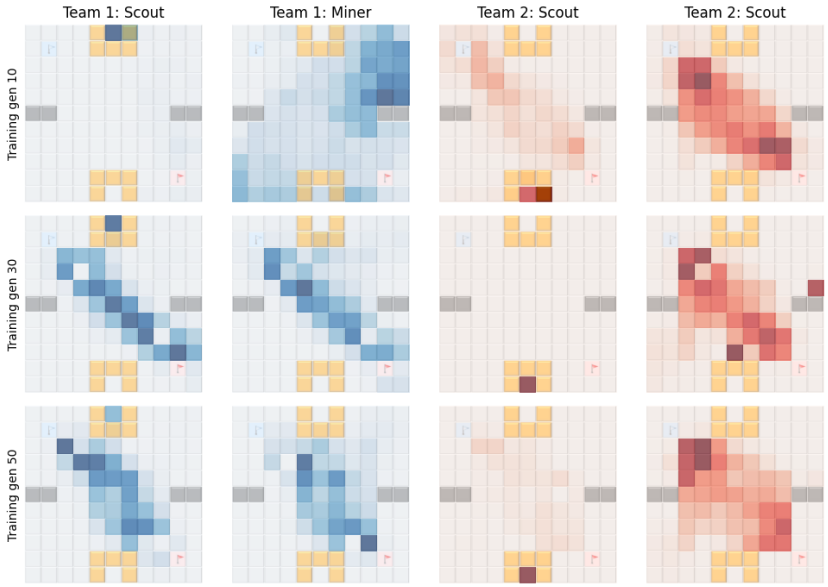

Figure 4.7: Experiment 3 Agent Visitation Maps

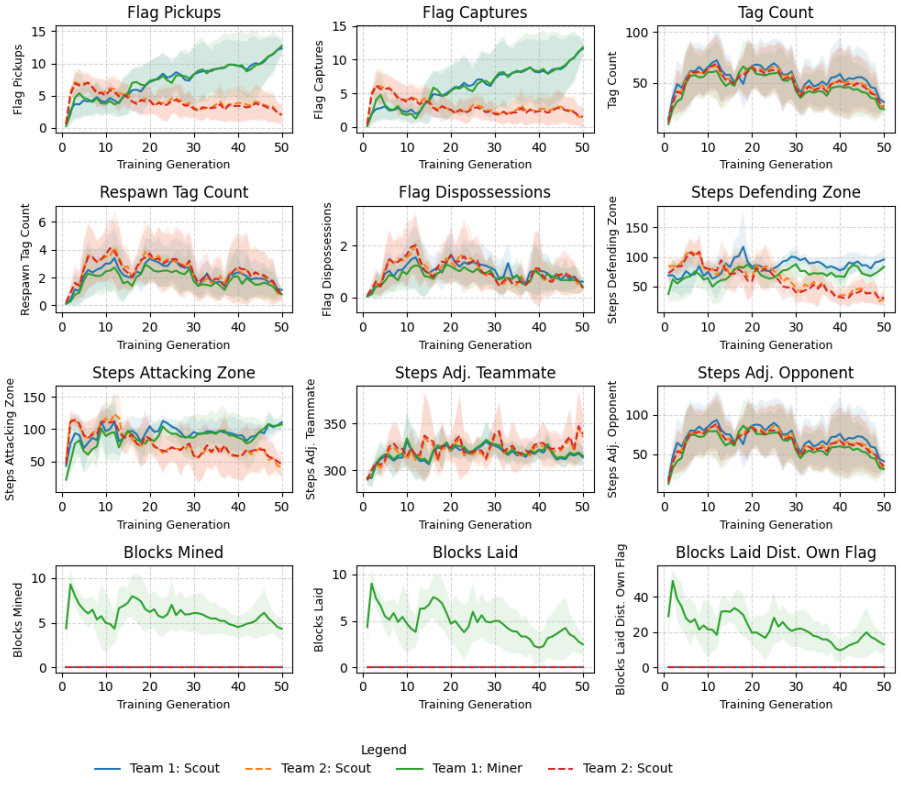

Figure 4.8: Experiment 4 Agent Metrics

agents have increased frequency in their spawning zone on the visitation maps, indicating they have been trapped. By training generation 50, the T1-M learns to remove a tile from the central barrier, expediting flag captures. T2 adapt once again and try and avoid being tagged by sticking to the peripheries of the main area. Although this mitigates being trapped early, ultimately it is not sufficient to win matches, as evidenced by the team flag capture metrics in table E.7.

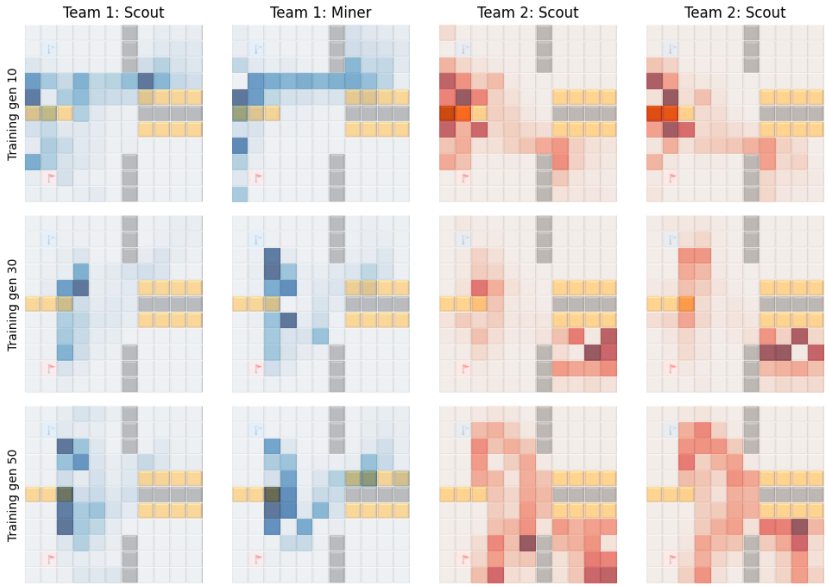

Figure 4.9: Experiment 4 Agent Visitation Maps

4.3 Mixed Agent Experiments

4.3.1 Experiment Descriptions and Hypotheses

Experiment 5 - Keyhole. This experiment tests mixed classes in a 2v2 setting, with a focus on how a Miner and Vaulter on opposing teams impacts the development of specialised behaviours. T1 consists of a Miner and a Scout, and T2 consists of a Vaulter and a Scout. As with experiment 4, the spawning zones for each team have a narrow opening from which agents can enter and exit. Several destructible tiles are placed throughout the map. A row of block tiles separates the two flags. These block tiles have a gap in them which can be utilised by the Vaulter for a shorter route between flags. I hypothesise that T1-M will learn to mine destructible tiles and place these in the entrance of T2, blocking agents from escaping upon respawn. I expect T2-V to learn to exploit the gap in block tiles in the centre of the map and become prolific in flag captures.

Experiment 6 - Skittles. Similar to experiment 5, this experiment examines a Miner and Vaulter on opposing teams in environment with more destructible tiles. T1 consists of a Miner and a Scout, and T2 consists of a Vaulter and a Scout. As with the previous experiment, the spawning zones for each team have a narrow opening from which agents enter and exit. A large number of destructible tiles are placed throughout the map, providing both obstacles for movement and giving greater creative freedom for the Miner agent to transform the environment. I hypothesise that the T1-M agent will learn to mine and place tiles to impede movements of T2. I expect T2-V to learn to vault obstacles and dominate flag captures.

Starting game states for experiments 5 and 6 are provided in figure 4.1.

4.3.2 Results

Experiment 5 - Keyhole. Overall results indicate that T1-M learns to mine and place blocks to trap T2 agents. T2-V is able to navigate this trap and also makes use of the shortcut between flags, but falls short of becoming a flag capture specialist. Further adaptations are also noted, where T1-V lays blocks to obstruct T2 agents, and also preservation behaviours learnt by T2-S.

Observing figure 4.11, both teams are evenly matched until training generation 20, when T1 begins to dominate. By training generation 50 (see E.9), T1 account for more flag pickups (17 vs

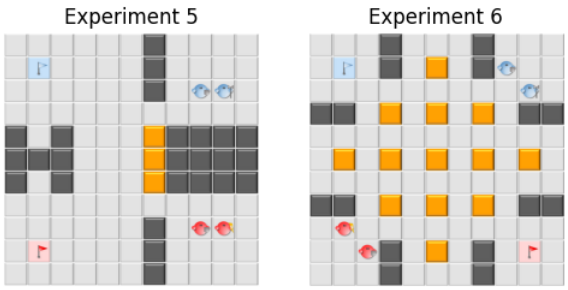

Figure 4.10: Experiment Starting Positions

9)**, flag captures (15 vs 5)**, tags resulting in respawns (12 vs 4)**, and steps adjacent to a teammate (606 vs 570)***.

At the agent level, no statistically significant differences are observed between agents in either team. This is driven by higher variances of metric values in this experiment.

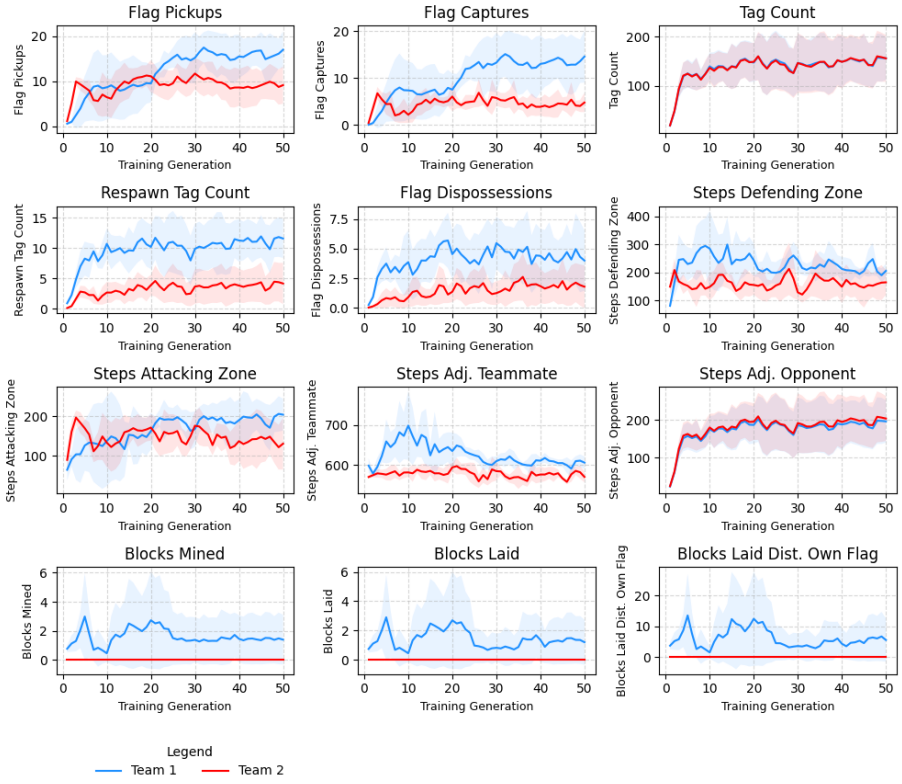

Figure 4.11: Experiment 5 Team Metrics

From observation of gameplay, agents make similar adaptations as observed in experiment 4. As depicted in figure 4.12, by training generation 10 T1 have learnt to defend against the mobility of the T2-V, staying in close proximity to their flag. In contrast, T2 agents have learned routes for flag capture. By training generation 30, T1 agents have learned paths for flag capture and T1-M has learned how to block the entrance of to the T2 spawning zone, effectively trapping T2-S. By training generation 50, the T2 Scout has adapted by sticking to the periphery of the map to avoid being tagged and trapped. The T2-V now spends more time in its spawning zone as T1-M learns to place tiles which obstruct it from escaping. The T1-S spends more time near its flag to ward off attacks.

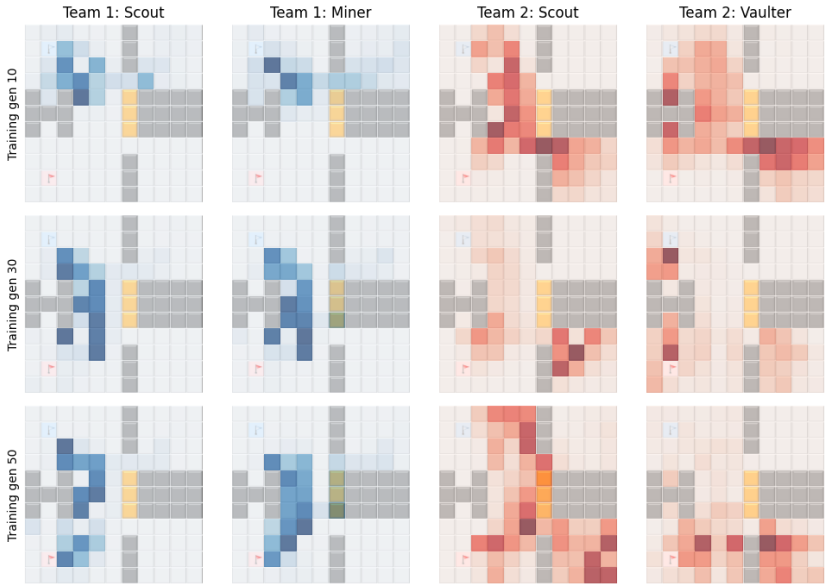

Figure 4.12: Experiment 5: Agent Visitation Maps

Experiment 6 - Skittles. Overall results show that T1-M learns to mine and place tiles to impede the movements T2 agents, and create a direct route for flag capture. T2-V becomes a specialist in flag capturing. Both T2 agents are able to navigate these obstacles, however they must travel further for flag capture. These results are consistent with my hypothesis.

Observing figure 4.13, differences across flag and tagging metrics for T2 agents are observed. By training generation 50 (see E.13, E.14), T2-V has more flag pickups (7.03 vs 2.63)*** and captures (4.54 vs 1.97)***. T2-S has consistently higher tag counts leading to respawns (0.91 vs 0.12)*** and flag dispossessions (0.45 vs 0.12)***. T1 agents are similar across most metrics, with the exception of tagging metrics, where T1-S makes more tags (50 vs 43)*** and tags resulting in respawn (4 vs 3)**. As with previous experiments, T1-S and T1-M display similar behaviours across most metrics.

At the team level, T2 account for more flag pickups and captures in early training generations, however this trend is reversed from about generation 10 onward. At this stage, T1-M has learnt to effectively mine and place blocks to create obstacles for T2. By training generation 50 (see E.12), T1 dominates gameplay, with more flag captures (12 vs 7)***.

Observing gameplay, the T1-M learns to obstruct the T2 by creating a diagonal wall, creating a much longer route for T2-S for flag capture. In creating this structure, T1-M also clears the path between the two flags, increasing the T1 capture velocity. This contrasts from previous experiments, where the Miner learns to block the entrance to the spawning zone directly. From the agent visitation map in figure 4.14, it can be seen that T1 agents learn a diagonal route for flag capture over the course of training. From training generation 30 onwards, T2-S gets trapped in its spawning zone quadrant by T1-M. Additionally, T2-V learns to traverse the perimeter of the map, navigating obstacles placed by T1-M and also minimising contact with T1.

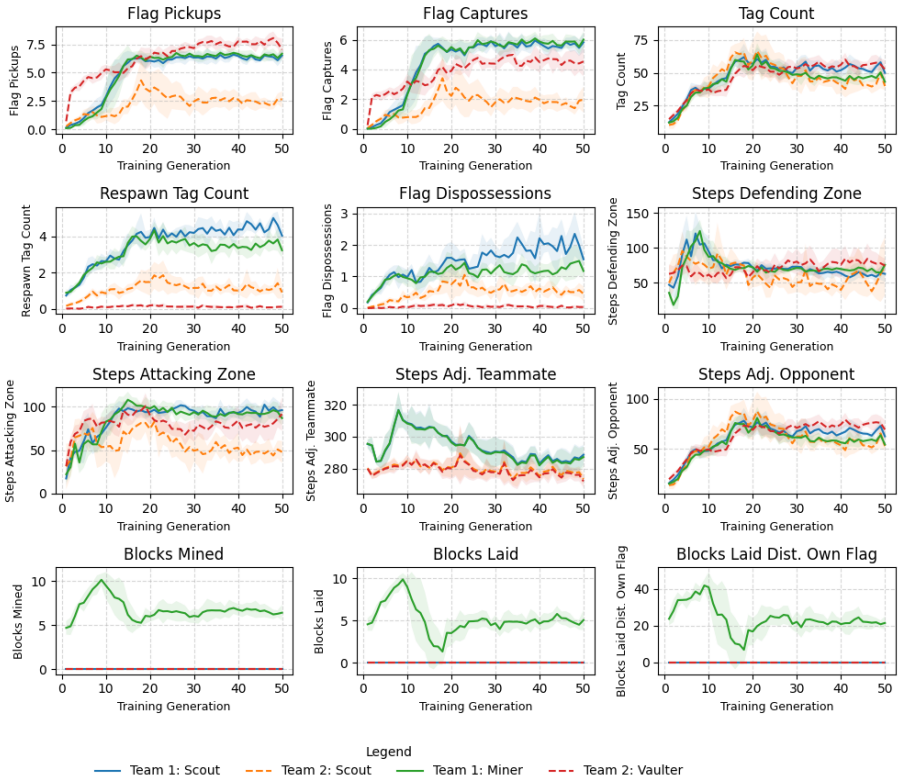

Figure 4.13: Experiment 6 Agent Metrics

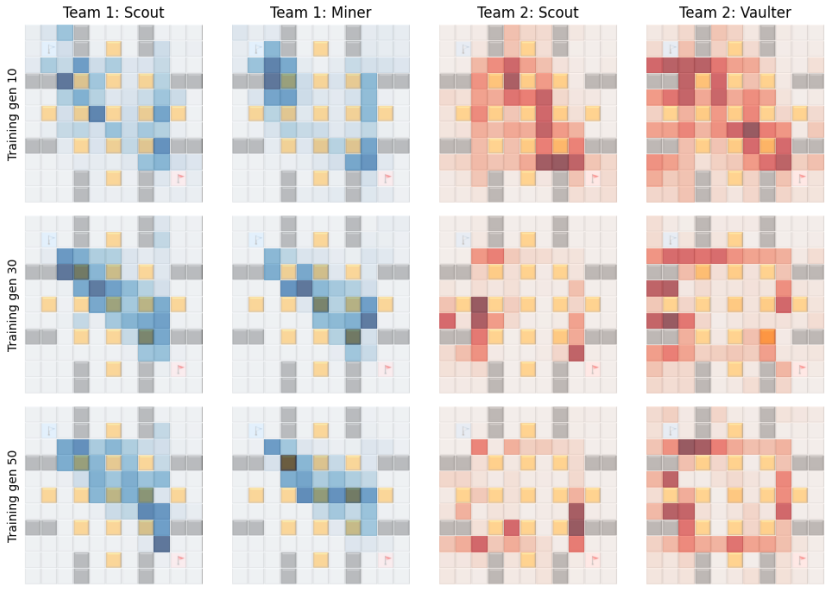

Figure 4.14: Experiment 6: Agent Visitation Maps

4.4 Complex Experiments

4.4.1 Experiment Descriptions and Hypotheses

Experiment 7 - The Wall. This experiment tests mixed teams in a 3v3 setting on a larger grid size of 13x13. This experiment examines whether specialised behaviours observed in earlier 2v2 experiments persist in a 3v3 setting. T1 consists of a Miner, Scout and Guardian. T2 consists of

a Miner, Scout and Vaulter. The defining feature of this map is that a large wall of destructible tiles separates the two sides. Both teams have a Miner agent to modify the environment. I hypothesise that the T1-V and T2-V will lead in flag captures, exploiting the single row of block tiles near the spawning points. I expect the T1-G will take up a defensive role to try and counter these attempts. As in other experiments, T1-M and T1-M are expected to create new routes and obstacles by mining tiles.

Experiment 8 - Gridlocked. Similar to experiment 7, this experiment tests for specialised behaviours using mixed teams on a 13x13 map. T1 team consists of a Scout, Miner and Guardian, and T2 consists of a Scout, Vaulter and Guardian. The spawning zones for each team have a narrow opening from which agents can enter and exit. A large number of destructible tiles are placed throughout the central area of the map, acting initially as obstacles but also providing opportunity for the Miner agent to transform the environment. I hypothesise that the T1-M will learn to mine destructible tiles and create obstacles to impede movements of T2. I expect T2-V will learn to navigate these obstacles and become a flag capturing specialist. T1-G and T2-G are expected to take up defensive duties and spend more time in proximity of their flags. T1-S and T2-S are expected to demonstrate a balance between offensive and defensive roles.

Experiment 9 - Arena. This is the final and most complex environment in this study. This experiment tests mixed classes in a 4v4 setting on a larger grid size of 15x15. Design elements from all previous experiments are combined. Teams are symmetric, consisting of a Scout, Guardian, Vaulter and Miner. Teams spawn in enclosed and team flags are positioned in the top and bottom central areas of the map. A large number of destructible tiles are scattered throughout the map. Training generations from are increased 50 to 100 for this experiment due to the increased complexity of the environment resulting from a larger map and more agents. I hypothesise that the Vaulter agent will learn to navigate obstacles and become a flag capturing specialist. I expect the Guardian agent will take up a defensive role near its flag and the Miner agent will learn to create obstacles to thwart opponents. As before, the Scout is expected to demonstrate a balance between offensive and defensive roles.

Starting positions for each experiment are provided in figure 4.15.

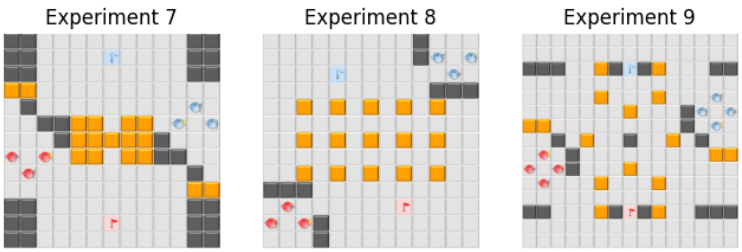

Figure 4.15: Experiment Starting Positions

4.4.2 Results

Experiment 7 - The Wall. Overall results are mixed with respect to my hypotheses as some agents learning specialised behaviours, while others do not. For example, T1-V learns to vault obstacles to make flag captures, whilst T1-M does not make use of its block mining ability. T2-V fails to specialise in capturing the flag due to the defensive abilities of T1-G. T1-G and T2-G learn to defend their flags. On aggregate, T1 has learnt more offensive behaviours, whilst T2 is more defensive.

From figure 4.16, T1-V has more flag captures, tags, steps in the attacking zone and steps adjacent than other agents over the course of training. At training generation 50 (see E.16, E.17),

T1-V has more flag pickups (16 vs 1 vs 1)** and more tags (53 vs 2 vs 3)** than its teammates. For T2, T2-M and T2-V demonstrate generally similar behaviours. T2-V and T2-S are more distinct, with T2-V making more flag captures (0.19 vs 0.01)** and taking more steps in the attacking zone (7.42 vs 1.69)**. T1-M makes little use of its mining abilities, mining 0.57 blocks and laying 0.55 blocks on average, whilst T2-M uses its abilities liberally, mining 10.43 and laying 10.2 blocks on average. One explanation for this might be that due to the defensive superiority of T1-G, evolutionary pressure is not exerted on T1-M to use its abilities.

At training generation 50 (see E.15), T1 has more flag pickups (7.65 vs 0.75)** and more steps in the attacking zone (77.58 vs 26.95)***. In contrast, T2 has more tags leading to respawns (4.0 vs 0.81)**, flag dispossessions (2.53 vs 0.36)** and more steps in the defensive zone (397 vs 179)**.

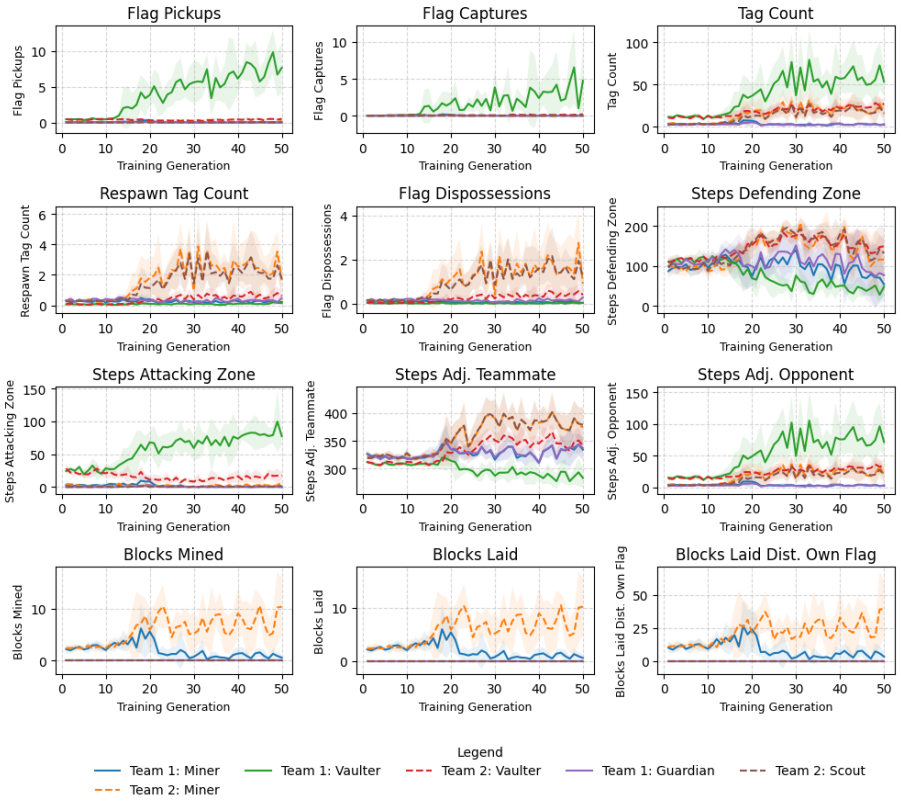

Figure 4.16: Experiment 7 Agent Metrics

From observation of gameplay and agent visitation maps presented in figure 4.17, T1-V initially launches attacks around the periphery of the map, but later also learns a direct route. T1-M rarely moves into the opposition half and T1-G stays near its flag as expected. T2 agents initially concentrate on the left side of the map, however in later generations T2 attack and defend centrally. The T2 miner learns to remove blocks from the wall, and actively places obstacles to impede T1 agents.

Experiment 8 - Gridlocked. Results from this experiment show that agent classes learn roles consistent with my hypothesis. T1-M learns to create defensive structures, T1-G and T2-G defend their flags and T2-V captures the flag more than other agents.

At training generation 50 (see E.19, E.20), T1-S and TS-M behaviours are not too dissimilar, whilst significant differences exist between T1-S and T1-G, and T1-M and T1-G. T1-S and T1-M both spend more time in the attacking zone compared with TG-G (39 vs 12)** and (22 vs 12)**. T1-G spends more time in the defensive zone than T1-S (207 vs 126)** and T1-G (207 vs 141)*. For T2, it is observed that T2-S engages in more tags than T2-V (43 vs 32)**, whilst T2-V has

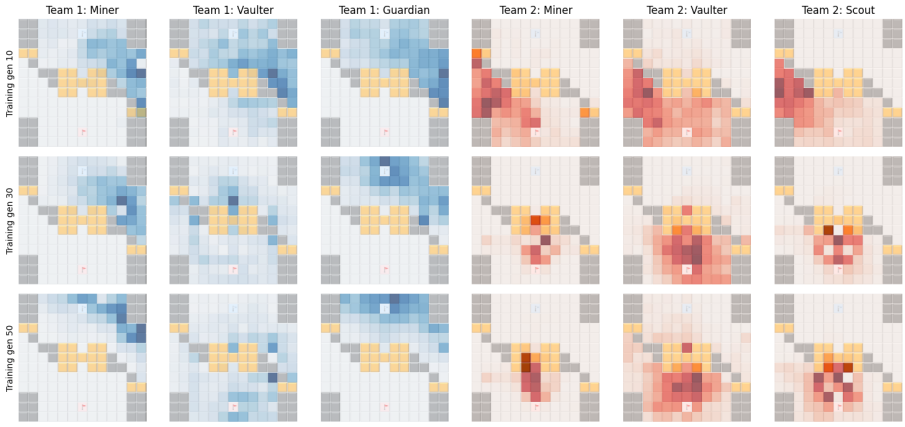

Figure 4.17: Experiment 7: Agent Visitation Maps

more flag captures than T2-S (1.63 vs 0.96)* and T2-G (1.63 vs 0.07)***. T2-V also spends more time in the attacking zone than T2-S (66 vs 33)** and T2-G (66 vs 3.55)***. T2-G spends more time in the defensive zone than both T1-S (221 vs 113)*** and T2-V (221 vs 68)***. These results align with expectations for each agent class.

Observing gameplay, T1-M learns to move blocks to create defensive structures around its own flag, which is then guarded by T1-G. The structures effectively force T2 agents (with the exception of T2-V) to attack through the middle of the map.

Experiment 9 - Arena. Results from this experiment show that specialised behaviours are learnt by each agent class, consistent with my hypothesis. This experiment examines symmetrical teams using pure self-play. As such, comparisons between T1 and T2 as not made as in previous experiments.

From figure 4.18, it is observed that T-V has more flag pickups, flag captures and steps in the attacking zone than the other agent classes. At training generation 50 (see E.21, E.22), these differences are significant at the 5% level. T-G has more tags that result in respawns, flag dispossessions and steps in the defensive zone compared with other agent classes. Again these differences are significant at the 5% level. T-M agent learns to mine and lay blocks, but is otherwise similar to T-S, however T-S attempts takes more steps in the attacking zone (9.19 vs 4.97)*** and makes more flag pickups (0.5 vs 0.18)**.

Observing figure E.27, T-G stays positioned near its flag throughout training. T-V initially launches attacks centrally, but has learned more diverse routes by training generation 50. T-M positions itself centrally over training. From gameplay observations, it is observed T-M learns to create defensive structures and obstacles to impede the opposing team. T-S behaves similarly to T-M, however it is slightly more aggressive and is more inclined to attempt flag captures.

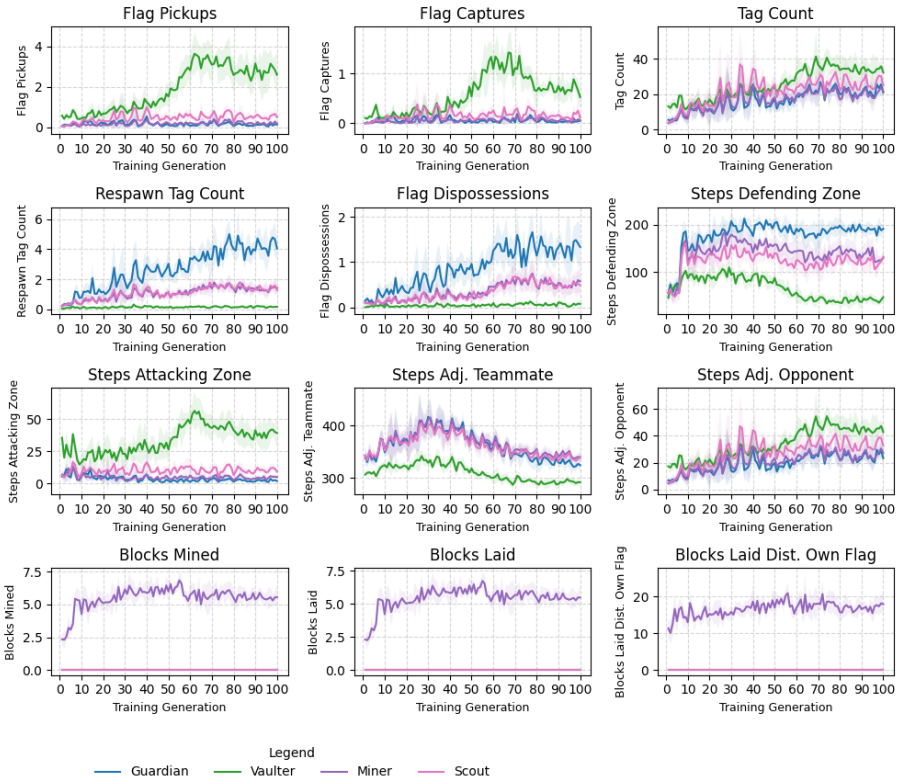

Figure 4.18: Experiment 9 Agent Metrics

4.5 Results Summary

Overall, the experiment results generally align with the stated hypotheses. Across experiments the Guardian learns to defend its flag, the Vaulter becomes adept at flag capturing, the Miner creates obstructions and defensive structures, and the Scout balances offensive and defensive behaviours.

The following exceptions were observed where agents did not learn expected specialisms:

- In experiment 2, T2-V fails to specialise in flag captures, due to defensive adaptations made by T2.
- In experiment 5, T2-V fails to specialise in flag captures, due to defensive adaptations made by T1 and obstacles created by T1-M.
- In experiment 6, T1-G causes T1 to be overpowered. As such, competitive pressures are not sufficient for T1-M to make use of its block mining abilities. Also T2-V is unable to specialise in flag captures as it cannot get near to the T1 flag due to the superior defensive capabilities of T1-G.

Chapter 5

Discussion and Conclusions

In this chapter I review the key findings of this study in the context of the research objectives, and draw comparisons with related research. I also highlight limitations this study, propose directions for future research and provide final conclusions.

5.1 Interpretation of Results in the Context of the Research Questions and Objectives

The objectives of this study have been threefold:

1. Develop a bespoke, 2D multi-agent capture the flag environment.
2. Apply reinforcement learning methodologies to train agents to play capture the flag.
3. Test for the emergence of specialisation over the course of training.

Develop a bespoke, 2D multi-agent capture the flag environment
I have developed a 2D multi-agent capture the flag environment with four distinct agent types. This environment uses common Python libraries and runs on modest hardware. I have described in detail the mechanics of this environment, and formulated experiments and hypotheses to test for specialised behaviours. Despite the simplicity of this environment, I have observed interesting and complex behaviours through the experiments and analysis.

Apply reinforcement learning methodologies to train agents to play capture the flag
I have implemented Proximal Policy Optimisation with self-play and successfully trained agents to play capture the flag within the environment. I have conducted hyperparameter tuning, implemented action masking and parameter sharing to facilitate learning, and implemented potentially novel mechanics in the form of a shared memory buffer.

Evaluate the emergence of specialised behaviours over the course of training
I have developed twelve evaluation metrics and utilised quantitative and qualitative methods to analyse the behaviours learnt by agents during training. Metrics cover flag capture, tagging, agent positions and environment interactions. The analytical methodology uses both statistical hypothesis testing as well as qualitative observations in order to draw inferences. I have designed and implemented nine experiments which test for specialisation across a range of scenarios of varying complexity.

In answer to the main research question; **this study finds strong evidence that heterogeneous agents learn specialised roles when trained using RL and self-play in a CTF setting**.

From the experiment results the Scout exhibits a balance between offensive and defensive behaviours. The Guardian learns to assume a defensive role, reliably guarding the area near its flag. The Vaulter uses its enhanced mobility to traverse the environment for flag captures whilst avoiding opponents. The Miner learns to transform the environment, displaying a range of complex behaviours, from freeing teammates, to trapping opponents, creating shortcuts and building defensive structures. The Miner and Scout agents often demonstrate similar behaviours, as outside of mining ability they share similar attributes. These findings generally hold as environment complexity increases through the addition of more agents and larger map sizes.

Furthermore, over the course of training agents adapt their behaviour to counter opponent strategies, leading to distinct phases of gameplay and changes in specialisms. For example, in experiment 4, the T1-M initially creates a shortcut between flags, which is then targeted by T2. In response, the T1-M learns to block the entrance to the T2 spawning zone. T2 agents then learn to avoid T1 agents to mitigate chances of being tagged and subsequently trapped. These findings of policy adaptation are consistent with those of Baker et al. (2020), who observe behavioural adaptations in a hide and seek setting. In some cases, these adaptations cause learned specialisations to be voided, such as experiment 2, where improved coordination amongst T2-S agents erodes the mobility advantage of the T1-V, causing the T1-S and T1-V behaviours to converge as training progresses. These findings suggest that reinforcement learning and self-play can not only be used to learn specialisation in heterogeneous agents, but to also find and exploit vulnerabilities in adversaries with heterogeneous agents. I also note that agent class imbalances can prevent agents from learning specialisation. In experiment 6, T1-G causes T1 to dominate games. T1-M does not learn to use its block mining ability as there is insufficient competitive pressure and T2-V does not succeed in capturing the T1 flag as the defensive capabilities of T1 are too strong.

In contrast to previous MARL studies on specialisation (Murciano, R. Millán, and Zamora, 1997; Gasparrini, Solé, and Sánchez-Fibla, 2019), I have examined the emergence of specialisation in teams of heterogeneous agents. Additionally, I illustrate how specialisms evolve and adapt over generations of self-play. In contrast to Wang et al. (2020), I analyse agent specialisation only in terms of different agent classes, in the absence of other factors that might conflate results. Unlike previous 2D CTF RL studies (Hefny et al., 2008; Ivanovic et al., 2014), I have demonstrated that agents can learn to proficiently play CTF using only sparse reward signals. I have built upon evaluation methods used by Baker et al. (2020) to test for emergence of behaviours using self-play through the addition of statistical hypothesis testing. The results from this study align with theories from evolution and economics. Given innate differences in agent attributes, evolutionary pressures force agents to learn behaviours which leverage their relative strengths and also exploit the weaknesses of opponents. Through task specialisation, agents are able to produce improved outcomes for the collective group. It is remarkable that reinforcement learning methods can be used to surface complex behaviours in the absence of any incentives beyond of playing the game as well as possible.

Beyond the environment of CTF, these findings imply that RL methods can be used to develop effective policies that leverage unique attributes of agents working in heterogeneous multi-agent teams. These methods can be applied to any domain that utilise teams of heterogeneous agents. System designers can use RL methods to develop multi-agent systems without having to hand-craft specialised behaviours or intricate reward signals. This is especially important where useful behaviours for solving a problem are not known a priori.

5.2 Limitations and Directions for Future Research

I identify several limitations of this study, and avenues for future research.

Full observability
This study assumes full observability of the environment for all agents. Each agent has access to the entire game grid and has complete knowledge of the positions of all other agents. In many real-life scenarios, this might not be a realistic assumption, where agents might need to work together but do not have line of sight or location data for the other agents. Partial observability could be implemented by reducing the field of vision to a fixed number of cells. Partial observability increases complexity and training time to achieve good results, but might better approximation of reality for some multi-agent settings.

Agent Analysis
I have examined agent specialisation using several curated experiments. One possible extension of this work could be to exhaustively trial all possible combinations of agent types up to a fixed number of agents. This would enable deeper analysis of agent behaviours and how agent roles vary with different agent combinations. This could be extended further by randomising map generation to understand specialised behaviours in a more general terms.

Learning methods
I have used self-play methods to train agents. Other methods, such as population-based training used by Baker et al. (2020) and league-training used by Vinyals et al. (2019) could yield superior results and more complex behaviours. I have implemented a version of league training, however I found training times to be excessive, and I achieved good results from using self-play. However, use of other learning methods could yield considerable improvements.

Scaling
I have considered team sizes of up to four agents. One future research direction would be to examine larger team sizes, maps, and action-sets to see how this impacts on learning and specialisation.

Compute Resource
I have conducted all of the experiments on a single M1 MacBook Pro, and all of experiments were run for less than 100 training generations. Whilst this has been sufficient to observe the emergence of specialisation and adaptations of strategy, utilisation of more powerful computing hardware would enable experiments to be run for more timesteps in less time. Run over many more timesteps, further interesting specialisms learnt by agents might be discovered, as well as strategical adaptations.

5.3 Conclusion

In this study I have examined how heterogeneous agents learn specialised roles in capture the flag using RL methods. Results show that heterogeneous agents learn specialised roles when trained using reinforcement learning and self-play, and these results hold as the number of agents and environment sizes are increased. Furthermore, this study has shown that agents adapt their behaviour in response to specialisms learnt by opponents. These results imply that RL can be used to train teams of heterogeneous agents to leverage the unique capabilities of different agent types without the need for hand-crafted behaviours or intricate reward design.

References

Smith, A., 1776. *An inquiry into the nature and causes of the wealth of nations* [Online]. McMaster University Archive for the History of Economic Thought. Available from: https://EconPapers.repec.org/RePEc:hay:hetboo:smith1776.

Ricardo, D., 1817. *On the principles of political economy, and taxation* [Online], Cambridge library collection - british and irish history, 19th century. Cambridge University Press. Available from: https://doi.org/10.1017/CBO9781107589421.

Lübeck, W., 1843. *Lehr- und Handbuch der deutschen Turnkunst* [Online]. Harnecker. Available from: https://books.google.co.uk/books?id=XMs3PwAACAAJ.

Baden-Powell, R., 1908. *Scouting for boys: the original 1908 edition* [Online], Dover books on sports and popular recreations. Dover Publications. Available from: https://books.google.co.uk/books?id=R2lJgMD9MRAC.

Durkheim, E., 1984. *The division of labour in society*, Contemporary social theory. Theoretical traditions in the social sciences. Basingstoke: Macmillan.

Littman, M.L., 1994. Markov games as a framework for multi-agent reinforcement learning. In: W.W. Cohen and H. Hirsh, eds. *Machine Learning Proceedings 1994* [Online]. San Francisco (CA): Morgan Kaufmann, pp.157–163. Available from: https://doi.org/10.1016/B978-1-55860-335-6.50027-1 [Accessed August 24, 2023].

Bourke, A.F.G. and Franks, N.R., 1996. *Social Evolution in Ants* [Online]. Princeton: Princeton University Press. Available from: https://doi.org/doi:10.1515/9780691206899 [Accessed August 27, 2023].

Murciano, A., R. Millán, J. del, and Zamora, J., 1997. Specialization in multi-agent systems through learning. *Biological cybernetics* [Online], 76(5), pp.375–382. Available from: https://doi.org/10.1007/s004220050351 [Accessed August 20, 2023].

Lee, H. and Lane, M., 2007. *The Republic* [Online], Penguin classics. Penguin Publishing Group. Available from: https://books.google.co.uk/books?id=0f5MQif-HCcC.

Hefny, A., Hatem, A., Shalaby, M., and Atiya, A., 2008. Cerberus: Applying Supervised and Reinforcement Learning Techniques to Capture the Flag Games. *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment* [Online], 4(1). Number: 1, pp.179–184. Available from: https://doi.org/10.1609/aiide.v4i1.18694 [Accessed August 21, 2023].

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., 2013. *Playing Atari with Deep Reinforcement Learning* [Online]. arXiv:1312.5602 [cs]. arXiv. Available from: http://arxiv.org/abs/1312.5602 [Accessed February 2, 2023].

Ivanovic, J., Zambetta, F., Li, X., and Rivera-Villicana, J., 2014. Reinforcement learning to control a commander for capture the flag. *2014 IEEE Conference on Computational Intelligence and Games* [Online]. ISSN: 2325-4289, pp.1–8. Available from: https://doi.org/10.1109/CIG.2014.6932880.

Gupta, J.K., Egorov, M., and Kochenderfer, M., 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In: G. Sukthankar and J.A. Rodriguez-Aguilar, eds. *Autonomous Agents and Multiagent Systems* [Online], Lecture Notes in Computer Science. Cham: Springer International Publishing, pp.66–83. Available from: https://doi.org/10.1007/978-3-319-71682-4_5.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., 2017. *Proximal Policy Optimization Algorithms* [Online]. arXiv:1707.06347 [cs]. arXiv. Available from: http://arxiv.org/abs/1707.06347 [Accessed August 20, 2023].

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G. van den, Graepel, T., and Hassabis, D., 2017. Mastering the game of Go without human knowledge. *Nature* [Online], 550(7676). Number: 7676 Publisher: Nature Publishing Group, pp.354–359. Available from: https://doi.org/10.1038/nature24270 [Accessed July 29, 2022].

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I., 2018. *Emergent Complexity via Multi-Agent Competition* [Online]. arXiv:1710.03748 [cs]. arXiv. Available from: http://arxiv.org/abs/1710.03748 [Accessed July 29, 2022].

Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: an introduction* [Online]. Second. The MIT Press. Available from: http://incompleteideas.net/book/the-book-2nd.html.

Gasparrini, M.J., Solé, R., and Sánchez-Fibla, M., 2019. *Individual specialization in multi-task environments with multiagent reinforcement learners* [Online]. arXiv:1912.12671 [cs]. arXiv. Available from: http://arxiv.org/abs/1912.12671 [Accessed August 15, 2022].

Hernandez-Leal, P., Kartal, B., and Taylor, M.E., 2019. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Autonomous agents and multi-agent systems* [Online], 33(6). arXiv:1810.05587 [cs], pp.750–797. Available from: https://doi.org/10.1007/s10458-019-09421-1 [Accessed January 14, 2023].

Jaderberg, M., Czarnecki, W.M., Dunning, I., Marris, L., Lever, G., Castaneda, A.G., Beattie, C., Rabinowitz, N.C., Morcos, A.S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J.Z., Silver, D., Hassabis, D., Kavukcuoglu, K., and Graepel, T., 2019. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *Science* [Online], 364(6443). arXiv:1807.01281 [cs, stat], pp.859–865. Available from: https://doi.org/10.1126/science.aau6249 [Accessed July 30, 2022].

Leibo, J.Z., Hughes, E., Lanctot, M., and Graepel, T., 2019. *Autocurricula and the Emergence of Innovation from Social Interaction: A Manifesto for Multi-Agent Intelligence Research* [Online]. arXiv:1903.00742 [cs, q-bio]. arXiv. Available from: https://doi.org/10.48550/arXiv.1903.00742 [Accessed August 9, 2022].

Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., and Graepel, T., 2019. *Emergent Coordination Through Competition* [Online]. arXiv:1902.07151 [cs]. arXiv. Available from: https://doi.org/10.48550/arXiv.1902.07151 [Accessed August 1, 2022].

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H.P.d.O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S., 2019. *Dota 2 with Large Scale Deep Reinforcement Learning* [Online]. arXiv:1912.06680 [cs, stat]. arXiv. Available from: `https://doi.org/10.48550/arXiv.1912.06680` [Accessed August 9, 2022].

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* [Online], 575(7782). Number: 7782 Publisher: Nature Publishing Group, pp.350–354. Available from: `https://doi.org/10.1038/s41586-019-1724-z` [Accessed August 12, 2022].

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O., 2020. *What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study* [Online]. arXiv:2006.05990 [cs, stat]. arXiv. Available from: `http://arxiv.org/abs/2006.05990` [Accessed August 18, 2023].

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I., 2020. *Emergent Tool Use From Multi-Agent Autocurricula* [Online]. arXiv:1909.07528 [cs, stat]. arXiv. Available from: `http://arxiv.org/abs/1909.07528` [Accessed July 26, 2022].

Dafoe, A., Hughes, E., Bachrach, Y., Collins, T., McKee, K.R., Leibo, J.Z., Larson, K., and Graepel, T., 2020. *Open Problems in Cooperative AI* [Online]. arXiv:2012.08630 [cs]. arXiv. Available from: `http://arxiv.org/abs/2012.08630` [Accessed August 29, 2022].

Morales, M., 2020. *Grokking Deep Reinforcement Learning* [Online]. Manning Publications. Available from: `https://books.google.co.uk/books?id=IpHJzAEACAAJ`.

Russell, S., Russell, S., and Norvig, P., 2020. *Artificial Intelligence: A Modern Approach* [Online], Pearson series in artificial intelligence. Pearson. Available from: `https://books.google.co.uk/books?id=koFptAEACAAJ`.

Wang, T., Dong, H., Lesser, V., and Zhang, C., 2020. *ROMA: Multi-Agent Reinforcement Learning with Emergent Roles* [Online]. arXiv:2003.08039 [cs]. arXiv. Available from: `http://arxiv.org/abs/2003.08039` [Accessed August 21, 2023].

Witt, C.S. de, Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P.H.S., Sun, M., and Whiteson, S., 2020. *Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?* [Online]. arXiv:2011.09533 [cs]. arXiv. Available from: `http://arxiv.org/abs/2011.09533` [Accessed January 28, 2023].

Evans, J., 2021. *Week 1: the basic rl problem - what is reinforcement learning?* Lecture notes. PHYS 101 at University of Bath.

Papoudakis, G., Christianos, F., Schäfer, L., and Albrecht, S.V., 2021. *Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks* [Online]. arXiv:2006.07869 [cs, stat]. arXiv. Available from: `https://doi.org/10.48550/arXiv.2006.07869` [Accessed August 23, 2023].

Zhang, K., Yang, Z., and Başar, T., 2021. *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms* [Online]. arXiv:1911.10635 [cs, stat]. arXiv. Available from: `http://arxiv.org/abs/1911.10635` [Accessed August 25, 2023].

Huang, S. and Ontañón, S., 2022. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *The international flairs conference proceedings* [Online], 35. arXiv:2006.14171 [cs, stat]. Available from: `https://doi.org/10.32473/flairs.v35i.130584` [Accessed August 24, 2023].

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y., 2022. *The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games* [Online]. arXiv:2103.01955 [cs]. arXiv. Available from: `http://arxiv.org/abs/2103.01955` [Accessed November 7, 2022].

Appendix A

Software Design

A.1 Module Structure

The code for this project is separated into four separate modules:

- **Training**: For each experiment, a LeagueTrainer instance is generated which orchestrates all of the training sub-tasks including environment configuration and creation, agent training using PPO, and metric logging.
- **CTF**: Contains the logic for the capture the flag environment. The training module generates a GridworldCtf instance for each experiment to manage the environment during training.
- **Metrics**: Aggregates and stores metrics from the GridworldCtf class. A MetricsLogger instance is created by the training module for each experiment, to generate and store evaluation metrics during training.
- **PPO**: Contains the logic for my implementation of Proximal Policy Optimisation. A PPOTrainer instance is created by the training module for each experiment to manage the training of agent policies.

A.2 Source Code

All code developed in this study can be found in the following publicly accessible github repository: `https://github.com/g-nightingale/marl-ctf-development`

The commit hash as at the time of submission of this report is:

```
317b1c4c8226a5499931da502d4f767b4af3f22f
```

Please refer to the README.md file for descriptions of the contents of the repository.

A.3 CTF Environment Implementation Details

The CTF environment is implemented using the Python programming language. Hash tables are used where possible to maximise speed of operations. Numpy is used extensively when working with arrays and matrices. Pytorch has been used for neural network implementation. The Ray library has been used to parallelise some operations, such as environment rollouts and duels. Use of Ray improved code speed significantly. In the case of duels, under one configuration use of Ray reduced the duelling operation time from over 2 minutes to less than several time, decreasing

overall training time significantly. The environment API has been designed to closely resemble that of the OpenAI gym environments.

Appendix B

Action Masking

To test the efficacy of action masking, I train a Vaulter and a Guardian (T1) against two Scouts (T2). T1 are trained with both action masking and no action masking. T2 follow a stochastic policy taking random actions. T1 rewards are recorded to analyse the impact of action masking.

Five trials were conducted over 500,000 timesteps and average rewards recorded. From table B.1, average rewards increase from 16.95 to 23.04 when using action masking. This difference is significant at the 1% level (using Welch's t-test) and represents an increase of 35.9% . Figure B.1 depicts that action masking consistently generates higher average rewards over the entire training period.

Table B.1: Action Masking Results

	Masking μ (σ)	**No Masking μ (σ)**	**p-value**
Rewards at $t = 500k$	23.04 (1.63)	16.95 (2.1)	0.001

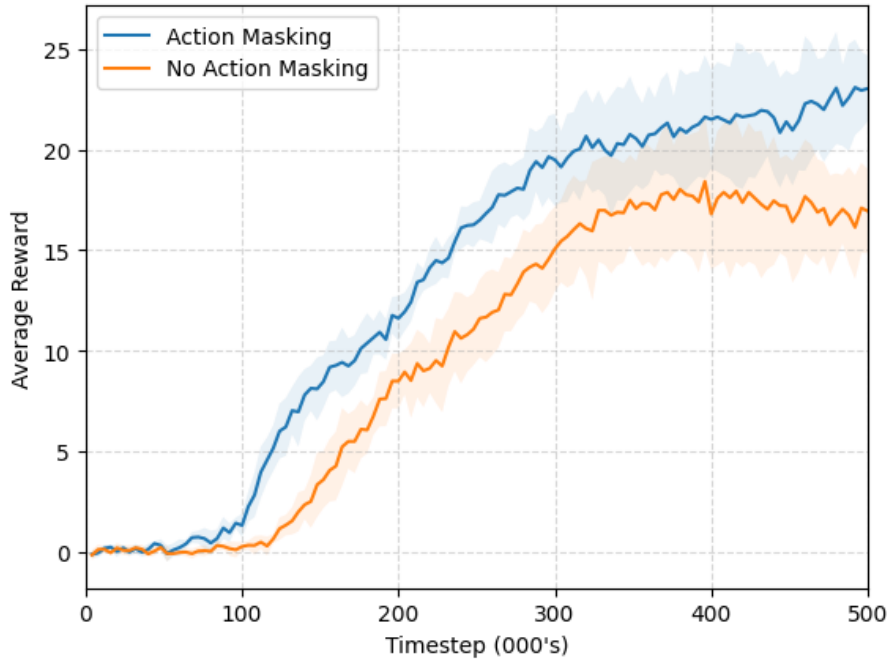

Figure B.1: Action Masking Results

Appendix C

Hyperparameter Tuning

There are a plethora of hyperparameters that can be adjusted within the deep learning and the wider self-play algorithm such as learning rate, batch size, discount rates, number of parallel environments and numerous others. This presents a challenge as poor choice of hyperparameters can impede learning. Jaderberg et al. (2019) use population based training, which combines traditional neural network training and evolutionary algorithms to optimise neural network weights and hyperparameters simultaneously during training.

This study uses a more simple approach. For the most part, this study uses established defaults proposed by Schulman et al. (2017). However, initial tests in our CTF environment found that agents had difficulty in learning gameplay basics such as capturing the flag. I hypothesise this is likely due to reward sparsity and have focused testing on hyperparameters that control discounting of rewards.

To find a combination of hyperparameters that would enable effective learning, a grid search was performed on γ, λ and n_{envs}. γ and λ are important as they control the level of discounting in the the environment. Recall that our agents receive sparse rewards, only receiving reward signals after successful flag capture or at the end of a match. Higher values of γ and λ enable rewards to be attributed to actions taken further in the past. Additionally, using more n_{envs} can reduce variance in training, as more samples are used when updating policies.

To conduct the hyperparameter tests, a team of two Scouts was trained against a second team of two Scouts who take random actions. A range of values for γ, λ and n_{envs} were selected for grid search. Five trials were conducted for each unique combination of the hyperparameters over 100,000 timesteps and average rewards over the five trials were recorded.

From results of the grid search, combinations with higher values of γ tend to produce larger rewards. This supports our hypothesis that attributing rewards to non-immediate actions is condusive to learning given the sparse rewards in our environment. I have chosen to use values from combination 40; $\gamma = 0.999, \lambda = 0.95$ and $n_{envs} = 8$. $n_{envs} = 12$ yielded the strongest results, however these were only marginally better than using $n_{envs} = 8$, and increased training times significantly.

Full results from hyperparameter testing are presented in table C.1, ordered by average rewards.

Table C.1: Hyperparameter Testing Results

Combination	γ	λ	**# of Environments**	**Avg Rewards**
41	0.999	0.95	12	25.83
28	0.99	0.95	8	25.13
39	0.999	0.95	4	22.83
40	0.999	0.95	8	22.29
25	0.99	0.9	8	21.54
32	0.99	0.99	12	21.5
35	0.99	0.999	12	21.33
38	0.999	0.9	12	20.86
36	0.999	0.9	4	20.75
29	0.99	0.95	12	20.64
20	0.95	0.99	12	20.33
27	0.99	0.95	4	20.25
47	0.999	0.999	12	20.17
31	0.99	0.99	8	20
34	0.99	0.999	8	19.13
44	0.999	0.99	12	17.44
43	0.999	0.99	8	16.79
22	0.95	0.999	8	16.71
33	0.99	0.999	4	15.92
16	0.95	0.95	8	15.75
46	0.999	0.999	8	15.25
23	0.95	0.999	12	15
37	0.999	0.9	8	14.63
19	0.95	0.99	8	14.04
13	0.95	0.9	8	13.96
42	0.999	0.99	4	13.75
8	0.9	0.99	12	13.53
1	0.9	0.9	8	12.58
26	0.99	0.9	12	12.53
24	0.99	0.9	4	12.5
3	0.9	0.95	4	11.08
30	0.99	0.99	4	10.5
11	0.9	0.999	12	10.36
5	0.9	0.95	12	9.94
4	0.9	0.95	8	9.83
6	0.9	0.99	4	9.58
7	0.9	0.99	8	9
21	0.95	0.999	4	8.67
14	0.95	0.9	12	8.61
18	0.95	0.99	4	8.5
45	0.999	0.999	4	8.42
17	0.95	0.95	12	8
0	0.9	0.9	4	7.5
2	0.9	0.9	12	6.89
12	0.95	0.9	4	6.25
9	0.9	0.999	4	6.25
15	0.95	0.95	4	4.5
10	0.9	0.999	8	1.96

Appendix D

Training Configuration

The following parameters were used in training across experiments. All experiments share the same parameters, with the exception of training iterations, where 100 iterations were used for experiment 9 and 50 iterations for all other experiments.

Table D.1: Training Configuration

	Value
Experiment trials	5
Training iterations	50 (exp. 1-8) or 100 (exp. 9)
Parallel rollouts per training iteration	8
Timesteps per rollout	500
Learning rate α	0.0003
Discount rate γ	0.999
GAE λ	0.95
Clipping ϵ	0.2
Value function coefficient	0.5
Entropy coefficient	0.01
Minibatches	4
Gradient updates per minibatch	4

Appendix E

Experiment Results

All results generated from the experiments are presented in this appendix. There is some duplication of artefacts presented in the results and analysis section (4). This is intentional so that the reader can examine all results in once place.

E.1 Experiment 1: The Split

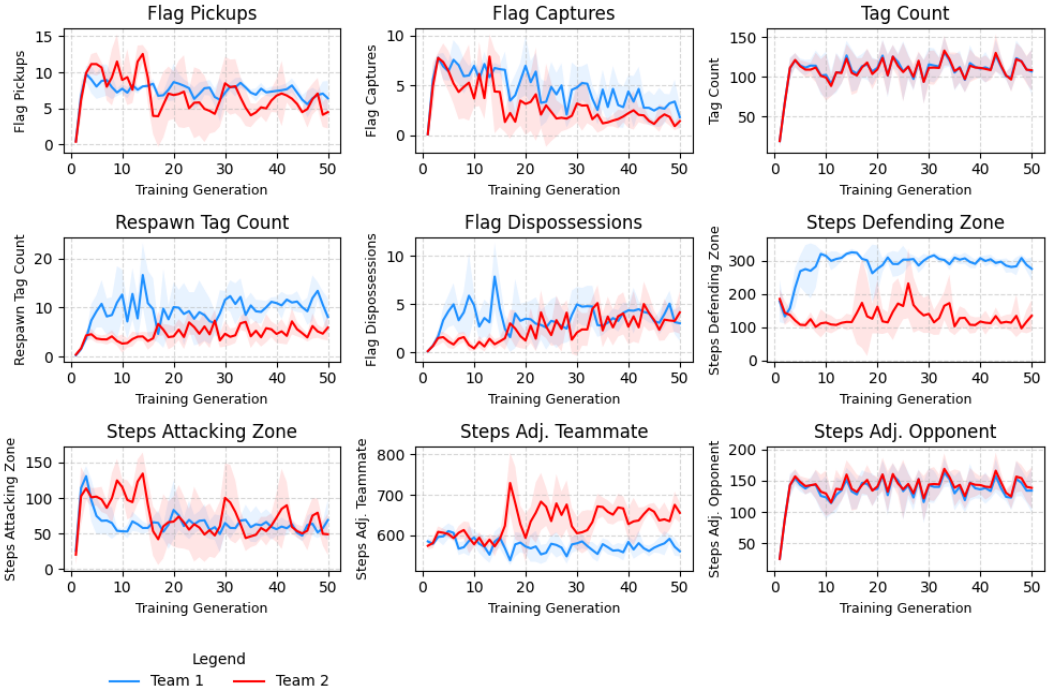

Figure E.1: Experiment 1: Team Metrics

Table E.1: Experiment 1: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	6.41 (2.39)	4.5 (2.28)	0.282
Flag captures	1.81 (1.22)	1.4 (0.69)	0.582
Tag count	106.65 (28)	108.39 (25.78)	0.929
Respawn tag count	8.07 (3.82)	5.93 (2.63)	0.388
Flag dispossessions	3.01 (1.84)	4.14 (2.2)	0.453
Steps defending zone	275.12 (33.88)	134.83 (47.15)	0.002
Steps attacking zone	68.85 (28.68)	48.85 (29.32)	0.358
Steps adj. teammate	560.91 (12.66)	655.65 (24.32)	<0.001
Steps adj. opponent	134.28 (34.86)	138.66 (32.69)	0.859

Figure E.2: Experiment 1: Agent Metrics

Table E.2: Experiment 1: Team 1 Agent Metrics at 50th Training Generation

Metric	**Guardian** μ (σ)	**Scout** μ (σ)	**p-value**
Flag pickups	0.05 (0.05)	6.35 (2.44)	0.007
Flag captures	0.05 (0.05)	1.75 (1.22)	0.049
Tag count	28.09 (12.06)	78.57 (27.12)	0.017
Respawn tag count	6.35 (3.35)	1.72 (0.6)	0.05
Flag dispossessions	2.73 (1.66)	0.28 (0.22)	0.042
Steps defending zone	231.2 (27.37)	43.92 (10.4)	<0.001
Steps attacking zone	2.87 (5.22)	65.97 (23.67)	0.005
Steps adj. teammate	280.17 (6.38)	280.73 (6.33)	0.904
Steps adj. opponent	31.21 (13.27)	103.07 (36.23)	0.013

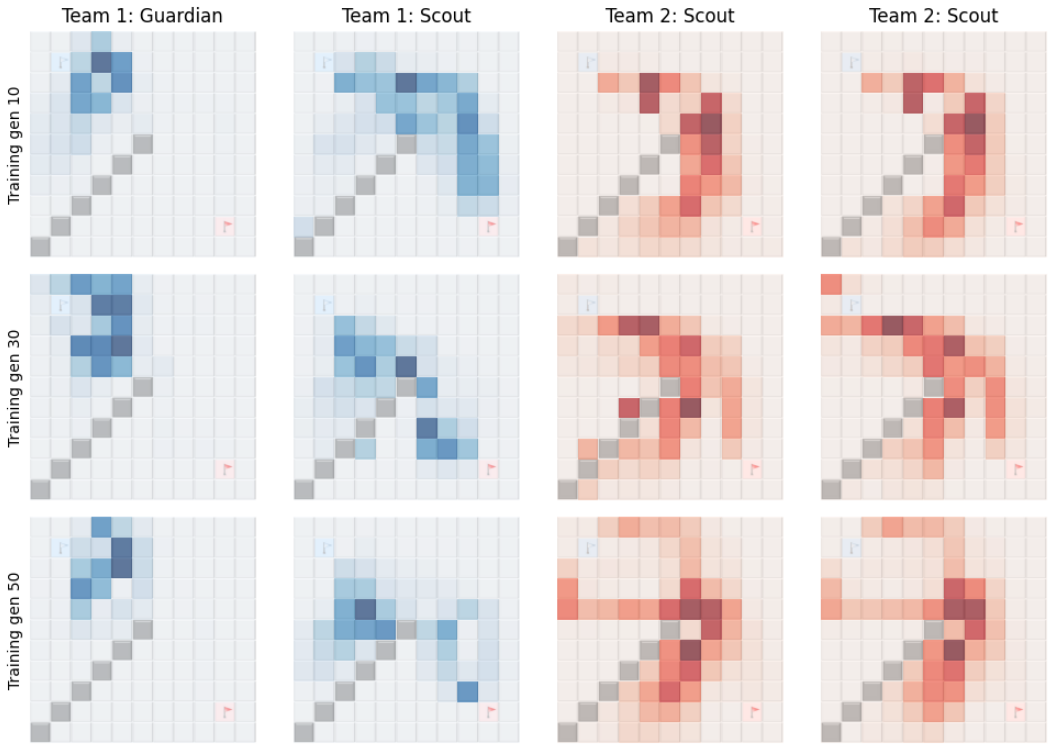

Figure E.3: Experiment 1: Agent Visitation Maps

E.2 Experiment 2: The Fence

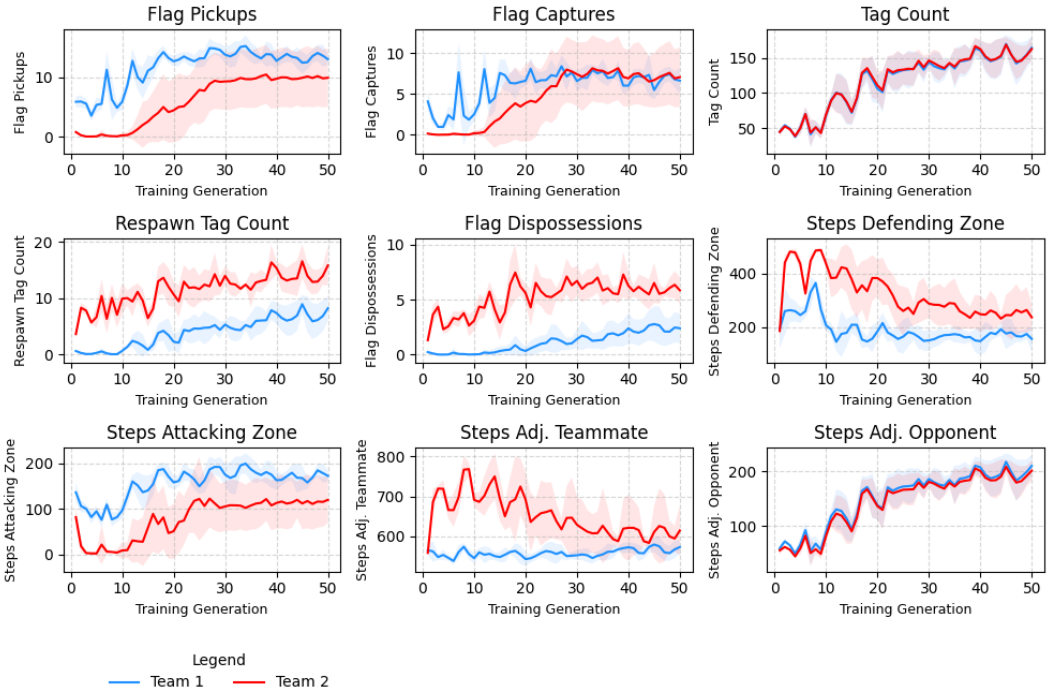

Figure E.4: Experiment 2: Team Metrics

Table E.3: Experiment 2: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	13.01 (0.95)	9.89 (4.88)	0.274
Flag captures	6.63 (1.55)	7.05 (3.65)	0.84
Tag count	164.26 (17.04)	162.48 (16.68)	0.885
Respawn tag count	8.24 (2.13)	15.82 (3.56)	0.009
Flag dispossessions	2.41 (1.34)	5.85 (1.21)	0.005
Steps defending zone	156.68 (46.7)	236.96 (47.86)	0.043
Steps attacking zone	172.79 (13.01)	120.17 (50.85)	0.107
Steps adj. teammate	573.58 (16.59)	614.93 (61.54)	0.256
Steps adj. opponent	209.87 (20.1)	201.09 (18.34)	0.537

Table E.4: Experiment 2: Team 1 Agent Metrics at 50th Training Generation

Metric	**Scout** μ (σ)	**Vaulter** μ (σ)	**p-value**
Flag pickups	6.1 (1.26)	6.91 (0.86)	0.325
Flag captures	3.83 (1.01)	2.8 (1.19)	0.223
Tag count	96.75 (9.5)	67.51 (11.29)	0.004
Respawn tag count	5.15 (1.23)	3.09 (1.09)	0.038
Flag dispossessions	1.57 (0.83)	0.84 (0.56)	0.191
Steps defending zone	83.9 (19.23)	72.78 (28.78)	0.541
Steps attacking zone	85.03 (13.76)	87.76 (6.48)	0.732
Steps adj. teammate	286.92 (8.12)	286.66 (8.49)	0.966
Steps adj. opponent	123.94 (10.64)	85.93 (13.9)	0.003

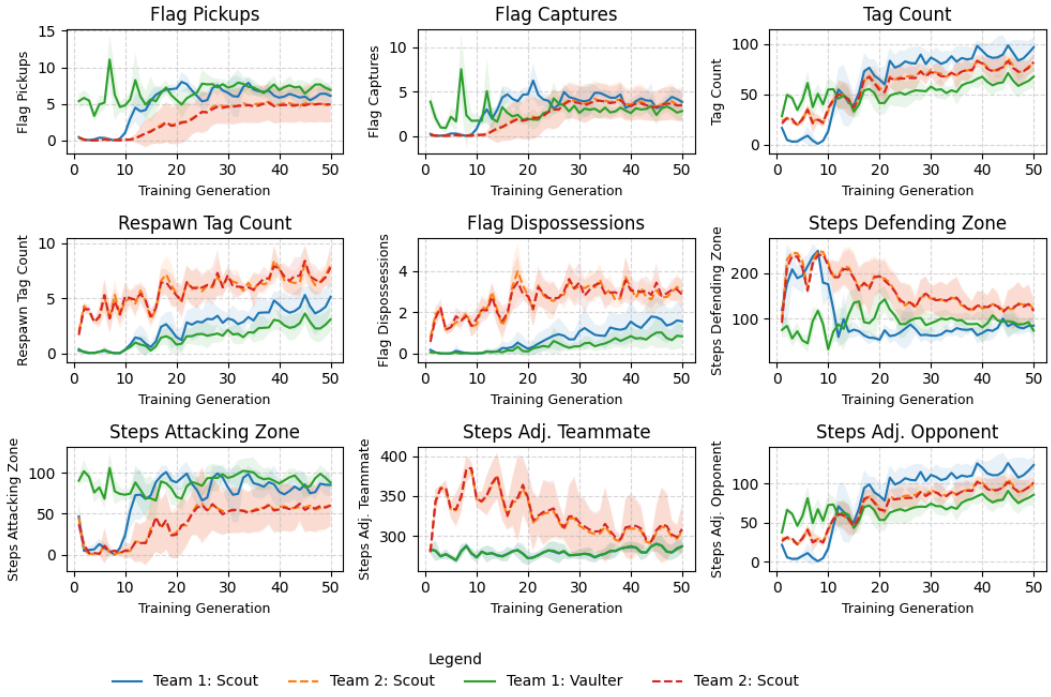

Figure E.5: Experiment 2: Agent Metrics

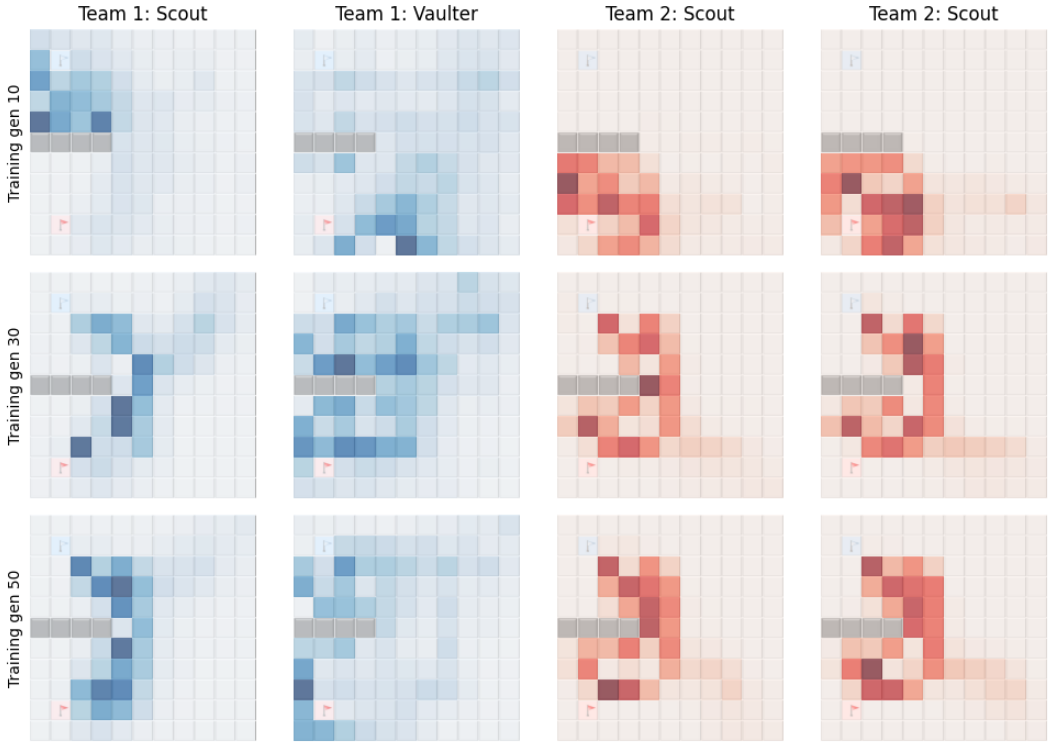

Figure E.6: Experiment 2: Agent Visitation Maps

E.3 Experiment 3: Jailbreak

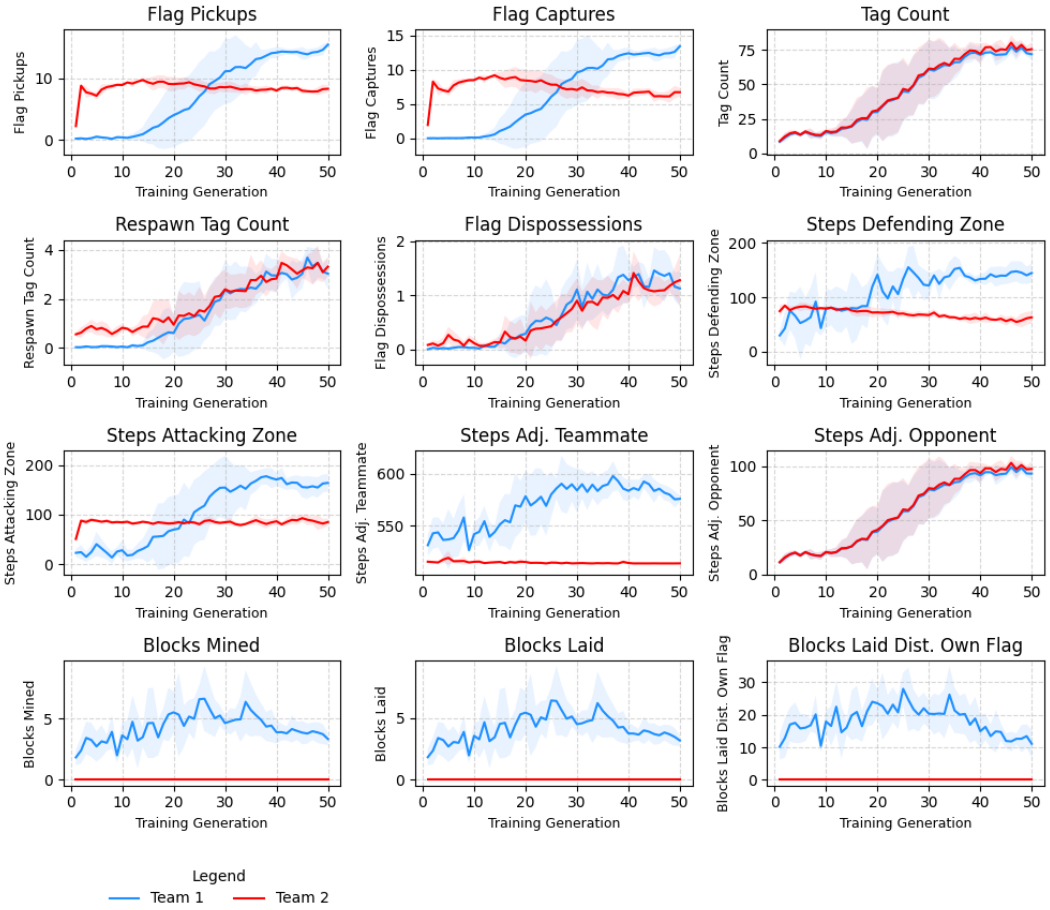

Figure E.7: Experiment 3: Team Metrics

Table E.5: Experiment 3: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	15.52 (0.37)	8.33 (0.53)	<0.001
Flag captures	13.39 (0.38)	6.71 (0.64)	<0.001
Tag count	71.71 (4.34)	75.42 (5.05)	0.299
Respawn tag count	3.03 (0.62)	3.32 (0.39)	0.462
Flag dispossessions	1.13 (0.33)	1.28 (0.47)	0.607
Steps defending zone	144.97 (17.68)	63.42 (11.02)	<0.001
Steps attacking zone	163.99 (17.58)	85.31 (7.06)	<0.001
Steps adj. teammate	575.75 (3.24)	514.06 (0.12)	<0.001
Steps adj. opponent	93.19 (5.42)	97.59 (6.93)	0.348
Blocks mined	3.34 (0.49)	0 (0)	<0.001
Blocks laid	3.17 (0.45)	0 (0)	<0.001
Blocks laid dist. from own flag	11.07 (4.32)	0 (0)	0.007
Blocks laid dist. from opp flag	18.33 (2.45)	0 (0)	<0.001

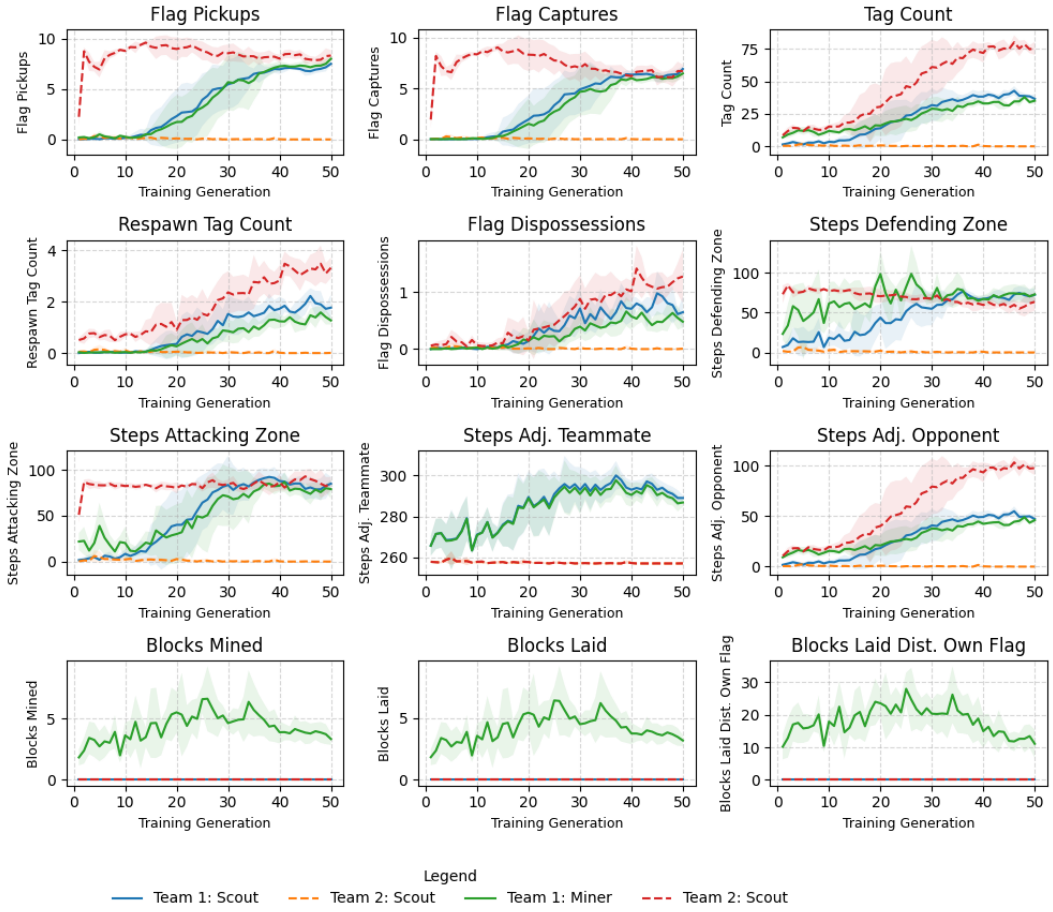

Figure E.8: Experiment 3: Agent Metrics

Table E.6: Experiment 3: Team 1 Agent Metrics at 50th Training Generation

Metric	**Scout μ (σ)**	**Miner μ (σ)**	**p-value**
Flag pickups	7.51 (0.17)	8.01 (0.22)	0.008
Flag captures	6.92 (0.22)	6.47 (0.33)	0.055
Tag count	36.57 (3.39)	35.14 (1.92)	0.489
Respawn tag count	1.77 (0.51)	1.27 (0.16)	0.121
Flag dispossessions	0.65 (0.28)	0.48 (0.11)	0.314
Steps defending zone	72.23 (8.01)	72.75 (11.23)	0.942
Steps attacking zone	85 (7.52)	78.99 (10.21)	0.373
Steps adj. teammate	288.97 (1.33)	286.77 (2.1)	0.121
Steps adj. opponent	47.13 (4.01)	46.07 (2.65)	0.673
Blocks mined	0 (0)	3.34 (0.49)	$<$0.001
Blocks laid	0 (0)	3.17 (0.45)	$<$0.001
Blocks laid dist. from own flag	0 (0)	11.07 (4.32)	0.007
Blocks laid dist. from opp flag	0 (0)	18.33 (2.45)	$<$0.001

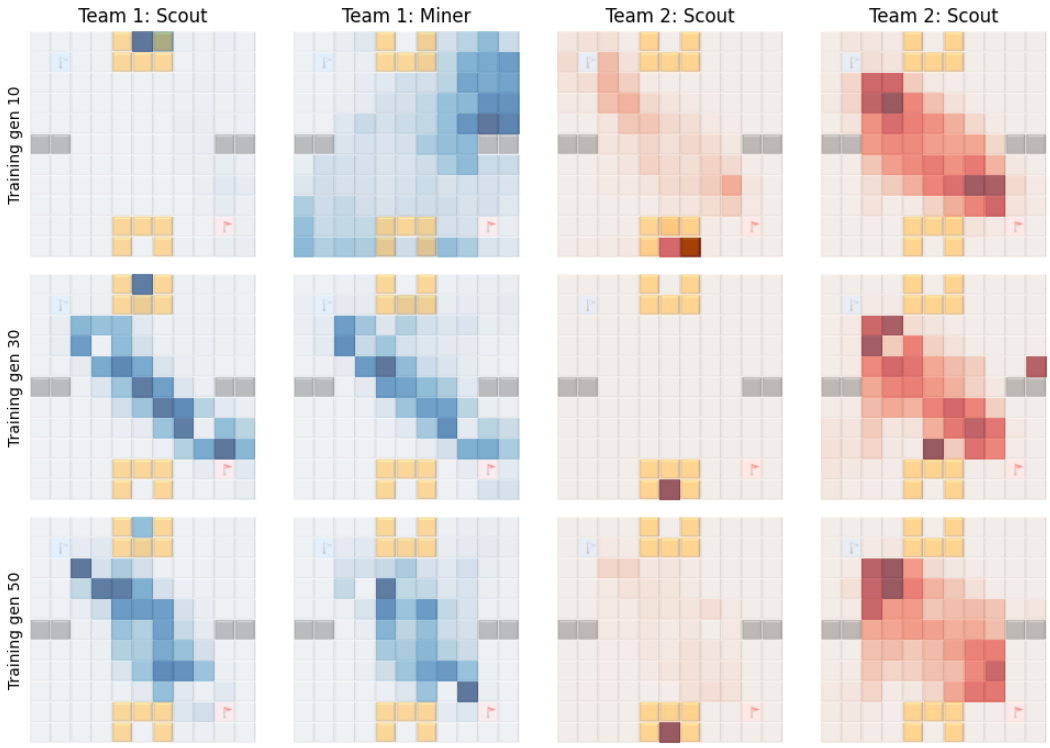

Figure E.9: Experiment 3: Agent Visitation Maps

E.4 Experiment 4: One way out

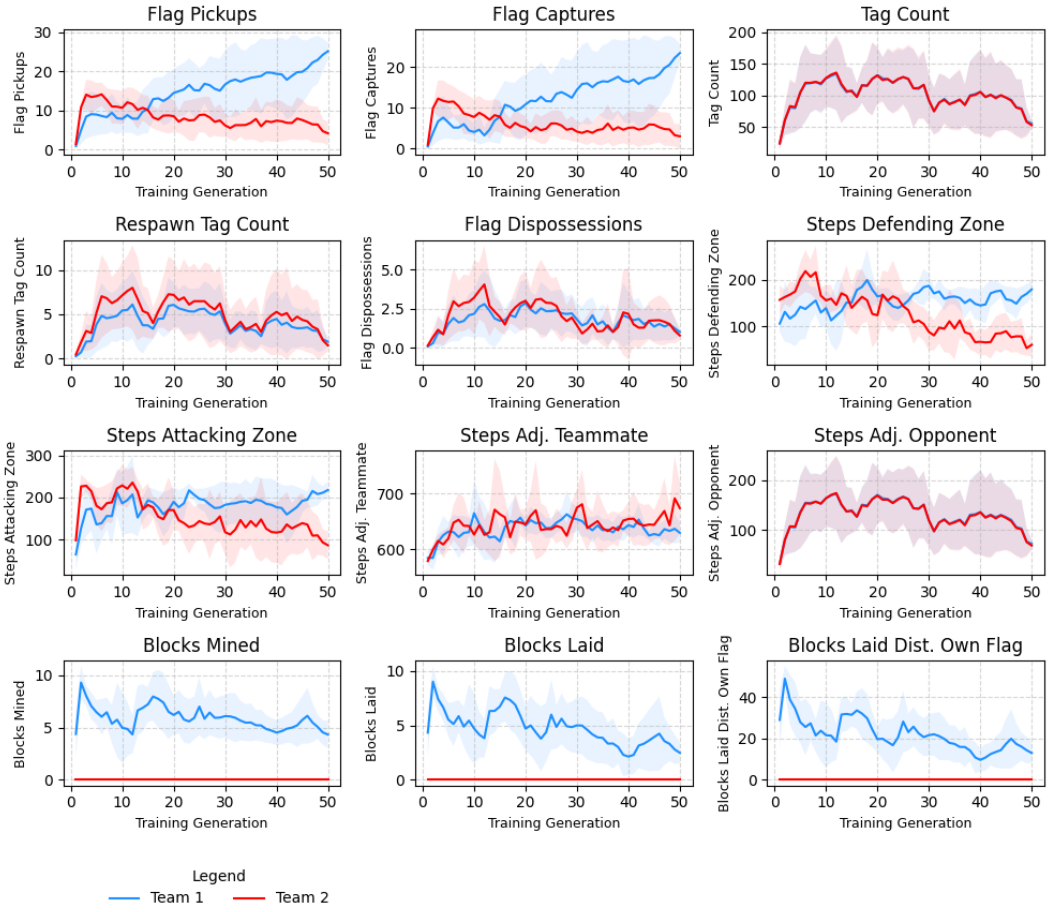

Figure E.10: Experiment 4: Team Metrics

Table E.7: Experiment 4: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	25.09 (2.17)	4.11 (3.05)	<0.001
Flag captures	23.45 (2.19)	2.95 (3.06)	<0.001
Tag count	55.68 (6.31)	53.06 (6.95)	0.592
Respawn tag count	1.9 (0.51)	1.47 (0.65)	0.328
Flag dispossessions	1.01 (0.28)	0.79 (0.34)	0.334
Steps defending zone	179.02 (17.41)	60.91 (24.26)	<0.001
Steps attacking zone	217.9 (20.49)	86.44 (32.57)	<0.001
Steps adj. teammate	629.71 (18.27)	674.24 (45.63)	0.127
Steps adj. opponent	72.54 (9.06)	68.84 (9.63)	0.591
Blocks mined	4.33 (1.31)	0 (0)	0.003
Blocks laid	2.49 (0.97)	0 (0)	0.007
Blocks laid dist. from own flag	12.97 (4.68)	0 (0)	0.005
Blocks laid dist. from opp flag	10.27 (4.57)	0 (0)	0.011

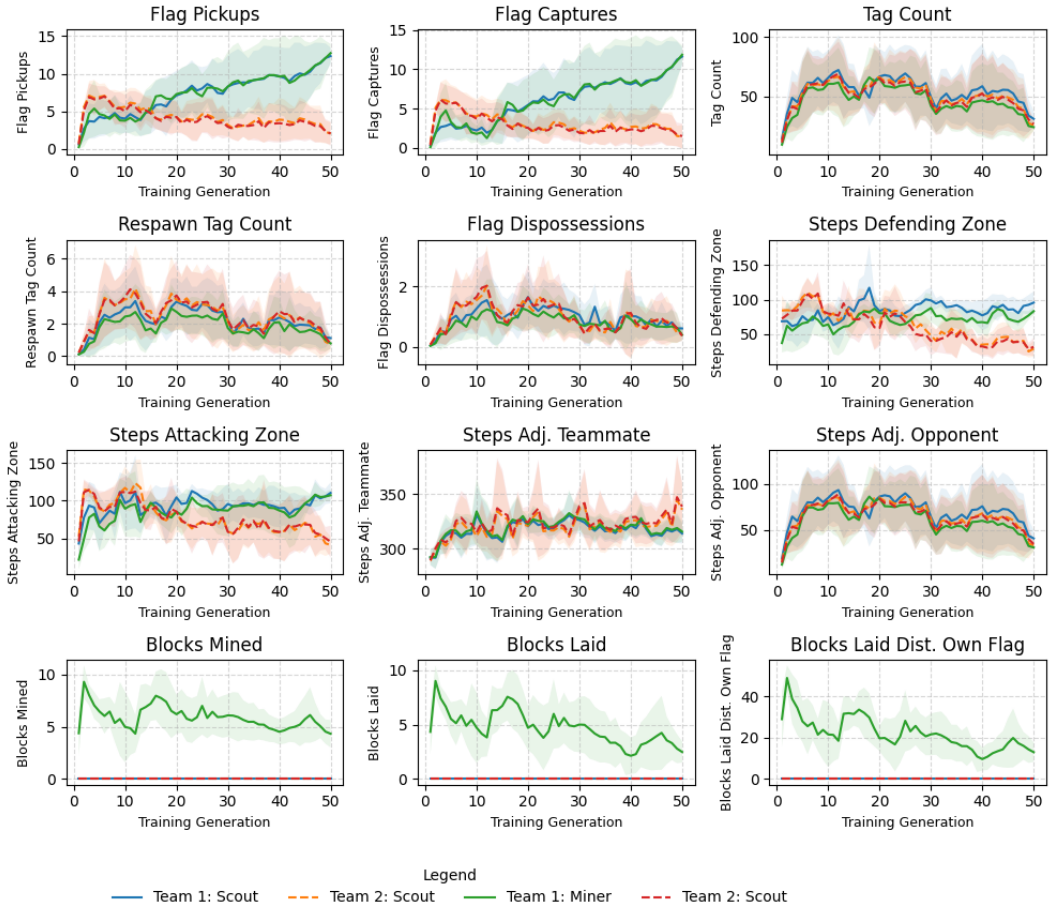

Figure E.11: Experiment 4: Agent Metrics

Table E.8: Experiment 4: Team 1 Agent Metrics at 50th Training Generation

Metric	**Scout** μ (σ)	**Miner** μ (σ)	**p-value**
Flag pickups	12.35 (1.12)	12.74 (1.24)	0.656
Flag captures	11.61 (1.16)	11.84 (1.14)	0.782
Tag count	31.36 (6.52)	24.32 (3.2)	0.102
Respawn tag count	1.11 (0.23)	0.79 (0.33)	0.149
Flag dispossessions	0.61 (0.22)	0.4 (0.19)	0.175
Steps defending zone	95.68 (15.99)	83.34 (6.08)	0.207
Steps attacking zone	110.53 (13.8)	107.37 (7.58)	0.702
Steps adj. teammate	313.97 (9.12)	315.75 (9.16)	0.79
Steps adj. opponent	41.19 (8.9)	31.35 (4.22)	0.095
Blocks mined	0 (0)	4.33 (1.31)	0.003
Blocks laid	0 (0)	2.49 (0.97)	0.007
Blocks laid dist. from own flag	0 (0)	12.97 (4.68)	0.005
Blocks laid dist. from opp flag	0 (0)	10.27 (4.57)	0.011

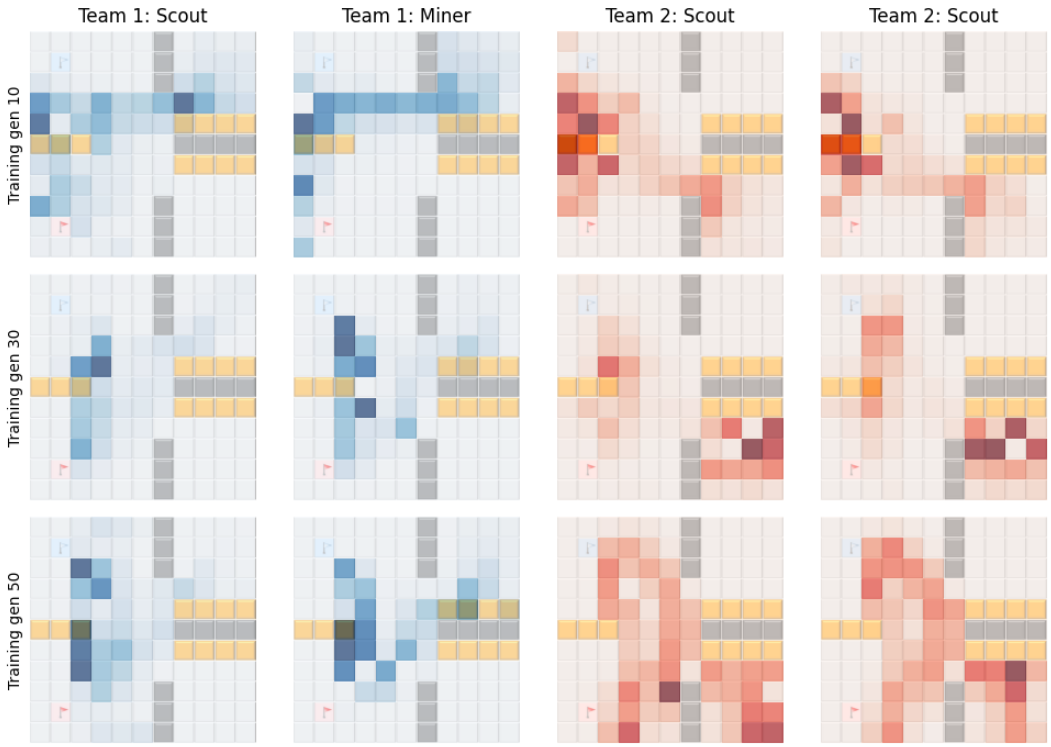

Figure E.12: Experiment 4: Agent Visitation Maps

E.5 Experiment 5: Keyhole

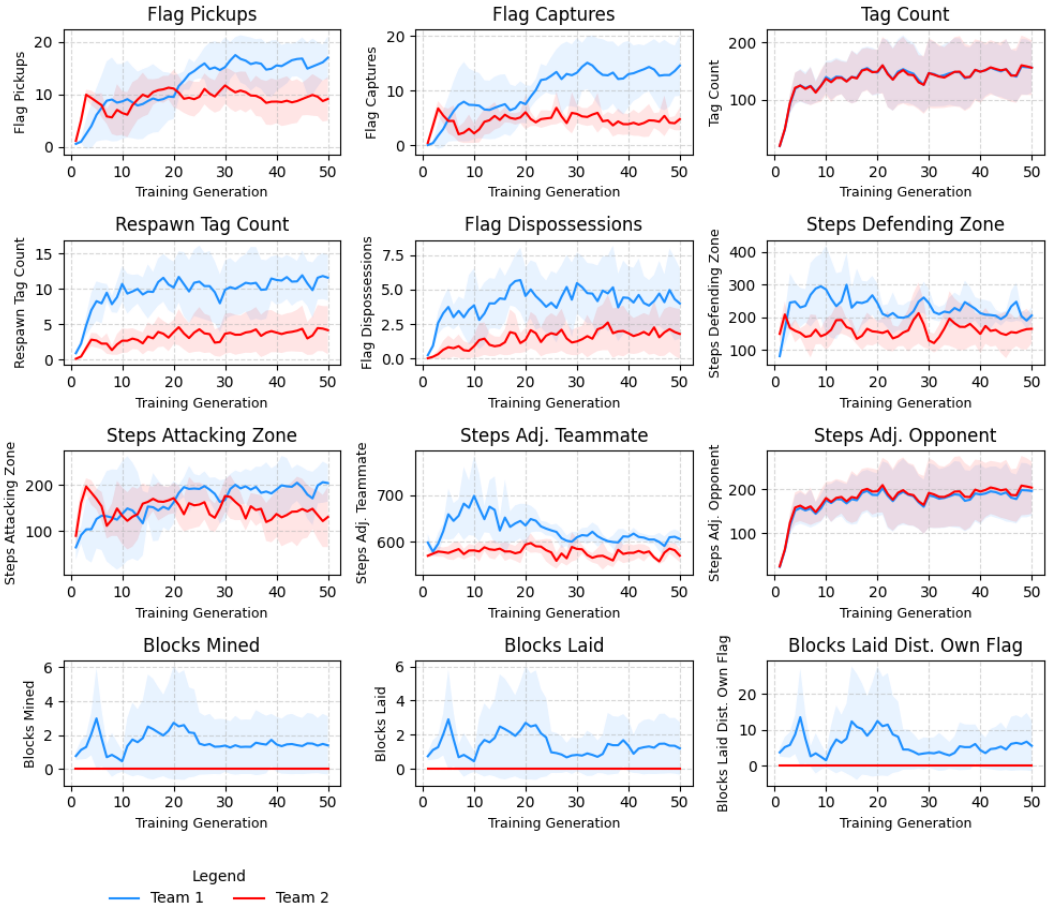

Figure E.13: Experiment 5: Team Metrics

Table E.9: Experiment 5: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	16.96 (4.05)	9.11 (4.14)	0.027
Flag captures	14.56 (5.51)	4.79 (2.07)	0.02
Tag count	155.81 (44.01)	155.75 (47.78)	0.999
Respawn tag count	11.58 (3.19)	4.18 (3.39)	0.013
Flag dispossessions	4 (2.54)	1.79 (1.8)	0.198
Steps defending zone	205.57 (24.63)	164.75 (44.34)	0.157
Steps attacking zone	204.65 (40.29)	131.04 (65.6)	0.1
Steps adj. teammate	605.99 (15.42)	570.16 (13.68)	0.009
Steps adj. opponent	195.74 (55.67)	203.23 (59.15)	0.858
Blocks mined	1.39 (1.71)	0 (0)	0.179
Blocks laid	1.21 (1.53)	0 (0)	0.191
Blocks laid dist. from own flag	5.53 (6.99)	0 (0)	0.188
Blocks laid dist. from opp flag	5.97 (7.74)	0 (0)	0.198

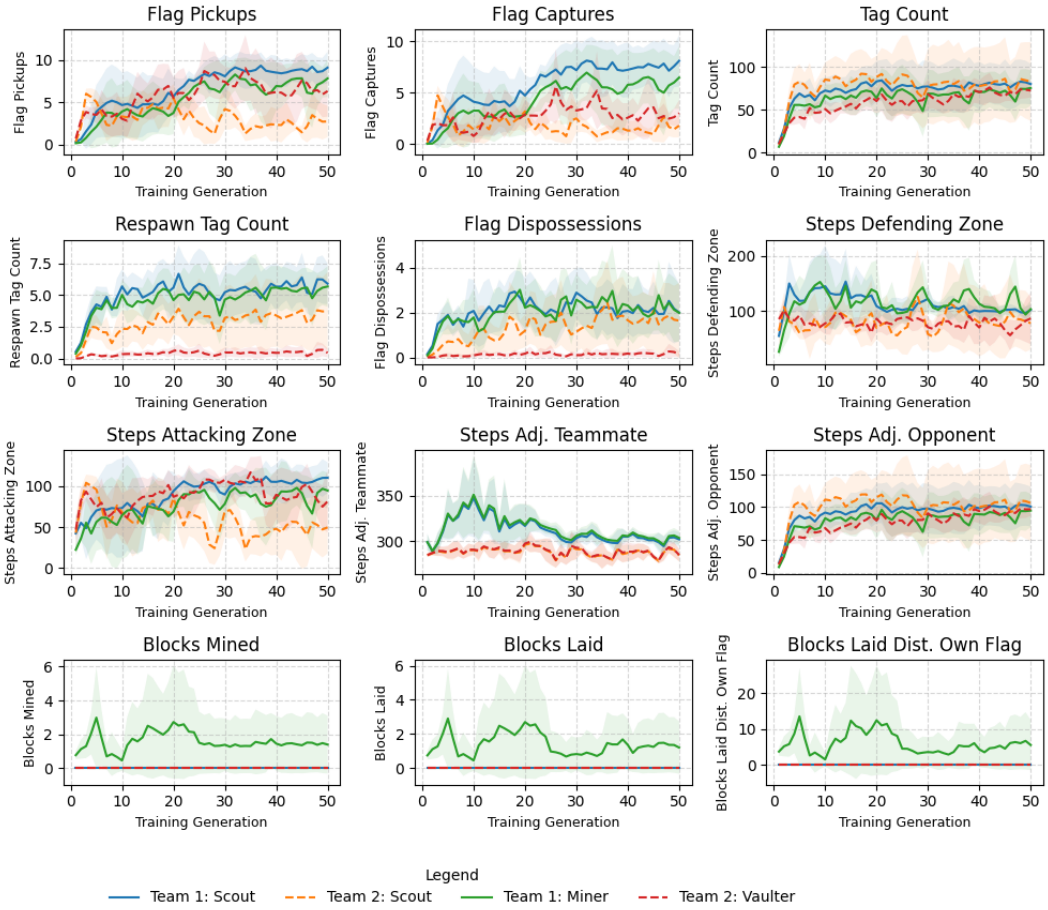

Figure E.14: Experiment 5: Agent Metrics

Table E.10: Experiment 5: Team 1 Agent Metrics at 50th Training Generation

Metric	**Scout** μ (σ)	**Miner** μ (σ)	**p-value**
Flag pickups	9.12 (1.87)	7.84 (2.19)	0.4
Flag captures	8.09 (2.49)	6.47 (3.03)	0.432
Tag count	80.2 (23.53)	75.61 (20.54)	0.776
Respawn tag count	5.9 (1.6)	5.68 (1.6)	0.851
Flag dispossessions	2.01 (1.28)	1.99 (1.28)	0.989
Steps defending zone	102.11 (9.75)	103.46 (15.32)	0.886
Steps attacking zone	110.11 (18.76)	94.54 (21.72)	0.31
Steps adj. teammate	302.17 (7.86)	303.81 (7.61)	0.772
Steps adj. opponent	100.91 (30.2)	94.83 (25.52)	0.766
Blocks mined	0 (0)	1.39 (1.71)	0.179
Blocks laid	0 (0)	1.21 (1.53)	0.191
Blocks laid dist. from own flag	0 (0)	5.53 (6.99)	0.188
Blocks laid dist. from opp flag	0 (0)	5.97 (7.74)	0.198

Table E.11: Experiment 5: Team 2 Agent Metrics at 50th Training Generation

Metric	**Scout μ (σ)**	**Vaulter μ (σ)**	**p-value**
Flag pickups	2.76 (1.85)	6.35 (3.02)	0.084
Flag captures	1.79 (1.14)	3.01 (1.29)	0.194
Tag count	83.14 (45.92)	72.61 (7.9)	0.674
Respawn tag count	3.73 (3.08)	0.45 (0.35)	0.101
Flag dispossessions	1.65 (1.68)	0.15 (0.14)	0.149
Steps defending zone	78.06 (47.7)	86.69 (46.8)	0.803
Steps attacking zone	49.69 (31.69)	81.35 (37.94)	0.237
Steps adj. teammate	284.87 (6.77)	285.29 (6.97)	0.934
Steps adj. opponent	106.43 (57.56)	96.81 (9.97)	0.758

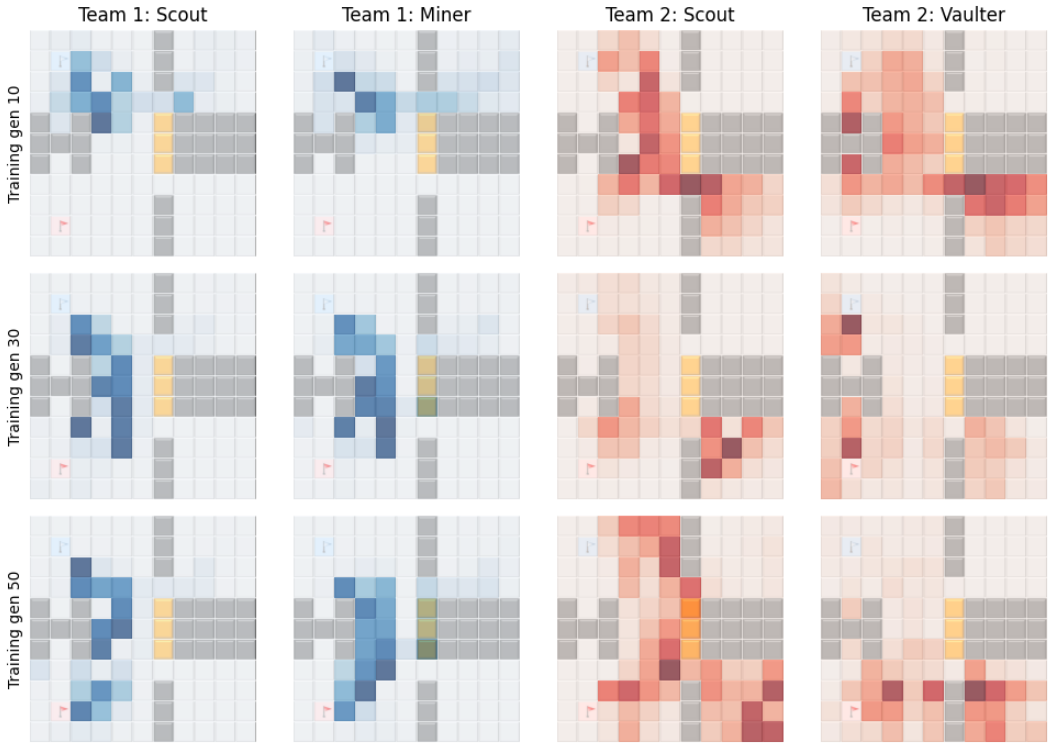

Figure E.15: Experiment 4: Agent Visitation Maps

E.6 Experiment 6: Skittles

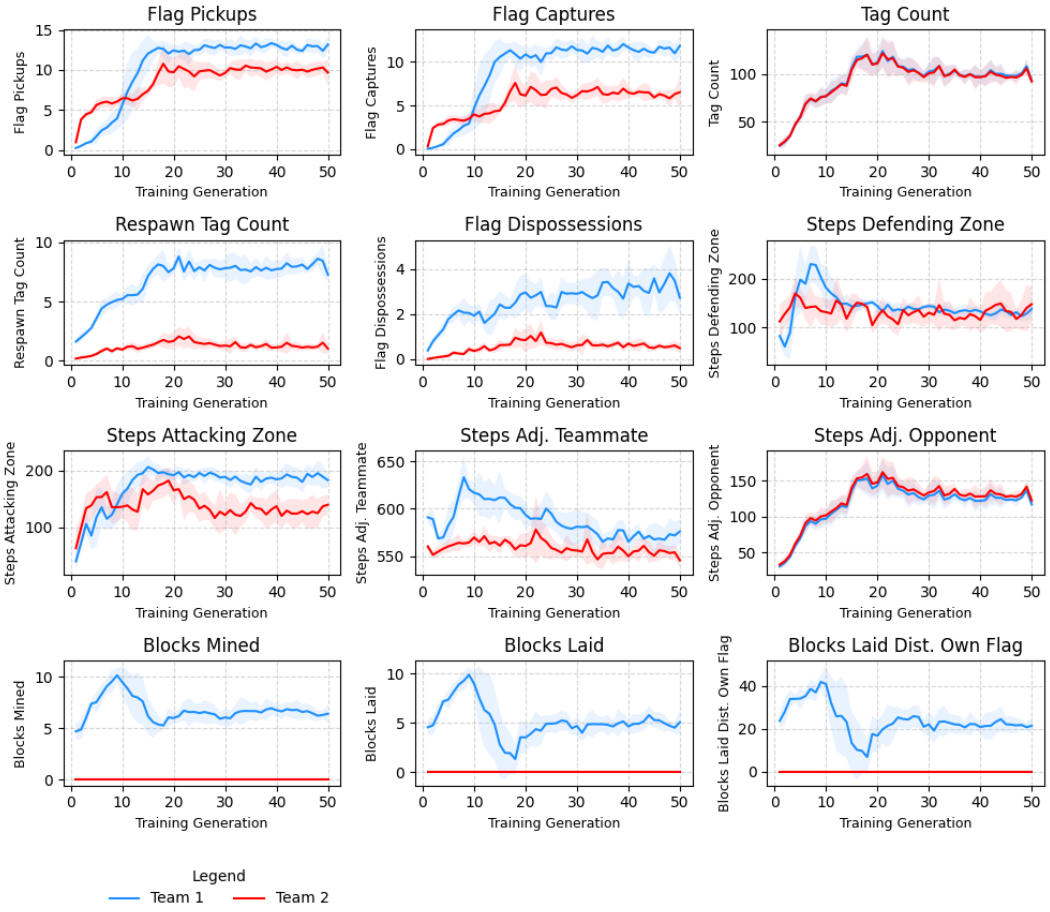

Figure E.16: Experiment 6: Team Metrics

Table E.12: Experiment 6: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	13.18 (0.96)	9.66 (1.63)	0.009
Flag captures	11.84 (1.12)	6.51 (1.78)	0.002
Tag count	93.24 (3.89)	92.53 (3.94)	0.805
Respawn tag count	7.27 (0.52)	1.03 (0.4)	$<$0.001
Flag dispossessions	2.72 (0.64)	0.49 (0.23)	0.001
Steps defending zone	137.99 (18.39)	147.77 (35.76)	0.644
Steps attacking zone	183.09 (12.29)	139.57 (18.03)	0.005
Steps adj. teammate	575.91 (12.67)	545.5 (7.68)	0.005
Steps adj. opponent	116.9 (4.9)	122.11 (5.14)	0.181
Blocks mined	6.41 (0.88)	0 (0)	$<$0.001
Blocks laid	5.08 (1.19)	0 (0)	0.001
Blocks laid dist. from own flag	21.39 (3.49)	0 (0)	$<$0.001
Blocks laid dist. from opp flag	25.04 (7.43)	0 (0)	0.003

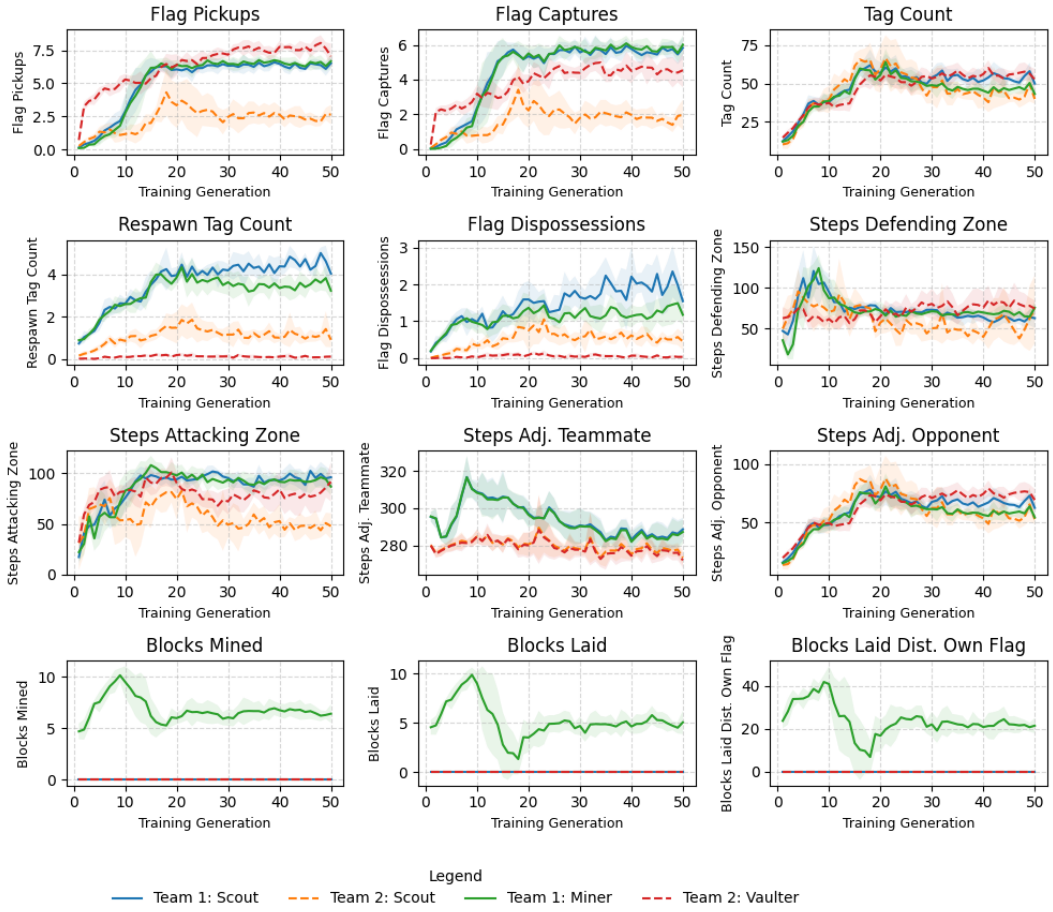

Figure E.17: Experiment 6: Agent Metrics

Table E.13: Experiment 6: Team 1 Agent Metrics at 50th Training Generation

Metric	**Scout μ (σ)**	**Miner μ (σ)**	**p-value**
Flag pickups	6.51 (0.45)	6.67 (0.54)	0.672
Flag captures	5.83 (0.64)	6.01 (0.54)	0.667
Tag count	50 (2.7)	43.24 (2.52)	0.006
Respawn tag count	4.04 (0.36)	3.23 (0.41)	0.019
Flag dispossessions	1.55 (0.4)	1.17 (0.28)	0.167
Steps defending zone	62.56 (11.92)	75.43 (7.09)	0.109
Steps attacking zone	96.24 (6.84)	86.85 (8.11)	0.116
Steps adj. teammate	288.66 (6.14)	287.25 (6.54)	0.762
Steps adj. opponent	62.55 (3.7)	54.35 (3.53)	0.013
Blocks mined	0 (0)	6.41 (0.88)	$<$0.001
Blocks laid	0 (0)	5.08 (1.19)	0.001
Blocks laid dist. from own flag	0 (0)	21.39 (3.49)	$<$0.001
Blocks laid dist. from opp flag	0 (0)	25.04 (7.43)	0.003

Table E.14: Experiment 6: Team 2 Agent Metrics at 50th Training Generation

Metric	**Scout μ (σ)**	**Vaulter μ (σ)**	**p-value**
Flag pickups	2.63 (0.73)	7.03 (0.96)	<0.001
Flag captures	1.97 (0.86)	4.54 (1)	0.005
Tag count	39.93 (3.05)	52.61 (6.92)	0.018
Respawn tag count	0.91 (0.35)	0.12 (0.08)	0.009
Flag dispossessions	0.45 (0.23)	0.03 (0.04)	0.021
Steps defending zone	73.46 (41.77)	74.31 (9.9)	0.97
Steps attacking zone	47.87 (12.64)	91.71 (19.34)	0.007
Steps adj. teammate	273.45 (3.85)	272.05 (3.88)	0.621
Steps adj. opponent	52.86 (4.24)	69.25 (9.31)	0.02

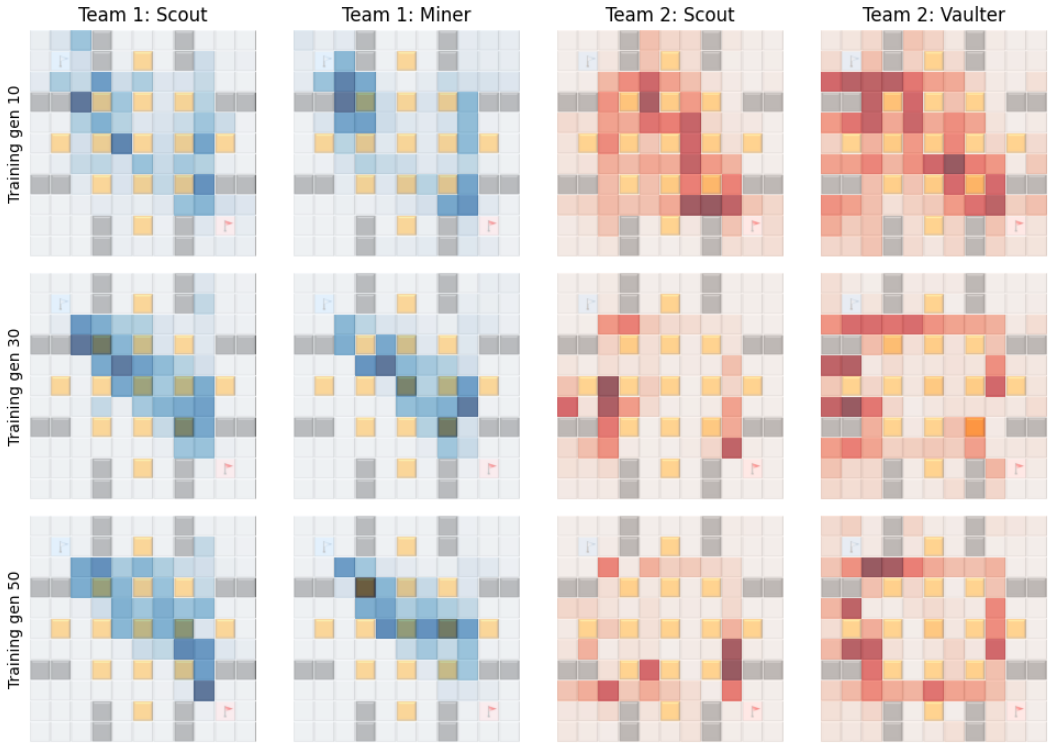

Figure E.18: Experiment 6: Agent Visitation Maps

E.7 Experiment 7: The Wall

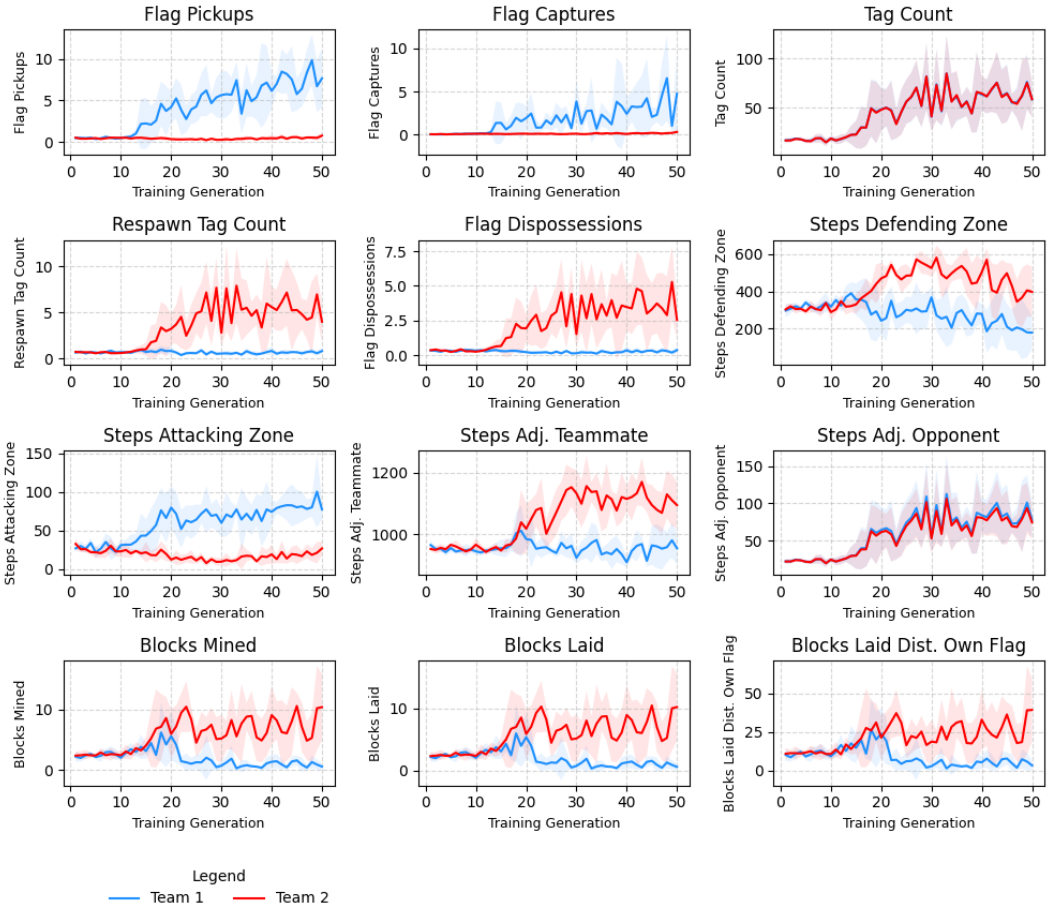

Figure E.19: Experiment 7: Team Metrics

Table E.15: Experiment 7: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	7.65 (3.55)	0.75 (0.39)	0.017
Flag captures	4.75 (4.69)	0.3 (0.21)	0.13
Tag count	58.65 (17.45)	58.82 (18.65)	0.99
Respawn tag count	0.81 (0.33)	4 (2.22)	0.044
Flag dispossessions	0.36 (0.24)	2.53 (1.31)	0.028
Steps defending zone	178.54 (90.99)	397.39 (133.47)	0.03
Steps attacking zone	77.58 (13.05)	26.95 (9.27)	$<$0.001
Steps adj. teammate	954.51 (52.37)	1094.91 (74)	0.017
Steps adj. opponent	77.78 (23.53)	74.37 (22.51)	0.839
Blocks mined	0.57 (0.57)	10.43 (5.5)	0.023
Blocks laid	0.55 (0.54)	10.2 (5.42)	0.023
Blocks laid dist. from own flag	3.19 (3.28)	39.35 (20.74)	0.024
Blocks laid dist. from opp flag	3.47 (3.38)	63.72 (33.66)	0.023

In complex environments with 3 or 4 agents, p-values are reported between each pair of agents. For example, in table E.16, $p_{(1,2)}$ represents the p-value between the Miner (agent 1) and Vaulter (agent 2). Similarly, $p_{(2,3)}$ represents the p-value between the Vaulter (agent 2) and Guardian (agent 3).

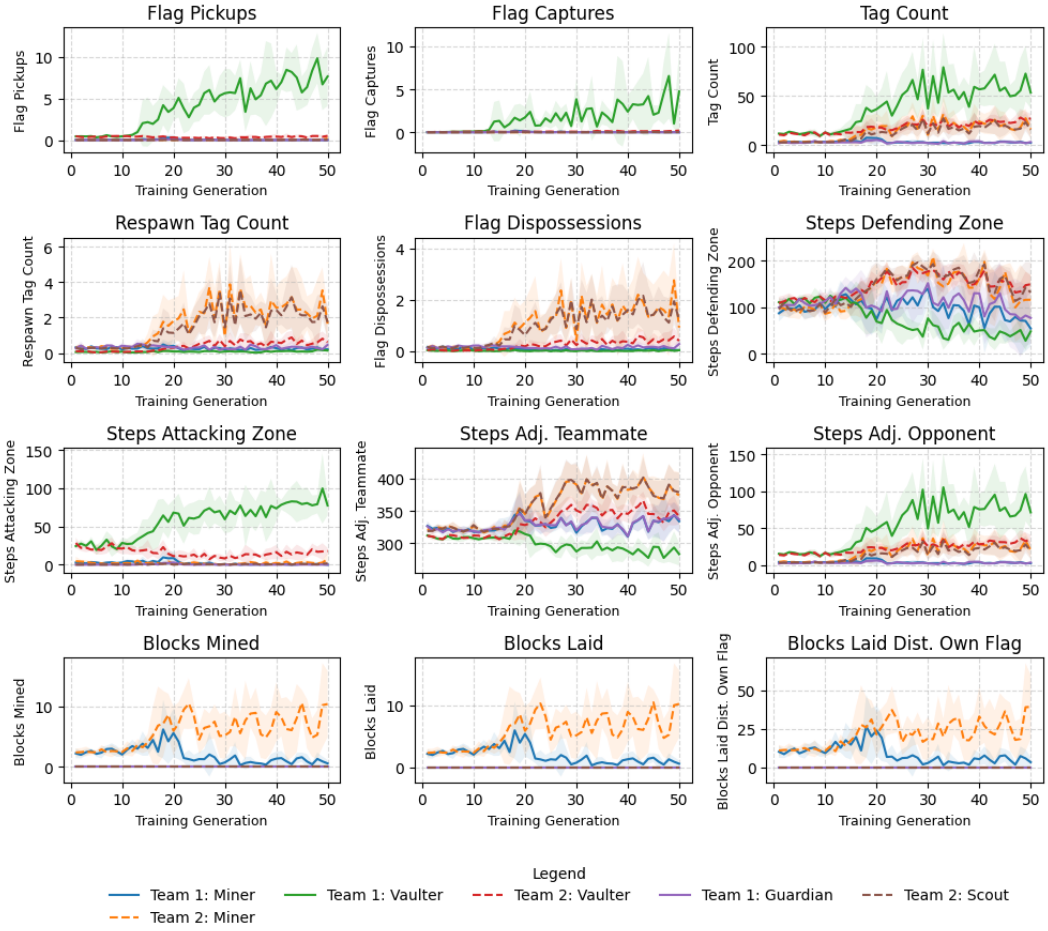

Figure E.20: Experiment 7: Agent Metrics

Table E.16: Experiment 7: Team 1 Agent Metrics at 50th Training Generation

Metric	**Miner**$_1$ μ (σ)	**Vaulter**$_2$ μ (σ)	**Guardian**$_3$ μ (σ)	$\mathbf{p}_{(1,2)}$	$\mathbf{p}_{(1,3)}$	$\mathbf{p}_{(2,3)}$
Flag pickups	0 (0)	7.65 (3.55)	0 (0)	0.013	-	0.013
Flag captures	0 (0)	4.75 (4.69)	0 (0)	0.112	-	0.112
Tag count	2.33 (0.62)	53.29 (15.92)	3.03 (1.34)	0.003	0.381	0.003
Respawn tag count	0.21 (0.09)	0.14 (0.1)	0.46 (0.34)	0.31	0.223	0.133
Flag dispossessions	0.04 (0.01)	0.04 (0.04)	0.28 (0.24)	1.00	0.112	0.112
Steps defending zone	53.9 (38.36)	47.86 (15.95)	76.78 (56.11)	0.782	0.522	0.37
Steps attacking zone	0 (0)	77.58 (13.05)	0 (0)	$<$0.001	-	$<$0.001
Steps adj. teammate	334.03 (18.13)	283.21 (17.38)	337.26 (20.79)	0.004	0.821	0.004
Steps adj. opponent	2.75 (0.67)	71.41 (21.71)	3.62 (1.54)	0.003	0.341	0.003
Blocks mined	0.57 (0.57)	0 (0)	0 (0)	0.115	0.115	-
Blocks laid	0.55 (0.54)	0 (0)	0 (0)	0.112	0.112	-
Blocks laid dist. from own flag	3.19 (3.28)	0 (0)	0 (0)	0.123	0.123	-
Blocks laid dist. from opp flag	3.47 (3.38)	0 (0)	0 (0)	0.109	0.109	-

Table E.17: Experiment 7: Team 2 Agent Metrics at 50th Training Generation

Metric	**Miner**$_1$ μ (σ)	**Vaulter**$_2$ μ (σ)	**Scout**$_3$ μ (σ)	$\mathbf{p}_{(1,2)}$	$\mathbf{p}_{(1,3)}$	$\mathbf{p}_{(2,3)}$
Flag pickups	0.21 (0.24)	0.51 (0.17)	0.04 (0.05)	0.081	0.246	0.004
Flag captures	0.1 (0.1)	0.19 (0.13)	0.01 (0.01)	0.278	0.132	0.04
Tag count	16.12 (8.73)	27.17 (4.75)	15.53 (8.67)	0.066	0.926	0.055
Respawn tag count	1.65 (0.91)	0.61 (0.37)	1.74 (0.98)	0.084	0.9	0.081
Flag dispossessions	0.92 (0.51)	0.41 (0.22)	1.21 (0.65)	0.118	0.509	0.069
Steps defending zone	115.95 (54.21)	149.42 (18.19)	132.01 (64.69)	0.296	0.714	0.628
Steps attacking zone	7.42 (7.62)	17.83 (4.17)	1.69 (2.54)	0.052	0.215	$<$0.001
Steps adj. teammate	374.37 (35.27)	341.59 (12.46)	378.96 (30.09)	0.14	0.848	0.067
Steps adj. opponent	20.03 (10.79)	35.11 (5.94)	19.23 (10.71)	0.048	0.919	0.039
Blocks mined	10.43 (5.5)	0 (0)	0 (0)	0.019	0.019	-
Blocks laid	10.2 (5.42)	0 (0)	0 (0)	0.02	0.02	-
Blocks laid dist. from own flag	39.35 (20.74)	0 (0)	0 (0)	0.019	0.019	-
Blocks laid dist. from opp flag	63.72 (33.66)	0 (0)	0 (0)	0.019	0.019	-

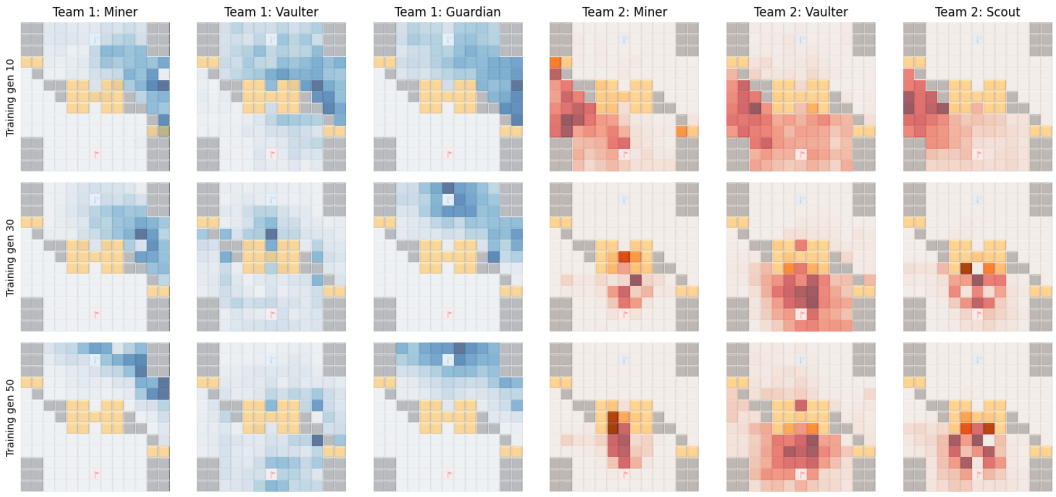

Figure E.21: Experiment 7: Agent Visitation Maps

E.8 Experiment 8: Gridlocked

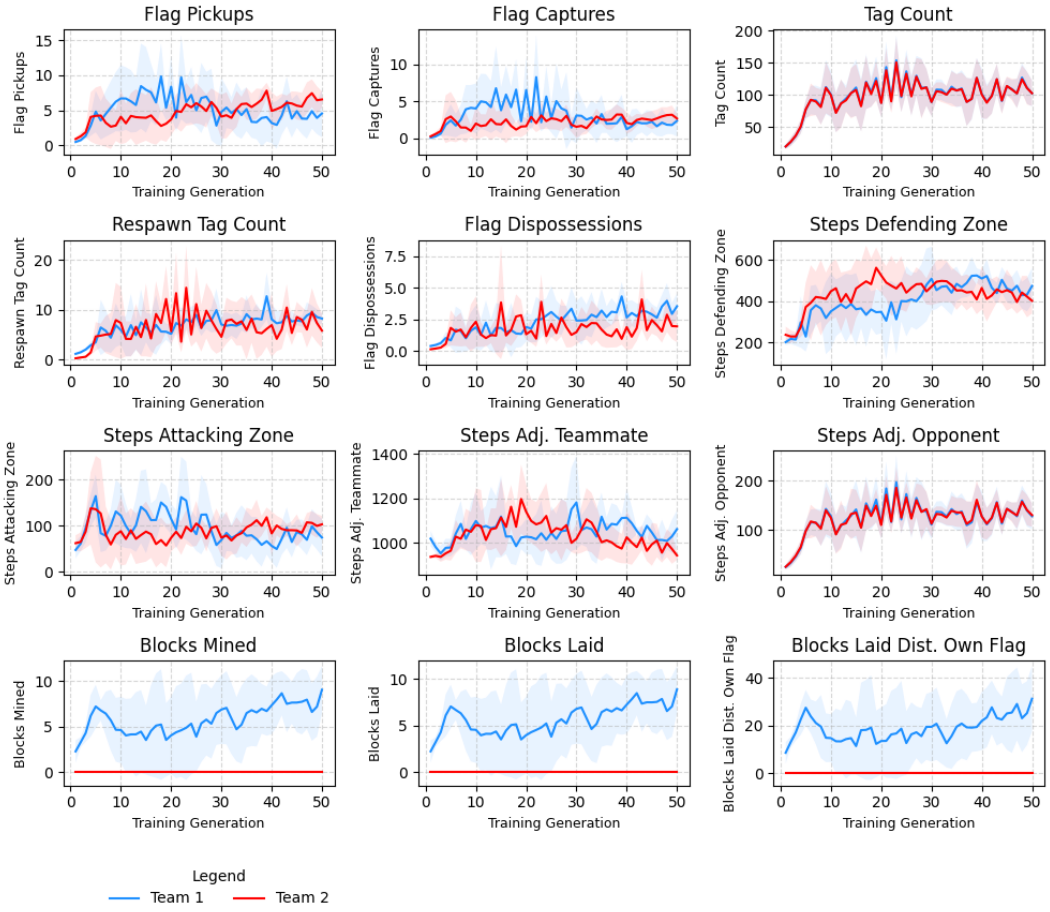

Figure E.22: Experiment 8: Team Metrics

Table E.18: Experiment 8: Team Metrics at 50th Training Generation

Metric	**Team 1** μ (σ)	**Team 2** μ (σ)	**p-value**
Flag pickups	4.57 (3.38)	6.56 (1.14)	0.316
Flag captures	2.36 (2.14)	2.66 (0.67)	0.8
Tag count	102.32 (20.51)	102.51 (19.7)	0.99
Respawn tag count	8.24 (0.97)	5.83 (3.44)	0.241
Flag dispossessions	3.55 (1.02)	1.95 (1.27)	0.087
Steps defending zone	474.07 (86.95)	402.08 (61.59)	0.218
Steps attacking zone	74.13 (40.42)	102.83 (18.25)	0.247
Steps adj. teammate	1062.75 (84.29)	944.4 (19.57)	0.047
Steps adj. opponent	128.16 (26.04)	130.49 (23.67)	0.898
Blocks mined	9.04 (2.58)	0 (0)	0.002
Blocks laid	8.88 (2.49)	0 (0)	0.002
Blocks laid dist. from own flag	31.29 (13.61)	0 (0)	0.01
Blocks laid dist. from opp flag	49.07 (11.64)	0 (0)	0.001

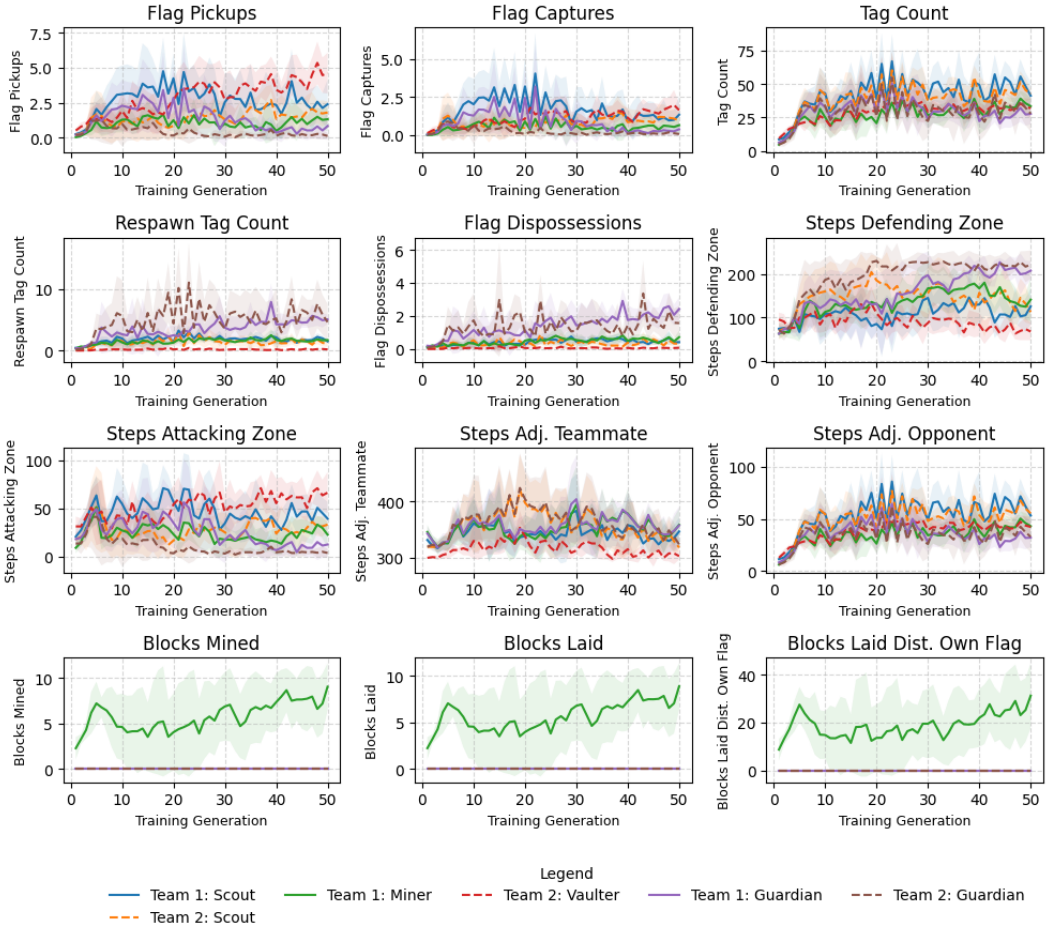

Figure E.23: Experiment 8: Agent Metrics

Table E.19: Experiment 8: Team 1 Agent Metrics at 50th Training Generation

Metric	**Scout**$_1$ μ (σ)	**Miner**$_2$ μ (σ)	**Guardian**$_3$ μ (σ)	$\mathbf{p}_{(1,2)}$	$\mathbf{p}_{(1,3)}$	$\mathbf{p}_{(2,3)}$
Flag pickups	2.4 (1.76)	1.33 (0.84)	0.84 (0.98)	0.315	0.17	0.471
Flag captures	1.33 (1.23)	0.65 (0.5)	0.38 (0.49)	0.344	0.205	0.465
Tag count	41.23 (10.25)	33.36 (5.72)	27.73 (7.59)	0.226	0.07	0.272
Respawn tag count	1.49 (0.22)	1.7 (0.3)	5.05 (0.75)	0.298	<0.001	<0.001
Flag dispossessions	0.41 (0.14)	0.71 (0.4)	2.42 (0.69)	0.218	0.004	0.004
Steps defending zone	125.68 (40.42)	141.08 (49.06)	207.31 (21.51)	0.641	0.012	0.052
Steps attacking zone	39.23 (17.36)	22.54 (13.13)	12.36 (13.58)	0.167	0.042	0.312
Steps adj. teammate	346.67 (24.56)	358.52 (29.56)	357.57 (32.94)	0.555	0.611	0.967
Steps adj. opponent	53.19 (13.14)	42.99 (7.95)	31.98 (8.96)	0.228	0.032	0.104
Blocks mined	0 (0)	9.04 (2.58)	0 (0)	0.002	-	0.002
Blocks laid	0 (0)	8.88 (2.49)	0 (0)	0.002	-	0.002
Blocks laid dist. from own flag	0 (0)	31.29 (13.61)	0 (0)	0.01	-	0.01
Blocks laid dist. from opp flag	0 (0)	49.07 (11.64)	0 (0)	0.001	-	0.001

Table E.20: Experiment 8: Team 2 Agent Metrics at 50th Training Generation

Metric	**Scout**$_1$ μ (σ)	**Vaulter**$_2$ μ (σ)	**Guardian**$_3$ μ (σ)	$\mathbf{p}_{(1,2)}$	$\mathbf{p}_{(1,3)}$	$\mathbf{p}_{(2,3)}$
Flag pickups	1.81 (0.79)	4.62 (1.45)	0.13 (0.21)	0.014	0.012	0.003
Flag captures	0.96 (0.51)	1.63 (0.47)	0.07 (0.1)	0.086	0.023	0.002
Tag count	42.62 (3.97)	32.02 (7.09)	27.87 (13.21)	0.039	0.089	0.599
Respawn tag count	1.03 (0.43)	0.19 (0.13)	4.61 (2.93)	0.015	0.07	0.039
Flag dispossessions	0.31 (0.21)	0.07 (0.06)	1.57 (1.04)	0.08	0.073	0.046
Steps defending zone	112.83 (27.14)	68.16 (19.43)	221.09 (32.25)	0.031	0.001	$<$0.001
Steps attacking zone	33.01 (6.01)	66.28 (21.34)	3.55 (4.69)	0.033	$<$0.001	0.003
Steps adj. teammate	317.91 (7.16)	302.69 (6.35)	323.8 (9)	0.013	0.337	0.006
Steps adj. opponent	55.38 (4.88)	42.29 (9.37)	32.81 (15.07)	0.048	0.037	0.322

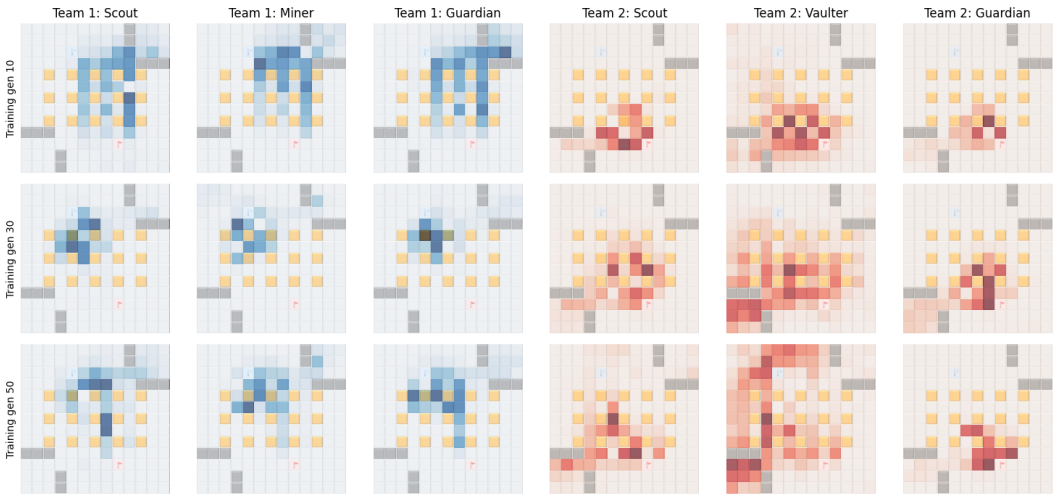

Figure E.24: Experiment 8: Agent Visitation Maps

E.9 Experiment 9: Arena

Note that for this experiment symmetrical teams are analysed using pure self-play, where one policy network is trained against its previous version. Therefore, the metrics reported are slightly different compared with other experiments. For instance, agent metrics and visitation maps for only a single team are plotted and statistical testing is not conducted between teams.

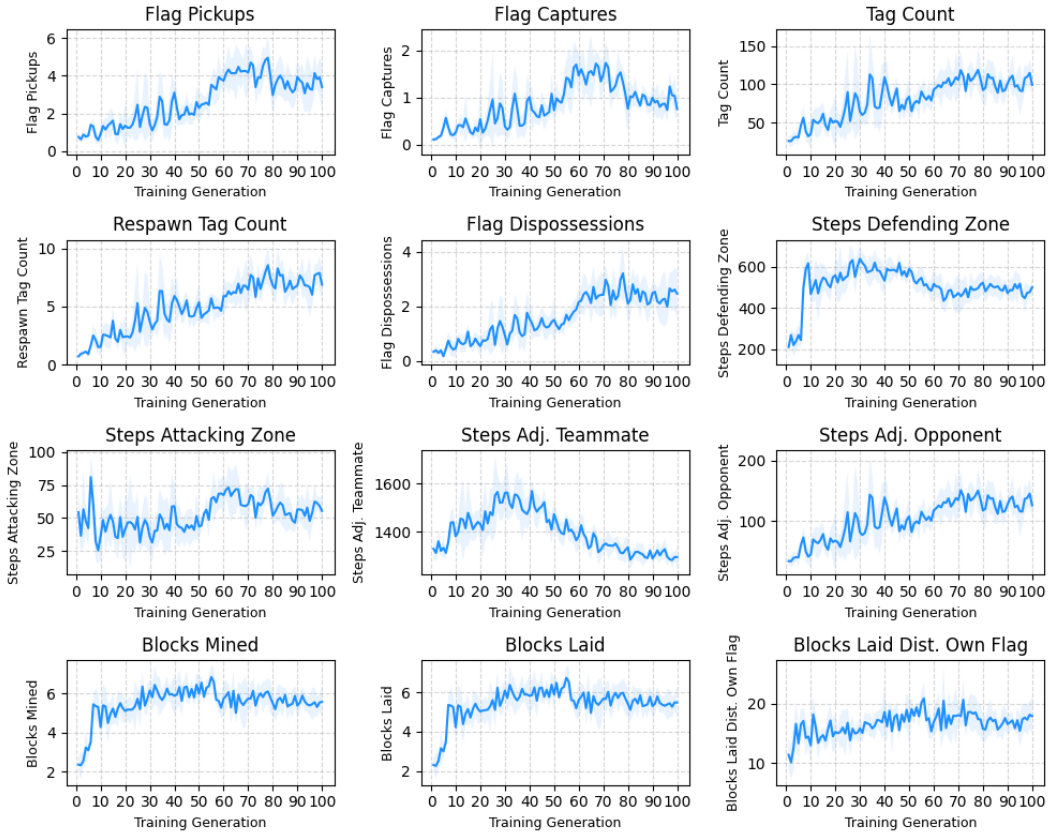

Figure E.25: Experiment 9: Team Metrics

In contrast to previous experiments, agent metrics and p-values are presented in two separate tables, namely table E.21 and E.22.

Table E.21: Experiment 9: Team 1 Agent Metrics at 50th Training Generation

Metric	**Guardian**$_1$ μ (σ)	**Vaulter**$_2$ μ (σ)	**Miner**$_3$ μ (σ)	**Scout**$_4$ μ (σ)
Flag pickups	0.13 (0.06)	2.6 (1.15)	0.18 (0.05)	0.5 (0.18)
Flag captures	0.05 (0.04)	0.53 (0.22)	0.06 (0.03)	0.11 (0.03)
Tag count	20.91 (1.79)	32.23 (4.63)	21.41 (4.76)	24.76 (4.44)
Respawn tag count	4.05 (0.79)	0.16 (0.02)	1.41 (0.41)	1.27 (0.28)
Flag dispossessions	1.33 (0.47)	0.07 (0.05)	0.57 (0.25)	0.49 (0.26)
Steps defending zone	191.95 (9.61)	46.37 (15.15)	129.88 (11.95)	132.53 (14.94)
Steps attacking zone	1.96 (0.78)	39.21 (11.75)	4.97 (1.15)	9.19 (1.96)
Steps adj. teammate	324.29 (15.99)	291.63 (7.4)	338.63 (13.61)	340.49 (12.93)
Steps adj. opponent	23.34 (2.21)	42.73 (6.25)	27.17 (6.03)	32.57 (5.88)
Blocks mined	0 (0)	0 (0)	5.57 (0.81)	0 (0)
Blocks laid	0 (0)	0 (0)	5.48 (0.8)	0 (0)
Blocks laid dist. from own flag	0 (0)	0 (0)	17.87 (3)	0 (0)
Blocks laid dist. from opp flag	0 (0)	0 (0)	40.31 (5.55)	0 (0)

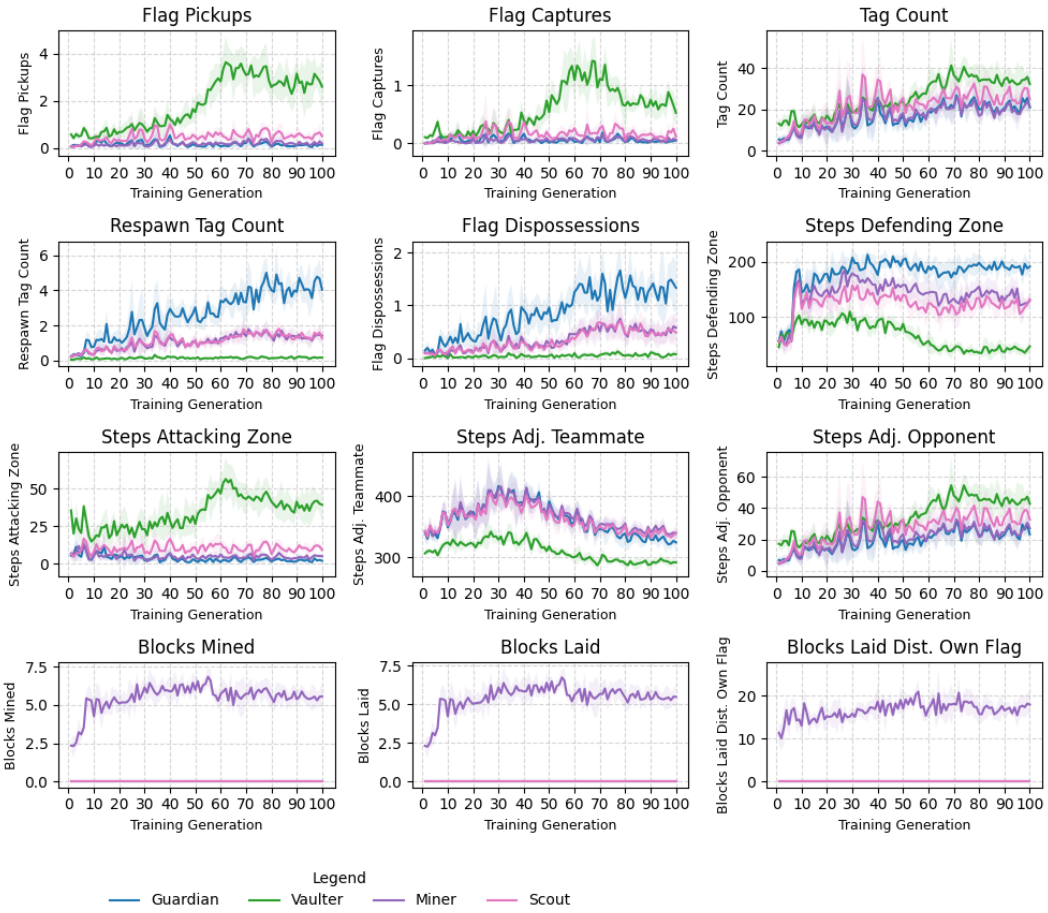

Figure E.26: Experiment 9: Agent Metrics

Table E.22: Experiment 9: Team 1 Agent Metric P-Values at 50th Training Generation

Metric	$\mathbf{p}_{(1,2)}$	$\mathbf{p}_{(1,3)}$	$\mathbf{p}_{(1,4)}$	$\mathbf{p}_{(2,3)}$	$\mathbf{p}_{(2,4)}$	$\mathbf{p}_{(3,4)}$
Flag pickups	0.013	0.184	0.011	0.014	0.021	0.02
Flag captures	0.012	0.803	0.052	0.013	0.019	0.054
Tag count	0.006	0.85	0.165	0.012	0.048	0.334
Respawn tag count	0.001	0.001	0.001	0.003	0.001	0.59
Flag dispossessions	0.005	0.027	0.018	0.015	0.03	0.64
Steps defending zone	<0.001	<0.001	<0.001	<0.001	<0.001	0.789
Steps attacking zone	0.003	0.003	0.001	0.004	0.006	0.009
Steps adj. teammate	0.011	0.21	0.155	0.001	<0.001	0.848
Steps adj. opponent	0.002	0.286	0.031	0.007	0.046	0.235
Blocks mined	-	<0.001	-	<0.001	-	<0.001
Blocks laid	-	<0.001	-	<0.001	-	<0.001
Blocks laid dist. from own flag	-	<0.001	-	<0.001	-	<0.001
Blocks laid dist. from opp flag	-	<0.001	-	<0.001	-	<0.001

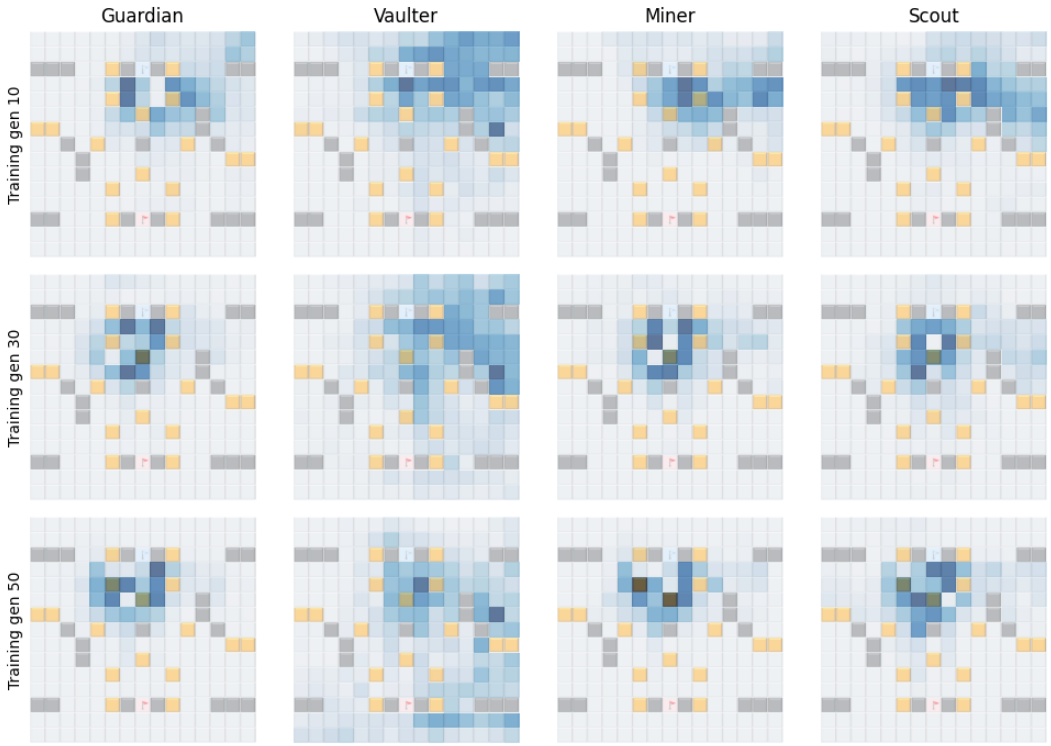

Figure E.27: Experiment 9: Agent Visitation Maps