Multi-agent reinforcement learning for radar waveform design

Radu Gaghi

MULTI-AGENT REINFORCEMENT LEARNING FOR RADAR WAVEFORM DESIGN

Thesis

to obtain the degree of Master in Computer Science with Specialization in Data Science and Technology at Delft University of Technology, to be publicly defended on 19.07.2024

by

Radu GAGHI

Multimedia Computing Group, Faculty of Electrical Engineering, Mathematics and Computer Science (Faculteit Elektrotechniek, Wiskunde en Informatica), Delft University of Technology, Delft, The Netherlands

Thesis committee:

Chair: prof. dr. E. Isufi
Daily supervisor: dr. M. A. Coutiño-Minguez
Committee member: prof. dr. Francesco Fioranelli

CONTENTS

Preface

My life at TU Delft has been the most stressful, challenging, and beautiful part of my journey thus far. My final trial is writing this preface, which is proving to be harder than I expected. I will face this final tribulation with the same stubbornness I faced my exams with and write it nonetheless.

I would like to thank my family for their unwavering support throughout my studies. Without them, I could not have accomplished any of my dreams: studying abroad, pursuing a master's degree in AI, and being able to enjoy my time while doing so. You were the first to believe in me, which inspired me to believe in myself. I could have failed many times along the way, but because of you, I did not.

I would also like to thank my girlfriend, Elena, for being there for me throughout the highs and lows of my thesis, of which there were many. Your unwavering optimism and love have made everything so much easier, and I cannot wait to be there for you in the same way when you will be writing your own thesis a year or so from now.

Thank you, Elvin, for providing invaluable feedback and pushing me to better myself along the way. Your help has made me a better researcher, and for that, I am grateful. In his infinite wisdom, Elvin introduced me to Mario, without whom this work would not have been possible. I could not have asked for a better supervisor. You always made time for me, even when I asked stupid questions. Your cheery attitude and encyclopedic knowledge were invaluable to me and made my time at TNO all the more enjoyable. I hope we can remain friends long after my thesis is concluded and perhaps even work together again someday. I would also like to thank a few people at TNO, without whom this work would not have been the same: Arne Theil, Marc Beekman, Bas Jacobs, Pepijn Cox, and Wim Rossum. You had no obligation to help me, but you took your time and patiently answered any questions I had. For that, I am grateful.

Finally, I would like to thank my good friends and cohabitants Sebastian, Alex, and Octav, for listening to me rant about my thesis every day of the week. Knowing you were going through the same thing helped me persevere, and I am glad we faced this enormous, wonderful thing called student life together.

I hope that readers will find this thesis interesting and that it helps someone along the way. If it does, I will be even more joyous than I am now, as my work comes to an end. It was incredibly interesting and enjoyable to do this piece of research, and I hope that readers will excuse my shortcomings when it comes to academic writing. Truthfully, I enjoyed that part the least.

Radu Gaghi
Delft, July 2024

SUMMARY

This thesis explores the application of reinforcement learning (RL) to the problem of radar waveform optimization, with a particular focus on multi-agent reinforcement learning (MARL). Radar technology is crucial in various fields such as aviation, maritime navigation, and defense, but it faces challenges like interference, clutter, and the need for high resolution and accuracy. Cognitive radar, which adapts to environmental changes in real time, offers a promising solution. This research investigates how MARL can be effectively utilized to optimize radar waveforms and whether integrating domain knowledge can enhance its performance.

The thesis provides a comprehensive overview of radar technology, reinforcement learning, and graph machine learning. Key concepts include Pulse Doppler Radar, Markov Decision Processes (MDPs), and Graph Neural Networks (GNNs). It also reviews essential reinforcement learning algorithms like Proximal Policy Optimization (PPO), Independent Proximal Policy Optimization (IPPO), and Multi-Agent PPO (MAPPO).

The survey covers classical methods for radar waveform optimization, such as mathematical optimizers and genetic algorithms, as well as recent approaches using neural networks and reinforcement learning. It identifies gaps in the literature, particularly the underexplored potential of MARL in optimizing radar waveforms.

The radar waveform optimization problem is framed within the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework. The thesis defines the radar environment, agents' observations, and actions, along with multiple objectives represented as reward functions. We experiment with different architectures, including decentralized actors with a centralized critic. The centralized critic, which has access to global state information, helps stabilize the learning process and mitigate the problems of non-stationarity and credit assignment. The use of GNNs as a centralized critic is proposed to leverage graph data sparsity, enhancing scalability.

The proposed models are trained and tested in a radar-tracking scenario. Their performance is evaluated in terms of Pareto optimality and optimization times, comparing the IAC and IACC models against traditional genetic algorithms and the single-agent baseline. The results show that both IAC and IACC models outperform traditional methods in terms of probability of detection, waveform duration, and optimization speed.

The findings highlight the effectiveness of MARL approaches in optimizing radar waveforms. Centralized critics improve robustness and coordination among agents, but the choice of architecture significantly impacts overall performance. While GNNs offer potential advantages in scalability, their integration of domain knowledge did not yield significant improvements in this study. The thesis lays a foundation for the use of MARL and GNNs in radar waveform optimization and encourages further exploration into more advanced techniques. Future work should consider increasing the number of agents, exploring variable burst budgets, and refining the integration of domain knowledge to enhance performance further.

1

Introduction

Radars, an acronym for Radio Detection and Ranging, are systems used for detecting and locating objects by transmitting and receiving electromagnetic waves. The primary components of a radar system include a transmitter, which generates electromagnetic waves, an antenna to direct these waves toward a target, and a receiver to capture the waves reflected back from the target. This reflected signal is then processed to extract information such as the distance, speed, and characteristics of the target. Radars hold a significant place in modern technology due to their ability to perform in various environmental conditions, including adverse weather and low visibility scenarios where optical and infrared systems may fail. Their capability to detect and track objects over long distances makes them indispensable in numerous fields such as aviation, maritime navigation, weather forecasting, and defense. The applications of radar technology are diverse, ranging from air traffic control, where radars manage and monitor aircraft movement, to military operations, where they are used for surveillance, target acquisition, and missile guidance. Additionally, radars are essential in meteorology for monitoring weather patterns and predicting severe weather conditions. In automotive industries, radar systems are integrated into vehicles for collision avoidance and adaptive cruise control.

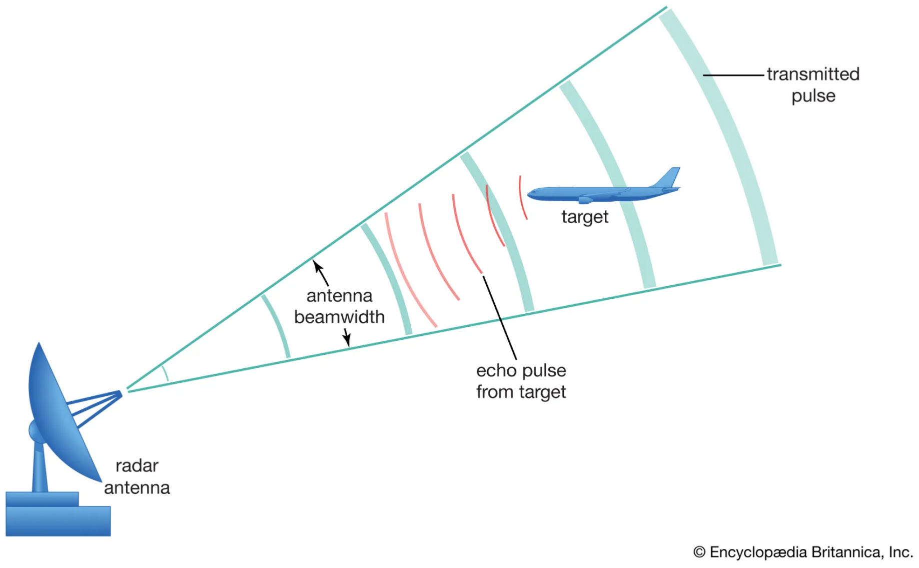
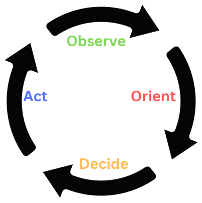


Figure 1.1: Radar System

Despite their versatility, radars face several challenges. These include interference from other electromagnetic sources, clutter from non-target objects, and the need for high resolution and accuracy. Additionally, modern radars must adapt to dynamic environments and emerging threats, requiring advanced signal processing and optimization techniques. The challenge is further compounded by the need to balance performance with constraints such as power consumption and computational efficiency.

Cognitive radar represents an advanced approach aimed at overcoming these challenges through increased adaptability and intelligence.



Cognitive radars are designed to perceive their environment and adjust their operational parameters in real-time to optimize performance. This concept of cognitive radar aligns closely with John Boyd's OODA loop (Observe, Orient, Decide, Act), a decision-making framework originally developed for military strategy [1]. By implementing the OODA loop, cognitive radar systems dynamically respond to environmental changes, and a crucial aspect of this adaptability is radar waveform optimization, which involves designing waveforms that maximize detection and identification capabilities under varying conditions.

Waveform design and optimization are important aspects of enhancing radar performance. It involves the selection of radar waveform parameters that maximize detection and identification capabilities under varying conditions. Mathematical optimization methods have been widely applied in the field of radar waveform design. These methods involve formulating the waveform optimization problem as a mathematical model, often as a linear or nonlinear programming problem. The objective is to maximize or minimize a specific performance metric, such as signal-to-noise ratio (SNR), resolution, or detection probability, subject to various constraints. Figure 1.2 shows the probability of detection at various ranges and velocities for a waveform optimized for heavy rain and wind. Areas with a low probability of detection can be seen before the 10km mark, caused by the rain. The radar is effectively blind in these regions, and waveform optimization is necessary to reduce these blind spots as much as possible.

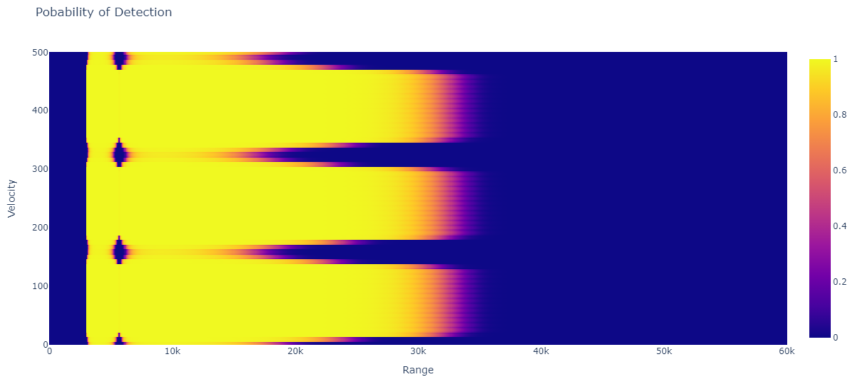


Figure 1.2: Probability of detection over tracking area for an optimized waveform

Common optimization techniques include gradient-based methods[2], convex optimization [3], and metaheuristic algorithms such as simulated annealing [4] and particle swarm optimization [5]. They can suffer from issues like slow convergence, computational complexity, and sensitivity to initial conditions and parameters. These methods often struggle with scalability and can get trapped in local minima, leading to suboptimal solutions. Additionally, they may require careful tuning and handling of constraints, and their performance can be affected by noise and the nonconvex nature of the problem landscape. These methods also need to be reapplied to each problem instance, making real-time adaptivity hard to achieve.

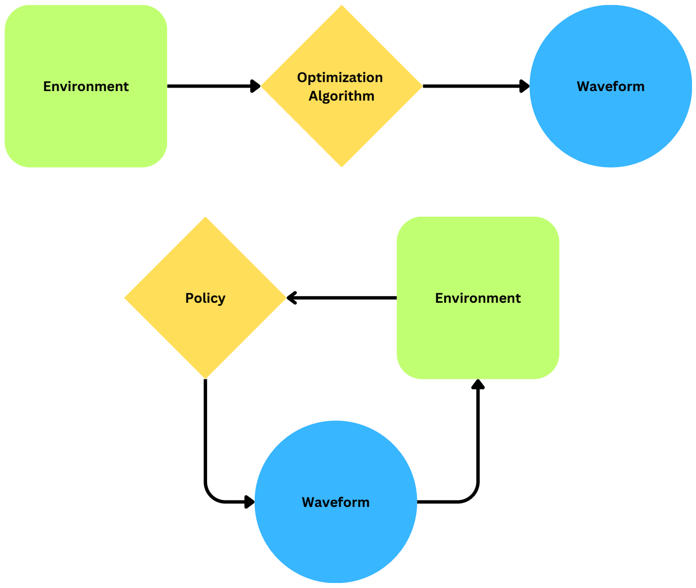


Figure 1.3: Optimization pipeline for classical algorithms compared to a policy. The optimization algorithm produces a waveform based on the state of the environment and the predefined constraints. A policy is more similar to the OODA Loop, where we can keep querying the policy for a waveform as the environment changes, once the policy is trained, without the need for re-optimization.

Given these limitations, neural networks (NN) have emerged as a powerful alternative for radar waveform optimization [6]. By leveraging large amounts of data, NNs can be trained to predict optimal waveform parameters in real-time, thereby overcoming some of the scalability and adaptivity issues associated with traditional optimization methods. Neural networks offer several advantages in this context. First, they can generalize from the training data to new, unseen conditions, providing robust performance across a variety of scenarios. This generalization capability allows for the creation of a model that can be applied broadly, reducing the need for constant re-optimization (Figure 1.3). Second, once trained, neural networks can make rapid inferences, enabling real-time adjustments to radar waveforms. However, one key disadvantage of supervised learning using NNs is their reliance on large labeled datasets for training. Supervised learning with NNs requires a substantial amount of data where the optimal waveform parameters are already known. Collecting and labeling such data can be time-consuming and expensive, especially in complex radar environments where the optimal waveforms may not be easily discernible or may vary significantly across different scenarios. Another disadvantage of NNs is their potential for overfitting, particularly when the training data does not adequately represent all possible operating conditions. Overfitting occurs when the model learns to perform well on the training data but fails to generalize to new, unseen data. This can lead to suboptimal performance in real-world scenarios that differ from the training conditions.

In contrast, reinforcement learning does not require labeled data in the same way. Instead, RL algorithms learn optimal strategies through interaction with the environment, receiving feedback in the form of rewards or penalties. This ability to learn from experience allows RL methods to adaptively discover optimal solutions even in the absence of pre-labeled data. This is particularly advantageous in dynamic and uncertain environments where the optimal solution is not static or known beforehand. Furthermore, radar waveform optimization is a multi-objective problem, which can be expressed as a multi-agent reinforcement learning problem. Each objective or figure of merit can be assigned to a different agent, decomposing a complex problem into manageable subproblems. This allows the policy of each agent to tackle a single aspect of the problem, which should be easier to learn. In the current literature, RL approaches are not very widespread and are only applied to very specific radar problems, such as spectral notching [7]. The work of [8] introduces the first Pulse-Doppler single-agent RL approach, which has inspired this thesis. To the author's knowledge, there are no MARL approaches in the radar waveform optimization literature. To address this gap, we pose the following research question:

RQ) **How can multi-agent reinforcement learning (MARL) be effectively utilized to optimize radar waveforms?**

In particular, we focus on answering:

(a) How can the multi-objective radar waveform optimization problem be modeled as a MARL problem?

(b) How can radar domain knowledge be integrated to improve the performance of the agents?

(c) What are the possible architectures that allow scalability in the number of radar waveform parameters?

Each architecture is tested using a radar tracking scenario, where the agents need to follow a moving target in difficult environmental conditions. Firstly, we focus on the challenges of multi-agent reinforcement learning such as the credit assignment problem and non-stationarity by introducing two architectures, using a decentralized critic and a centralized critic, respectively. We then build on the centralized critic architecture by swapping its policy with a GNN. This addresses the common issue of scalability in MARL and presents a way of incorporating radar domain knowledge into the graph-building process.

This thesis explores the application of reinforcement learning to radar waveform optimization, with a focus on multi-agent reinforcement learning. Chapter 2 covers the fundamentals of radar systems, including Pulse Doppler Radar, reinforcement learning, and graph learning. Chapter 3 delves into current radar waveform optimization techniques, from classical methods to supervised and reinforcement learning.

For readers already familiar with these topics, it may be beneficial to proceed directly to Chapter 4 and beyond, where the core research begins. These chapters detail the framing of the radar waveform optimization problem within the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework, and the introduction of centralized critics using Graph Neural Networks (GNNs). Chapter 5 details the various proposed architectures, while Chapter 6 evaluates the proposed models in radar-tracking scenarios. The findings demonstrate the effectiveness of MARL in optimizing radar waveforms, highlighting advancements over traditional methods and single-agent RL, providing a foundation for future research.

2 Background

Chapter 2 provides relevant information regarding the rest of the thesis. In Section 2.1, the basics of radar are introduced, focusing on Pulse-Doppler technology. The main parameters and their trade-offs are explained. Section 2.2 introduces the concept of reinforcement learning and its extension to multiple agents, covering fundamental concepts such as Markov Decision Processes (MDPs), Partially Observable Markov Decision Processes (POMDPs), and Decentralized POMDPs (Dec-POMDPs). It also details key reinforcement learning algorithms, including Proximal Policy Optimization (PPO) and its multi-agent variants, MAPPO and IPPO. Finally, Section 2.3 talks about graphs and how they are defined, as well as graph shift operators and their use in graph convolutions. This section also introduces Graph Neural Networks (GNNs) and their application in modeling complex interactions within graph-structured data.

2.1. RADAR

2.1.1. FUNDAMENTAL CONCEPTS

Radar systems operate by transmitting electromagnetic (EM) waves into a region of space to detect and analyze reflected signals from various targets. Pulse Radar emits short bursts of radiofrequency energy and measures the time taken for signals to return from targets. Although there are various types of radar systems, this thesis specifically focuses on Pulse-Doppler radar. However, the concepts and methods introduced here are designed to be easily adapted to other types of radar systems as well. Therefore, the framework presented in this thesis is general and applicable across different radar technologies. In this work, we will tackle the problem of radar waveform optimization for tracking radars.

A burst refers to a sequence of rapid, consecutive radar pulses sharing the same carrier frequency transmitted at a specific repetition frequency over a short period. In our specific case, each pulse in a burst is linearly frequency modulated, enabling the use of pulse compression (PC) techniques. By spreading the signal over a longer time period and then compressing it on reception, the energy of the pulse is concentrated. PC enhances signal-to-noise ratio and range resolution [9].

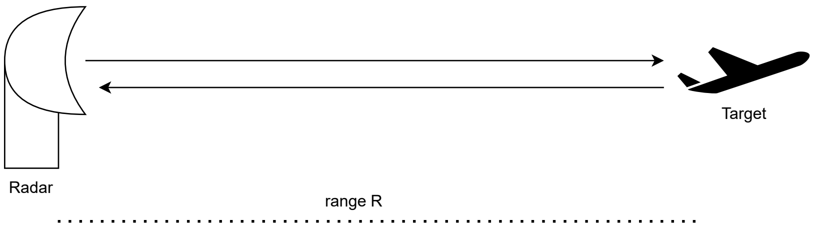


Figure 2.1: Range measurement using the two-way time to the target

In pulse-Doppler radar systems, range measurement involves determining the time delay between the transmission of a radar pulse and the reception of its echo from a target (Figure 2.1. This process, known as pulse-ranging, is essential for calculating the distance between the radar system and the target object. The formula for calculating range in Pulse-Doppler radar is presented in Equation 2.1, where R represents the range to the target, c is the speed of light, and τ is the two-way time.

$$R = \frac{c \cdot \tau}{2} \tag{2.1}$$

The Doppler effect occurs when there is relative motion between a wave source and an observer. In radar, the Doppler effect helps determine the velocity of a target by analyzing the frequency shift of reflected radar waves. The Doppler shift (f_d) is approximated using Equation 2.2, where v represents the target's radial velocity, f_c is the carrier frequency, λ is the transmitted radar wavelength, and c is the speed of light in a vacuum.

$$f_d = \frac{2v}{c} f_c = \frac{2v}{\lambda} \tag{2.2}$$

Two significant ambiguities can affect target measurements: range ambiguity and Doppler ambiguity (Figures 2.3 and 2.4).

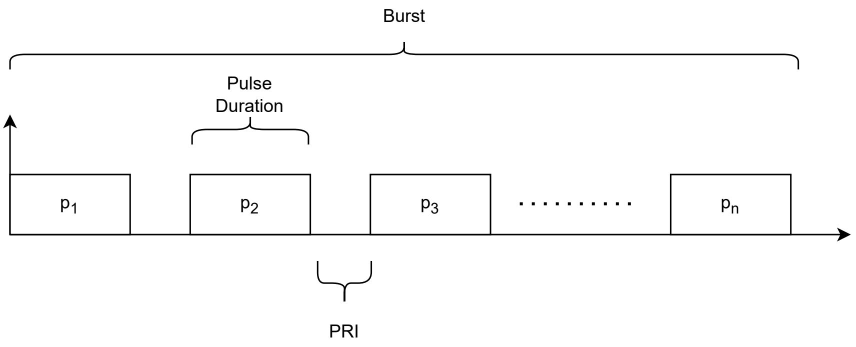


Figure 2.2: Example of a radar burst. Note that while the PRI is typically much longer than the pulse duration, it is depicted this way here for illustrative purposes.

Range ambiguity occurs when an echo of a previous pulse is received after a consecutive pulse has already been sent. If the time it takes for the radar pulse to return from a distant target exceeds the time between consecutive pulses, the radar may misinterpret the range, resulting in ambiguous measurements. The formula for range ambiguity (R_a) in Pulse-Doppler radar is given by Equation 2.3.

$$R_a = \frac{c}{2 \cdot PRF} \tag{2.3}$$

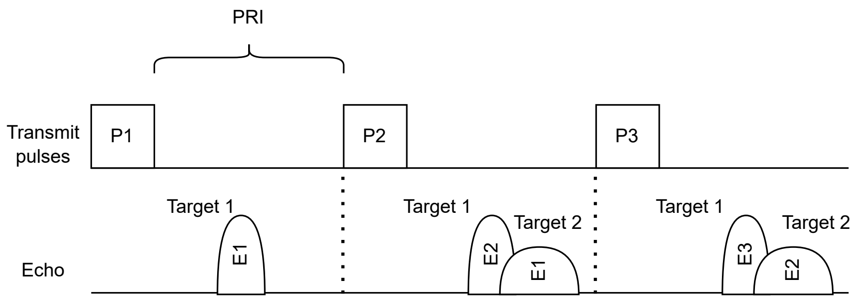


Figure 2.3: Range ambiguity: the echo from the 2nd target arrives only after the 2nd pulse is sent, thus appearing at a shorter range than in reality

Doppler ambiguity arises when the radar observes targets with velocities that result in a frequency shift equal to or exceeding half the PRF. In such cases, the radar may interpret the Doppler shift incorrectly, leading to velocity measurement ambiguities. The maximum unambiguous Doppler shift that can be detected is half the PRF, as seen in Equation 2.4.

$$f_d^{max} = \pm\frac{PRF}{2} \tag{2.4}$$

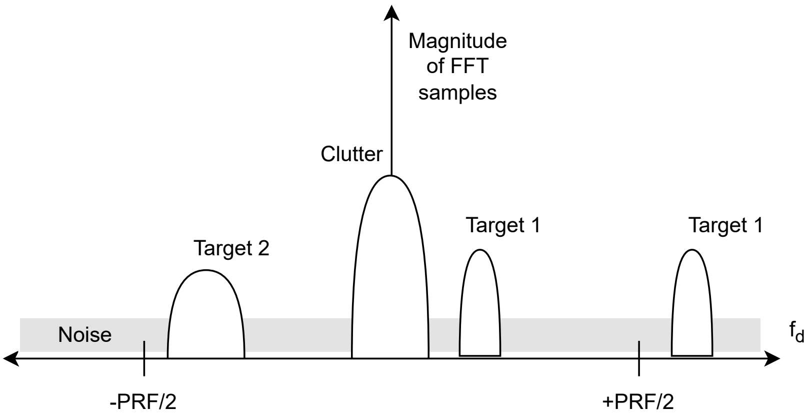


Figure 2.4: Doppler ambiguity: targets at frequencies larger than half the PRF are aliased within this range

Range resolution in Pulse Doppler radar refers to the ability to distinguish between two targets at different distances along the radar's line of sight. Range resolution depends on the bandwidth B of the transmitted radar pulse. Frequency modulation allows us to distinguish between potentially overlapping pulses, so a higher bandwidth provides a better range resolution, as shown in Equation 2.5.

$$\Delta R = \frac{c}{2 \times B} \tag{2.5}$$

Doppler resolution, on the other hand, refers to the ability to differentiate between targets moving at different velocities. Larger PRI results in better Doppler resolution, as shown in Equation 2.6, where Δf_d is the Doppler resolution and N is the number of pulses in the burst.

$$\Delta f_d \propto \frac{PRF}{N} \tag{2.6}$$

Pulse Doppler radar systems involve trade-offs between maximum unambiguous range and velocity measurements, as well as between Doppler and range resolution. These trade-offs stem from inherent limitations in radar system design and operation The pulse repetition frequency (PRF) significantly influences both maximum unambiguous range and velocity measurements. A higher PRI reduces the likelihood of range ambiguity by allowing less frequent pulse transmissions. However, it also reduces the PRF, limiting the unambiguous Doppler velocity. To address this trade-off, radar systems often employ techniques like staggered PRF (Figure 2.5). This technique involves transmitting pulses at different intervals rather than at a constant rate, making it easier to resolve ambiguities[10].

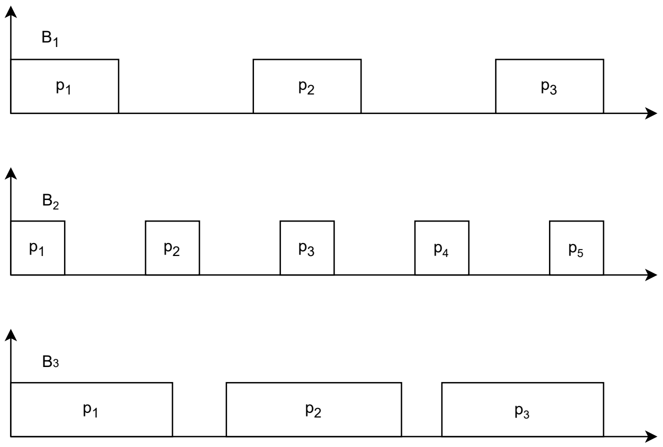
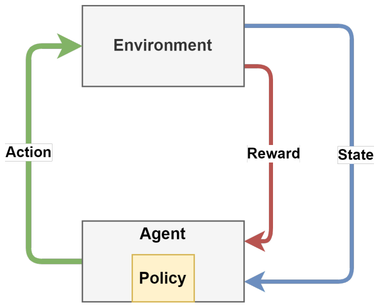


Figure 2.5: PRF Staggering: Three bursts with different PRFs. Note that while the PRI is typically much longer than the pulse duration, it is depicted this way here for illustrative purposes.

2.2. REINFORCEMENT LEARNING

In RL, the agent and the environment play central roles. The agent, an entity capable of learning, observes and acts within the environment, which encompasses everything with which the agent interacts. This interaction is modeled as a Markov Decision Process (MDP), a framework for modeling decision-making where outcomes are partly random and partly under the control of the agent. The core of an MDP involves states, actions, and rewards. At each time step, the agent observes a state from the state space of the environment and selects an action from its action space. In response to the action, the environment presents a new state and provides a reward to the agent (Figure 2.6). The agent's objective is to learn a policy, a strategy of choosing actions based on states, to maximize the total reward it receives.

Figure 2.6: Reinforcement learning loop



Definition 1. *A finite Markov decision process is a tuple $\langle S, A, T, R \rangle$, where S represents the set of environment states, A denotes the set of agent actions, $T : S \times A \times S \rightarrow$ is the state transition probability function, and $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function.*

The state $s_t \in S$ describes the environment at each time step t. The agent observes the state and selects an action $a_t \in A$. Consequently, the environment transitions to a new state $s_{t+1} \in S$ according to T. Subsequently, the agent receives a reward $r_t \in \mathbb{R}$ determined by the output of the reward function R. This process iterates until the agent reaches a terminal state or satisfies some predefined stopping conditions. One realization of this process is called an episode. However, sometimes the agent cannot directly observe the state of the environment. Instead, it must maintain a sensor model of the environment, known as observations.

Definition 2. *A partially observable Markov decision process (POMDP) is a tuple $\langle S, A, \Omega, T, R \rangle$, where S represents the set of environment states, A denotes the set of agent actions, Ω is the set of observations, $T : S \times A \times S \rightarrow$ is the state transition probability function, and $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function.*

We can extend a POMDP to multiple decentralized agents using a Dec-POMPD.

Definition 3. *A Dec-POMDP is a tuple $\langle S, A_1, ..., A_n, \Omega_1, ..., \Omega_n, T, R_1, ..., R_n \rangle$, where n is the number of agents, S represents the set of environment states, A_i denotes the set of actions for agent i. The joint actions space can be represented as $\boldsymbol{A}$. Ω_i is the set of observations of agent i and $\boldsymbol{\Omega}$ is the set of joint observations. $T : S \times \boldsymbol{A} \times S \rightarrow$ is the state transition probability function, and $R_i : S \times \boldsymbol{A} \times S \rightarrow \mathbb{R}$ are the reward functions of the agents.*

The (joint) expected discounted reward is $G = \sum_{T}^{t=1} \gamma^t r_t$, where γ represents a discount factor and r_t is the global reward at time t. To solve a Dec-POMDP is to find a set of policies $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ that maximize the joint expected discounted reward.

In the context of reinforcement learning, a policy is a strategy or a rule that an agent follows to decide its actions based on the current state of the environment. Formally, a policy is a mapping from states to a probability distribution over actions. The policy guides the agent's behavior in the environment, determining which actions to take in order to maximize cumulative rewards over time.

The state-value function, denoted as $V^{\pi_\theta}(s_t)$, is a measure of the expected cumulative reward starting from a state s_t and following a certain policy π_θ, parameterized by θ. It is given by the expected value with respect to the policy, visible in Equation 2.7. In this case t is the current time instance, γ is the discount factor, influencing the importance of future rewards, and R is the reward.

$$V^{\pi_\theta}(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s_t \right] \tag{2.7}$$

The action-value function, or Q-function, denoted as $Q^{\pi_\theta}(s_t, a_t)$, extends this idea to consider the value of taking a specific action a_t in a given state s_t. It is expressed as:

$$Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s_t, A_t = a_t \right] \tag{2.8}$$

The advantage function is a component that helps in evaluating the relative value of an action compared to the average action in a given state. The advantage function, denoted as $A^{\pi_\theta}(s_t, a_t)$, is defined as the difference between the action-value function $Q^{\pi_\theta}(s_t, a_t)$ and the state-value function $V^{\pi_\theta}(s_t)$. Mathematically, it can be expressed as:

$$A^{\pi_{\theta_i}}(s_t, a_t) = Q^{\pi_{\theta_i}}(s_t, a_t) - V^{\pi_\theta}(s_t) \tag{2.9}$$

The use of the advantage function helps to reduce the variance of policy gradient estimates. By comparing the value of an action to the average value of all possible actions in a given state, the advantage function provides a more stable and centered estimate. This centering helps in obtaining more reliable and efficient updates during the learning process. Usually, the state-value function or the action-value function is estimated by another neural network, called a critic. Critics, in general, reduce variance during training and lead to faster convergence [11].

2.2.1. PROXIMAL POLICY OPTIMIZATION

Proximal Policy Optimization (PPO) is a reinforcement learning algorithm, part of the policy gradient method family [12]. PPO operates by making small, controlled updates to the policy, which helps to maintain stability and prevent drastic changes that could destabilize the learning process. The PPO algorithm functions through the use of a "clipped" objective function, which restricts the extent to which policy updates can deviate from the current policy. This clipping mechanism ensures that the updates are conservative, reducing the risk of significant deviations that could lead to poor performance or instability.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2.10}$$

$$J(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{2.11}$$

where:

- $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t in state s_t under the current policy parameterized by θ.
- $\pi_{\theta_{old}}(a_t|s_t)$ is the probability of taking action a_t in state s_t under the old policy parameterized by θ_{old}.
- $\hat{A}_t$ is the advantage estimate at time step t.
- ϵ is a small hyperparameter that defines the clipping range.

PPO offers a favorable compromise with minimal tuning required, ease of implementation, and efficient sample usage by aiming to find a policy that is only slightly modified from the previous one. Additionally, it is versatile enough to be used in a model-free context, where knowledge of state transitions in the environment is not needed, and it can easily handle both discrete and continuous variables.

2.2.2. MAPPO AND IPPO

Multi-agent learning involves training multiple agents that interact within a shared environment, where agents can either cooperate to achieve a common goal or compete against each other with conflicting objectives. Multi-agent systems can scale to large and complex tasks that would be infeasible for a single agent. Agents can have different objectives, which simplifies the reward mechanism for the agent, instead of expressing multiple objectives as a sum of rewards, as a single agent has no way of telling which reward came from which objective. Stationarity is an important property that guarantees the convergence of many RL algorithms. In Dec-POMDPS each agent assumes the other agents are part of the environment, which violates stationarity due to the transition probabilities changing. The Markov property is not fulfilled so the environment appears non-markovian. This poses a significant challenge to multi-agent learning algorithms. Furthermore, stochasticity in the environment can destabilize the learning process. When the agent needs to distinguish between multiple distinct rewards, given the same action, it cannot know if the difference in reward arises from the actions of the other agents, or stochasticity in the environment. This is known as the credit assignment problem, another common challenge in MARL. The exploration of other agents is also an issue, since the exploration yields low rewards, the other agents cannot tell why the reward was low. If they adapt to this too quickly, the policy can be 'ruined' and the optimal policy is never reached. All of these issues destabilize learning, and a possible way to tackle this is by using centralized critics. Centralized critics are claimed to stabilize the learning process by reducing variance in value function estimates.

Multi-Agent Proximal Policy Optimization (MAPPO) [13] extends the Proximal Policy Optimization (PPO) algorithm to environments with multiple interacting agents, which may either cooperate or compete (Figure 2.7). The goal is to enable each agent to learn optimal policies while considering the interactions with other agents. MAPPO uses shared experience replay buffers, allowing agents to learn from each other's experiences. This facilitates faster and more stable learning by providing a richer set of experiences. During training, MAPPO employs a centralized critic that has access to the global state and actions of all agents. Centralized critics have seen reasonable success in literature [14], and the suggested benefits of using a centralized critic are reduced variance, improved coordination, stability, and training time [15]–[19]. However, [20] shows that a centralized critic does not necessarily enhance cooperation compared to decentralized critics, as both provide the same gradients to decentralized policies in expectation. They find that centralized critics provide more stable value function estimates, but have increased variance in the policy gradients. The authors empirically show that decentralized critics are more robust and do not promote cooperation as intuition would guide us to believe, but also recognize that the choice of architecture depends on the environment.

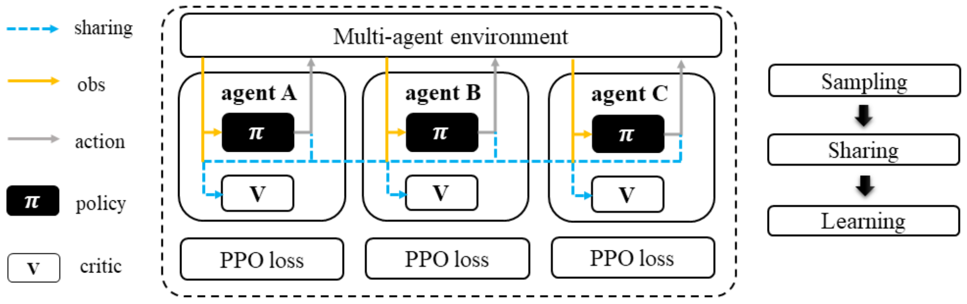


Figure 2.7: A diagram of MAPPO. The agents each learn their own policy, but share the parameters of the critics.

Given the possible disadvantages of MAPPO, we also consider IPPO in this work [21]. Independent Proximal Policy Optimization (IPPO) applies the PPO algorithm independently to each agent in a multi-agent environment (2.8). Unlike MAPPO, IPPO does not explicitly account for interactions among agents during the learning process. This means that IPPO uses decentralized critics, where each agent has their own critic network and global information is not shared between them. IPPO should theoretically have more biased value function estimates, but less variance in the policy gradients. IPPO should also have better scaling with an increased number of agents due to the lack of joint observations in the critic.

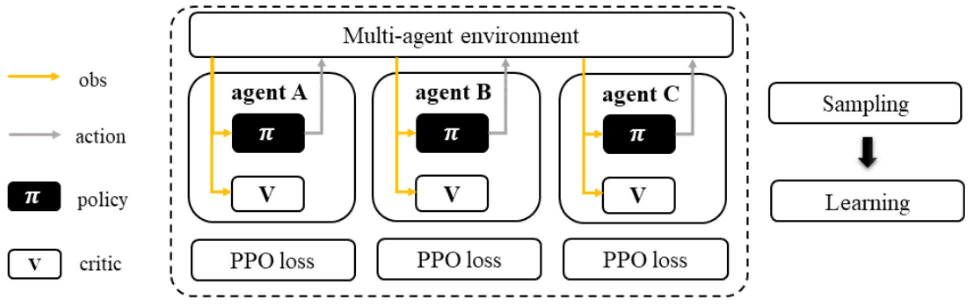


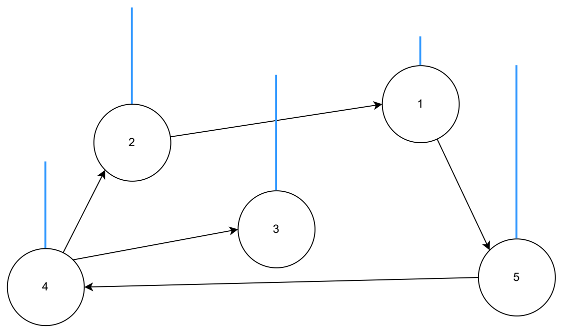
Figure 2.8: A diagram of IPPO. Each agent is a fully independent learner.

2.3. GRAPH MACHINE LEARNING

2.3.1. GRAPH THEORY

In mathematical terms, a graph is an ordered pair $G = (V, E)$ consisting of a set V of vertices (or nodes) of cardinality $|V| = N$ and a set E of edges (or links) of cardinality $E = |M|$ which are two-element subsets of V. Each edge in E is an unordered pair $\{v, w\}$ or an ordered pair (v, w) for an undirected or directed graph, respectively. In the case of an undirected graph, edges lack orientation, implying that $\{v, w\}$ is identical to $\{w, v\}$. Conversely, in a directed graph (or digraph), edges are ordered pairs, meaning (v, w) differs from (w, v). A graph may also be characterized as weighted if a numerical value or weight is assigned to each edge.

Figure 2.9: Directed graph with blue bars denoting the magnitude of the graph signal



A graph signal refers to a set of scalar values $x = [x_1, x_2, x_3, ..., x_N]^T \in \mathbb{R}^N$ corresponding to each vertex in a graph. This concept extends the idea of signal processing, traditionally defined on time or spatial domains, to the domain of graph theory. The concept of a graph signal is particularly useful in the analysis of data that reside on the nodes of a graph, such as sensor networks, social networks, or communication networks, where each node has a value or a set of values associated with it, and the edges represent the relationships or connections between these nodes.

The k-hop neighborhood of a vertex within a graph refers to the set of all vertices that can be reached from the given vertex by traversing at most k edges. For a vertex $v \in V$, its k-hop neighborhood is defined as follows:

$$\mathcal{N}_k(v) = \{u \in V \mid \text{there exists a path from } v \text{ to } u \text{ with at most } k \text{ edges}\}$$

This definition applies to both directed and undirected graphs. In the context of a directed graph, the path must respect the direction of the edges. The k-hop neighborhood includes the vertex itself (since zero edges are needed to reach it) and any other vertices that are within k edge traversals. For example, in a 1-hop neighborhood (also known as the immediate neighborhood or simply the neighborhood), you include all vertices directly connected to the vertex v by a single edge. As k increases, the neighborhood expands to include vertices that are progressively further away from the original vertex in terms of edge hops.

2.3.2. Graph Shift Operator

In graph signal processing, the graph shift operator (GSO) is a fundamental linear operator that plays a role analogous to that of the time shift in classical signal processing. It is used to represent and analyze signals defined on the vertices of a graph. Let $G = (V, E)$ be a graph with vertex set V and edge set E, and let $x \in \mathbb{R}^N$ be a graph signal. The graph shift operator is defined as a matrix S that acts on the graph signal x.

Mathematically, the graph shift operator S is a $|V| \times |V|$ matrix, where $|V|$ is the number of vertices in the graph. The action of S on the graph signal x is given by Equation 2.12, which is a new graph signal, represented by $x^{(1)}$.

$$x^{(1)} = Sx \tag{2.12}$$

The specific structure of S depends on how the notion of "shifting" is defined on the graph, but it generally adheres to the following properties:

1. **Sparsity**: S is often sparse, with non-zero entries corresponding to edges in the graph. This means that $S_{ij} \neq 0$ if and only if there is an edge between vertices i and j, or $i = j$.

2. **Locality**: The action of S on a signal value at a vertex should only depend on values at neighboring vertices. In other words, $(Sf)(v)$ for a vertex v is a function of $f(u)$ for vertices u that are adjacent to v.

Common choices for S include the adjacency matrix of the graph, the graph Laplacian, and their normalized versions. For example, if A is the adjacency matrix of the graph, then $S = A$ is a possible choice for the graph shift operator. The (i, j)-th element of A is non-zero (often set to 1) if there is an edge from vertex i to vertex j and zero otherwise. This choice of S shifts the signal value at each vertex to its immediate neighbors.

Alternatively, the graph Laplacian $L = D - A$ (where D is the degree matrix, a diagonal matrix where D_{ii} is the degree of vertex i) can also be used as a shift operator. The Laplacian-based shift reflects a notion of diffusion or spreading of the signal across the graph.

2.3.3. GRAPH CONVOLUTIONS

Graph convolutions are a fundamental operation in graph signal processing and Graph Neural Networks (GNNs), extending the concept of convolution from traditional signal processing to data represented on graphs. Mathematically, a graph convolution operation blends local neighborhood information in a graph signal according to the structure defined by a GSO. The graph convolution of the signal $\mathbf{x}$ with a filter $\mathbf{h} \in \mathbb{R}^{|V|}$ is defined as:

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x} \tag{2.13}$$

Here, $\mathbf{H}(\mathbf{S})$ is a graph filter, h_k are the filter coefficients, and K is the filter length. The term $\mathbf{S}^k\mathbf{x}$ represents the k-th power of the GSO applied to the signal, effectively aggregating information from k-hop neighbors.

2.3.4. GRAPH NEURAL NETWORKS

Graph Neural Networks (GNNs) are a class of neural networks designed to operate on data represented as graphs. These networks effectively capture both the graph structure and the features associated with its nodes or edges, making them particularly useful for tasks involving irregularly structured data, such as social networks, molecular structures, and communication networks.

In a GNN, the input is a graph $G = (V, E)$ and a feature vector $\mathbf{x}_v$ for each vertex $v \in V$. The goal of a GNN is to learn a state embedding $\mathbf{h}_v$ for each vertex that captures both its features and its topological role within the graph. This learning process typically involves iterations of message passing and aggregation. Consider a GSO $\mathbf{S}$ representing the graph's connectivity, and let $\mathbf{Z}^{(l)}$ denote the matrix of node embeddings at layer l. The basic operation in a GNN layer can be represented as:

$$\mathbf{Z}^{(l+1)} = \sigma\left(\mathbf{S}\mathbf{Z}^{(l)}\mathbf{W}^{(l)}\right)$$

where $\mathbf{W}^{(l)}$ is a learnable weight matrix at layer l, and σ is a non-linear activation function, such as ReLU. This formula represents the aggregation of information from the neighbors of each node (via the GSO $\mathbf{S}$) and the transformation of this information through learnable parameters. In real-world applications, the filter coefficients h_k are typically constrained to a small number of parameters to reduce computational complexity and overfitting. This leads to localized filters that aggregate information only from close neighbors. Moreover, in many GNN architectures, the filter is simplified to operate on one-hop neighbors only (i.e., $K = 1$) for efficiency.

3 Literature Survey

Chapter 3 provides an extensive literature survey on various techniques and methodologies related to radar waveform optimization. In Section 3.1, classical optimization methods for radar waveforms are reviewed, including linear programming, convex optimization, and gradient-based methods in Subsection 3.1.1. Subsection 3.1.2 explores the use of genetic algorithms in waveform design, highlighting their effectiveness and limitations. Section 3.2 discusses the application of neural networks, particularly deep learning models, in optimizing radar waveforms. Section 3.3 delves into the utilization of reinforcement learning for waveform optimization, emphasizing its advantages in dynamic environments. Finally, Section 3.4 examines the integration of Graph Neural Networks with Multi-Agent Reinforcement Learning, showcasing their potential in handling complex, structured environments. Section 3.5 concludes the chapter with a short discussion.

3.1. CLASSICAL WAVEFORM OPTIMIZATION

The main goal of radar waveform optimization is to adaptively design radar signals that are best suited for specific operational scenarios and objectives, such as detecting small or fast-moving targets and operating in cluttered environments. The optimization process involves adjusting parameters like waveform shape, frequency, duration, and power to suit specific tasks or environments. Classical solvers often use well-established mathematical techniques such as linear programming, convex optimization, and gradient-based methods. These approaches focus on achieving the best trade-off between conflicting requirements, such as maximizing range resolution while minimizing power consumption.

In this section, we will briefly describe different traditional methods of optimizing waveforms.

3.1.1. MATHEMATICAL OPTIMIZATION TECHNIQUES

The paper [22] introduces a radar waveform design using Lagrange multipliers to minimize mismatch loss, focusing on constraints from cross-correlation and peak correlation responses. The technique creates group-complementary code sets to enhance signal distinction and accuracy by managing sidelobes but requires complex computations and expertise. Conversely, [23] discusses an iterative convex optimization for cognitive radar waveform synthesis, aiming to minimize spectral density errors and suppress interference, contrasting with traditional methods by emphasizing faster convergence and reduced errors.

Solvers can find optimal solutions over time, but suboptimal ones may be more computationally efficient [24]. The paper addresses robust waveform design against colored Gaussian noise using a max-min strategy for optimizing detection across Doppler shifts, tackling the complex QCQP problem with a polynomial complexity algorithm for a viable sub-optimal solution. Additionally, it explores non-convex optimization for MIMO radar by jointly designing transmit sequences and receive filters through Riemannian geometric methods [25], efficiently solving constant-envelope waveform challenges with advanced algorithms for reduced iteration complexity.

Classical solvers, grounded in mathematical and physical principles, provide robust and well-understood methodologies for radar waveform optimization. These techniques are typically deterministic, offering precise control over waveform characteristics, and are effective in well-defined scenarios. However, they may lack flexibility in adapting to dynamic or complex environments and can be computationally intensive for large-scale problems. The next sections will showcase the importance of methods such as genetic algorithms or neural networks. While they are not guaranteed to reach the optimal solution, they provide benefits in terms of speed.

3.1.2. GENETIC ALGORITHMS

Genetic algorithms (GAs) are optimization methods mimicking natural selection, starting with a random solution set. Each solution's viability is determined by a fitness function. GAs evolve solutions through selection, crossover, and mutation, gradually optimizing toward the best outcome. The process continues until a satisfactory solution is found or a preset generation limit is reached, effectively navigating large solution spaces to identify optimal solutions for complex problems.

In [26], the authors demonstrate how evolutionary algorithms can be used to minimize blind areas in range/Doppler space, leading to PRF sets that compare very favorably against both exhaustive search results and existing methods. This advancement in medium PRF radar design showcases the potential of evolutionary algorithms in optimizing complex systems where multiple conflicting criteria must be balanced.

Selecting the optimal waveform based on the current scenario can pose a challenge even for experts [27]. In the paper, the focus is on exploring the utilization of the radar ambiguity function in various configurations, including monostatic, bistatic, and multistatic setups. A genetic algorithm is employed with a specific emphasis on incorporating the autocorrelation and ambiguity functions into the evaluation of fitness. The results show that genetic algorithms are an effective tool, achieving a near-perfect (within 0.005% of the optimal one) solution in the case of multistatic radar. In a somewhat similar fashion, the work in [28] introduces a modified genetic algorithm (GA) to design orthogonal discrete frequency-coding waveforms (DFCWs) for MIMO radar systems. These waveforms exhibit desirable aperiodic autocorrelation and cross-correlation properties. The paper highlights the enhanced correlation properties of the designed waveforms compared to others in existing literature. It also explores the impact of Doppler frequency shift on these signals and their ambiguity function.

While GAs have been useful in solving complex problems with large solution spaces, including radar waveform selection, they are increasingly being supplanted by more efficient and specialized algorithms. GAs often require significant computational resources and time, especially for complex problems with large solution spaces [27]. GAs may face challenges in effectively handling multi-objective optimization problems, which are common in radar waveform design. This limitation has prompted researchers to explore other optimization methods [29].

3.2. SUPERVISED LEARNING

Supervised learning models, particularly deep learning, excel in learning and adapting to complex, dynamic environments such as radar waveform optimization. Once trained, these models offer quick predictions for new data, showcasing greater computational efficiency and scalability compared to the evolutionary strategies of Genetic Algorithms (GAs), which require reapplication for each new problem instance.

The study in [30] presents a deep learning technique for creating unimodular waveforms in MIMO radar, optimizing a wide range of metrics efficiently and outperforming older methods limited to fewer metrics. This approach achieves excellent autocorrelation and cross-correlation, and is time-efficient for practical use, but faces challenges like complexity, potential generalizability issues across radar systems, and dependence on training data quality and quantity.

Radar technology, essential for the automotive industry and advanced driver-assistance systems, is improved by NeuroWav, a novel supervised learning approach for waveform design in Vehicular Ad-Hoc Networks (VANETs), introduced in [31]. It effectively tackles radio frequency interference, outperforming traditional algorithms by being faster and more efficient without sacrificing performance, making it more suitable for real-time applications.

While most methods use fully connected neural networks, the work in [32] introduces a supervised learning method for designing radar waveforms using convolutional neural networks (CNNs). This method designs sequences with improved auto- and cross-correlation properties, utilizing group convolution and identity mapping in the neural network framework. CNNs demonstrate better correlation properties than traditional algorithms, with a significant reduction in the number of parameters, reducing the computational burden. Their use of kernels leads to better feature extraction and utilization of spatial information in the waveform.

The work [33] explores using recurrent neural networks for designing radar waveforms in dynamic environments. This approach aims to reduce the convergence time of traditional adaptive radar waveform design methods, which typically require complex optimization routines. The strength of this method lies in its application of RNNs, offering efficiency and speed suitable for dynamic environments. Initial trials show promising results, achieving comparable characteristics to the Error Reduction Algorithm [34]. However, the method faces challenges, including the complexity and resource requirements of RNNs.

These supervised learning algorithms offer significant improvements over traditional methods, primarily in their ability to handle complex, high-dimensional problems with greater efficiency and accuracy. Their adaptability to dynamic environments and proficiency in modeling non-linear relationships make them particularly suitable for radar waveform optimization. However, challenges such as computational complexity and dependency on the quality of training data can impact their practical deployment.

3.3. REINFORCEMENT LEARNING

Reinforcement learning algorithms excel in situations where they must adapt to changing environments. In radar systems, this means adjusting the radar waveform in response to the evolving state of the target and interference environments [35]. One of the significant advantages of reinforcement learning (RL) in applications like radar waveform optimization is its ability to use simulated environments, reducing the reliance on varied, informative, error-free datasets. It is much easier to measure the quality of a waveform than to produce input-output pairs since we do not know what the optimal waveform is for a certain input a priori. Simulations can also be tailored to specific requirements or conditions of the radar system. This allows for more targeted and effective training of the RL model, as the simulated environment can be adjusted to represent a wide range of conditions and challenges the radar might face. RL is inherently designed to handle sequential decision-making processes. The position of the target depends on its previous position and velocity. RL's framework, particularly when modeled as a Markov Decision Process, is well-suited for this task [36], as opposed to non-Markovian RL [37], [38].

The authors of [39] explore the usage of the Deep Deterministic Policy Gradient(DDPG) algorithm for the selection of phases for a phase-coded waveform that optimally shares spectrum, evident through the creation of a low-power notch in the power spectrum. Similarly, the authors in [7] propose a solution to the same problem using deep reinforcement learning (Deep RL), specifically focusing on varying radar waveform bandwidth and center frequency. They demonstrate that their Deep RL approach, which employs a Deep Q-Learning (DQL) algorithm, significantly enhances radar performance metrics, outperforming traditional methods such as policy iteration and sense-and-avoid strategies.

The work in [40] presents a cognitive beamforming algorithm for colocated Multiple-Input Multiple-Output (MIMO) radars using a Reinforcement Learning (RL) framework. This study focuses on addressing the challenge of detecting multiple targets in an environment where the number and positions of targets are unknown. The proposed RL-based optimization protocol enables the MIMO radar system, treated as the agent within this framework, to iteratively sense the unknown environment and adjust its transmitted waveforms accordingly. The key contribution lies in the radar's ability to synthesize a set of transmitted waveforms whose beam patterns are tailored based on the knowledge acquired about the environment. The authors extend their approach to environments with unknown disturbance statistics, outperforming classical solutions [41].

The study in [35] addresses the challenge of a continuously changing cognitive radar tracking system, which results in an infinite number of states in the environment and targets. To tackle this, the authors design a novel deep Q-network that maps the state-action pair to its Q-values, with the radar acting as the agent, the state being the entropy of the environment, and the action being the transmitted waveform. This approach significantly improves the precision of target tracking, illustrating the potential of DRL in enhancing the adaptability and efficiency of cognitive radar systems in dynamic environments.

In [8] the use of reinforcement learning (RL) in the optimization of radar waveforms is explored. The authors approach the problem of constrained waveform optimization as a sequential decision problem. The authors demonstrate the effectiveness of their proposed method by optimizing an agent's policy to define the number of pulses, as well as their duration and modulation parameters while optimizing a user-defined figure of merit.

Traditional single-agent models may not adequately capture the interactions and dependencies between different components of the environment, such as various frequency bands. MARL allows for a more nuanced and comprehensive modeling of these interactions [42]. The authors propose a framework where each frequency band is treated as an independent agent within a multi-agent system. These agents interact with the environment, receive individual observations, share rewards, and collectively update a Q-network. The key achievement of this approach is the significant improvement in mutual information and Signal-to-Interference-plus-Noise Ratio (SINR) over traditional methods like the water injection method. The paper demonstrates that the SINR of waveforms designed using this multi-agent RL approach is substantially higher than that of Linear Frequency Modulation (LFM) waveforms.

3.4. Graph Neural Networks & Multi-Agent Reinforcement Learning

Graph Neural Networks combined with Multi-Agent Reinforcement Learning are gaining significant attention due to their ability to efficiently handle complex, structured environments. GNNs excel in capturing the relationships and dependencies within data structured as graphs. When integrated with MARL, they provide a powerful framework for modeling and solving problems where multiple agents interact within an environment, such as in social networks, traffic management, and strategic games.

Addressing the complex dynamics of connected autonomous vehicle (CAV) networks, the authors present a deep reinforcement learning algorithm that combines a graph convolution neural network (GCNN) with a deep Q-network (DQN) [43]. The GCNN processes the network's data, treating vehicles as nodes and their communications as edges, effectively capturing vehicle interactions crucial for CAV operations. The DQN component optimizes decisions based on GCNN's insights, focusing on safety, efficiency, and smooth traffic flow. The approach demonstrates improved cooperative control, safety, and mobility in CAV environments by effectively utilizing sensory and connectivity data.

The authors of [19] introduce a combination of MARL and GAT for real-time strategy games. The foundational aspect of their approach is the QMIX algorithm, which is a distributed MADRL framework. QMIX is important for enabling effective distributed computation and decision-making among multiple agents in the complex environment of an RTS game. State categorization serves as a means to preprocess and structure the game data in a way that is more amenable to analysis via graph neural networks. The paper leverages self-attention mechanisms within the graph neural network framework to discern the inter-agent relationships in the form of graphs.

The paper [44] introduces a method merging graph neural networks (GNNs) with vector-valued Q-learning for solving multi-objective factored MDPs, transitioning towards multi-objective RL (MORL) that accommodates multiple reward functions for diverse objectives. This facilitates adaptable agent training across varying utility functions by employing a Q-value estimator that outputs a matrix for deriving scalarized action-state values. The authors develop Batch-FMODQN, an algorithm leveraging a message-passing GNN architecture to handle inputs of state variables and corresponding rewards, aiming at environments with variable numbers of state variables, actions, and local rewards.

3.5. DISCUSSION

In radar systems, the need for adaptivity in waveform optimization arises from the dynamic and unpredictable nature of target environments and interference. Classical optimization techniques, while useful, do not offer the real-time adaptivity needed in some scenarios, where the response time is measured in seconds. A possible solution to this problem is to learn a policy, which is faster than the classical methods, but can also adapt to the changing environment. Neural networks are effective at learning complex policies from data, enabling rapid and flexible responses to new situations, essentially surogating the optimization time to the training phase. However, supervised learning approaches have their challenges. Due to the nature of the problem, reinforcement learning is a suitable alternative, due to the ease with which we can simulate radar environments. The work that has been done up to now shows that RL can be successfully applied to radar problems, but the adoption of this technology in the field is not yet widespread, perhaps due to the complexity of the algorithms and the computational complexity. Multi-agent reinforcement learning offers a way of expressing the multi-objective problem which is radar waveform optimization as a multi-agent game, where agents use policies to output waveforms according to predefined objectives and the target environment. The advantage of this solution is that there is no need for a dataset since we can simulate the radar environment, and the learned policy can adapt in real-time, with minimal computational requirements in the field. The agents can be exposed to a wide range of scenarios, improving their generalization capabilities, and can be trained to optimize one or more desired figures of merit. Furthermore, we can use graph neural networks to express the dependencies between the different agents or their objectives, as a graph, and enhance learning by infusing our already existing knowledge of radars into the behavior of the agents. GNNs also have the advantage of scalability when compared to fully connected neural networks, a desirable property in MARL, where the computational complexity increases rapidly with the number of agents.

4 MARL for Radar Waveform Design

Chapter 4 presents radar waveform design as an optimization problem framed within the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework. Section 4.1 introduces the parameters and metrics used in defining radar waveforms and their optimization. Section 4.2 details the reinforcement learning environment, including the state, observation, and action spaces, as well as the reward and transition functions. In Section 4.3, the concept of a centralized critic is introduced to address instability and non-stationarity in multi-agent learning. We also propose the use of Graph Neural Networks (GNNs) as a centralized critic to improve scalability and efficiency by leveraging graph data sparsity.

In this section, we present radar waveform design as an optimization problem, framing it within the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework. We define the state, observation, and action spaces, as well as the reward and transition functions.

4.1. Radar waveform optimization

A burst β is a sequence of pulses transmitted by a radar system within a short duration. It can be characterized by the following parameters: the pulse duration (τ), which represents the duration of each pulse within the burst; the pulse repetition frequency (PRF), indicating the rate at which pulses are transmitted within the burst; and the number of pulses (N), which is the total count of pulses transmitted within the burst and the carrier frequency (f_c), which is the frequency around which the pulses are modulated.

A waveform Ψ is a collection of bursts, where each burst can have different parameters. The bursts are transmitted sequentially. The parameters of all the bursts can be defined as vectors, where we will use $(\boldsymbol{\tau}, \boldsymbol{N}, \boldsymbol{PRF}, \boldsymbol{f_c})$ to refer to all of the parameters comprising a burst.

The duration of a burst can then be expressed in terms of the number of pulses, PRF, and pulse length.

$$T_\beta = N(\tau + \frac{1}{PRF}) \tag{4.1}$$

The duration of a waveform can be computed by summing over all the burst durations T_i, where M is the total number of bursts.

$$T_\Psi = \sum_{m=1}^{M} T_{\beta_i} \tag{4.2}$$

For a desired Doppler resolution $\Delta f_d^{'}$, we define the waveform duration ratio T_Ψ^{ratio} as the ratio between the waveform duration and the minimal waveform duration achievable $T_\Psi^{'}$ given $\Delta f_d^{'}$.

$$T_\Psi^{'} = \frac{c}{2 f_c \Delta f_d^{'}} \tag{4.3}$$

$$T_\Psi^{ratio} = \frac{T_\Psi}{T_\Psi^{'}} \tag{4.4}$$

The probability of detection P_d in radar systems refers to the likelihood that the radar system will correctly detect a target. A detection threshold is set to distinguish between noise and target signals. The choice of threshold impacts both the P_D and the probability of false alarm P_{fa}. Q is the tail probability of the normal distribution. The SNR is the signal-to-noise ratio of the waveform that has been sent, for a specific target.

$$P_d = Q(Q^{-1}(P_{fa}) - \sqrt{2 \cdot \text{SNR}}) \tag{4.5}$$

It is clear that a high probability of detection for a low probability of false alarm is desirable. A high probability of detection can be easily achieved by sending more bursts or pulses, increasing the power sent to the target. However, this will also increase the duration of the waveform, which is a quantity that should be minimized. Short waveforms are desirable because they improve the operational efficiency of the radar system.

The primary objective of this research is to optimize radar waveform design to achieve a high probability of detection while maintaining a waveform duration ratio close to 1. This optimization problem is framed within the DEC-POMDP framework, as a way of tackling the multi-objective nature of radar waveform design.

4.2. Reinforcement Learning Environment

Now that we have defined a model for bursts and waveforms, we can introduce the RL formulation of the radar waveform optimization problem. The simplest way of modeling the problem would be as an MDP, but that would not be very useful in realistic scenarios. An alternative would be to model the problem as a POMDP, which is more akin to real situations where we do not know the true state of the environment. In RL, learning is guided by the reward signal, and in this case, the reward can be expressed as a sum of two sub-rewards, for the probability of detection and for the waveform duration ratio. The problem associated with this approach is that the agent cannot distinguish between getting a high reward for the P_D or for the waveform duration, so it might face difficulties in achieving both objectives simultaneously. Alternatively, we can model the problem as a Dec-POMDP, where one agent aims to maximize the probability of detection while the other agent minimizes the waveform duration. We will now provide a Dec-POMDP formulation of the environment.

State space: The state space describes the physical environment in which the agents take their actions, as well as the characteristics of the target of the radar system:

- Range r: the distance of the target from the radar
- Velocity v: the radial velocity of the target
- Altitude h: the distance of the target from sea level
- Radar cross-section rcs: a measure of how much energy is reflected by the target.
- Wind Speed v_w: contributes to ground and sea clutter
- Rainfall rate ρ: volumetric clutter
- Target Doppler Resolution Δf_{target}: it is correlated with the waveform duration

The true state of the environment is not visible to the agents unless it is included in the observation space. Formally the state is a set:

$$S = \{r, h, v_{target}, v_w, \mathrm{rcs}, \rho, \Delta \mathrm{f}_{\mathrm{target}}\}$$

Action space: The action space defines how the agent interacts with the environment. Each agent has a fixed burst budget, for which they can decide the free parameters. These parameter vectors are:

- The duration of the pulses in each of the bursts, $\boldsymbol{\tau}$
- The number of pulses per burst, $\boldsymbol{N}$
- The PRF of each burst,$\boldsymbol{PRF}$
- The carrier frequency $\boldsymbol{f_c}$ of each burst

While each agent decides the parameters of the bursts allocated to them, all the bursts from all the agents are sent sequentially as one waveform. The actions of the other agents directly influence the reward received by the agent at each time instance, so the agents are forced to cooperate. The action space for each of these parameters is discreet so that the agent can choose between predefined values. The set describing the actions of agent i is defined as:

$$A = A_1 \cup A_2$$

$$A_i = \{\boldsymbol{\tau}, \boldsymbol{N}, \boldsymbol{PRF}, \boldsymbol{f_c}\}$$

Observation space: The observation space defines the state of the environment as it appears to the agents. The agents can observe the position and velocity of the target, with some uncertainty which depends on the previously sent waveform. The observation space also includes the actions of all the agents at the previous time instance.

- Apparent Range $\hat{r}$: the measured distance of the target from the radar
- Apparent Velocity $\hat{v}$: the measured radial velocity of the target
- Altitude $\hat{h}$: the measured distance of the target from sea level
- Wind Speed v_w: it can be measured on-site
- Probability of detection P_d at the target position at the previous time step
- The waveform duration ratio T_{Ψ}^{ratio} of the previous waveform
- The set of actions A at the previous time step

The P_d and T_{Ψ}^{ratio} allow the agent to gauge how well it is doing, and if it needs to adjust the burst parameters since the two metrics are also part of the reward. If the P_d is low, then the agent can also realize that the measured position of the target is not accurate. Given all of this, the observation space Ω becomes:

$$\Omega = A \cup \{\hat{r}, \hat{h}, \hat{v}_{target}, v_w, P_d, T_{\Psi}^{ratio}\}$$

Transition function: The transition model used to update the state only modifies the target position, while the other parameters remain constant throughout the episode. The target velocity remains constant throughout the episode, with the addition of a noise term.

$$T(S_{t-1}) = K_t S_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, C_t)$$

where K_t is the transition matrix and C_t models the noise covariance.

Reward function: The reward function defines what the agent should aim to maximize over time, guiding it toward desired behaviors. One agent maximizes the P_d, while the other agent tries to get the waveform duration ratio T_Ψ^{ratio} as close to 1 as possible. These goals are specific to this setting, but the rewards could be used to guide the agent towards waveforms with good autocorrelation properties or ambiguity functions. In the case of settings with only one agent, the reward becomes a sum of the other sub-rewards. Finally, the agents receive a small bonus reward at each time step if both metrics have values over a certain threshold. This helps agents maximize both objectives and cooperate in the multi-agent setting.

4.3. CENTRALIZED CRITIC

Multi-agent learning presents significant challenges compared to single-agent reinforcement learning due to the complex dynamics and interactions between multiple agents. As explained in Chapter 2, non-stationarity and the credit assignment problem destabilize learning. To tackle these problems, we introduce a centralized critic. One can refer to Figures 2.7 and 2.8 or to Chapter 5 for a better understanding of the various components.

The centralized critic uses the joint observation space, which contains information about the observed state and the joint agent action to produce central value function estimates. Since the actions of the other agents are considered part of the state from the perspective of each agent, both the actions and the state need to be part of the value function computation. We expect that the reduced variance in the central value function estimates will help stabilize learning and boost performance.

The objective function for agent i with respect to its policy parameters θ_i is given by:

$$J(\theta_i) = \mathbb{E}_t\left[\min\left(r_t(\theta_i)\hat{A}_t, \operatorname{clip}(r_t(\theta_i), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{4.6}$$

Where $\hat{A}_t := \hat{A}(s_t, a_t)$ is the estimate of the advantage function. The advantage function is now computer using the central value function instead of the decentralized one. The centralized critic is defined by parameters ϕ, captured by some neural network.

$$A^{\pi_{\theta_i}}(s_t, a_t) = Q^{\pi_{\theta_i}}(s_t, a_t) - V_{central}^{\pi_\phi}(s_t) \tag{4.7}$$

The main challenge introduced by the addition of a centralized critic is scalability. As the joint observation space increases, the number of parameters in the centralized critic also grows, making it infeasible for a large number of agents or radar waveform parameters. Additionally, fully connected networks suffer from position bias, meaning the order of input data matters. Radar bursts are permutation invariant, so if the order in which they are fed to the network changes (e.g., agent 2 first instead of agent 1), the neural network's performance may degrade. Moreover, given the ability to simulate radar environments, we can identify optimal bursts to maximize detection and minimize blind ranges. While agents could potentially learn this, integrating radar domain knowledge into the centralized critic could provide less biased estimates.

To address these issues, we propose using a Graph Neural Network (GNN) to learn the central value function estimate provided by the centralized critic. GNNs have fewer parameters than fully connected networks, which scale poorly with input size. GNNs leverage the sparsity of graph data, performing computations only when two nodes are connected by an edge. This reduction in parameters accelerates training time and decreases computational costs by minimizing the size of matrix multiplications during the network's forward pass. This addresses the scalability issue (point 1).

Graph neural networks do not only provide scalability but also offer a way of representing and accounting for agent interactions in the centralized value function. To process the joint observation space as a graph, each burst intended for the final waveform is represented by a node in the graph (Figure 4.1). The burst parameters—such as PRF, pulse duration, RF, and number of bursts—form the node signal associated with each burst. The observed state information is processed using an encoder and concatenated to the feature vector of each node. Domain knowledge integration is achieved by connecting only nodes that minimize range-doppler blindness. We compute a metric to measure this compatibility and connect nodes based on the metric's value, which also serves as the edge weight. Since GNNs are permutation invariant (the order in which nodes are processed is irrelevant), this addresses the issue of position bias (point 2).

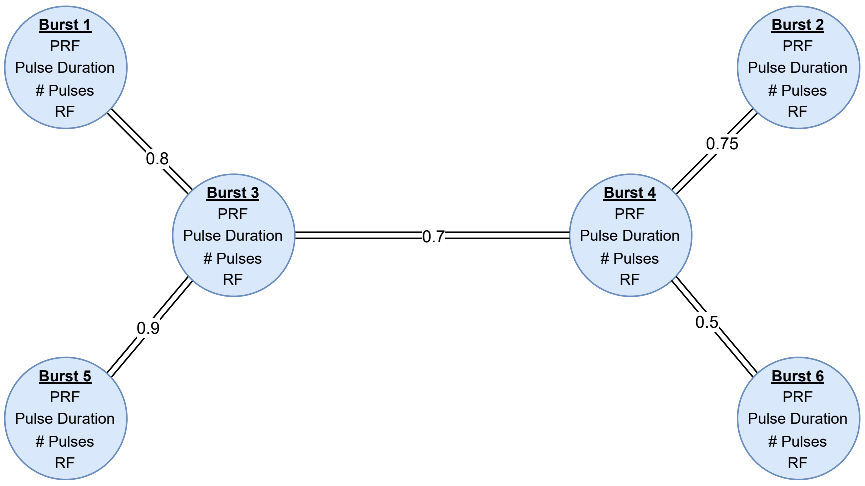


Figure 4.1: A realization of a burst graph. There are 3 bursts per agent, for a total of 6 bursts. The nodes have been connected based on the burst parameters

To summarise, the agents decide on the burst parameters by taking their actions. These bursts are then used to construct the burst graph, which the GNN critic processes to produce a single value for the centralized value function. The agent actions, the bursts (which make up the waveform), are also used to compute the reward for each agent and advance the state to the next time instance. Figure 5.3.2 illustrates the entire pipeline.

5 Model Architecture

Chapter 5 details the architecture of various models used in the thesis. 5.1 introduces the Single Agent Baseline (SA) model, which serves as a benchmark for comparison. It then describes the Independent Actor-Critic (IAC) model in 5.2, where each agent learns a decentralized policy and critic locally. 5.3 further introduces a multi-agent model with a centralized critic using a fully connected network, which leverages global state information to stabilize the learning process. Additionally, 5.3.2 discusses a multi-agent model with a centralized critic utilizing GNNs, detailing methods of graph construction and their impact on performance. Hyperparameter tuning is detailed in Section 5.4.

In this chapter, we delve into the various model architectures proposed and implemented in this research. The primary focus is on three distinct models: the Single Agent Baseline (SA), the Independent Actor-Critic (IAC), and the Independent Actor with Centralized Critic (IACC). The Single Agent Baseline model serves as a foundational benchmark, demonstrating the capabilities and limitations of a single agent controlling the entire radar system. This model provides a crucial point of comparison, highlighting how more sophisticated, multi-agent approaches can potentially outperform a single-agent setup. The Independent Actor-Critic model represents an advancement over the SA model by introducing a decentralized approach where multiple agents operate independently. The Independent Actor with Centralized Critic model further refines the multi-agent approach by incorporating a centralized critic that has access to global state information. This centralized critic is designed to stabilize the learning process by providing consistent value estimates, which can mitigate the challenges of non-stationarity and credit assignment often encountered in multi-agent systems.

5.1. SINGLE AGENT BASELINE (SA)

This model consists of a single PPO agent controlling the entire burst budget. The agent's rewards are the sum of the rewards for each objective: probability of detection and waveform duration ratio. The individual value function is estimated by a separate neural network, the critic. Both the policy and the critic are multi-layer perceptrons (MLPs) with two hidden layers of 256 neurons and tanh activation functions.

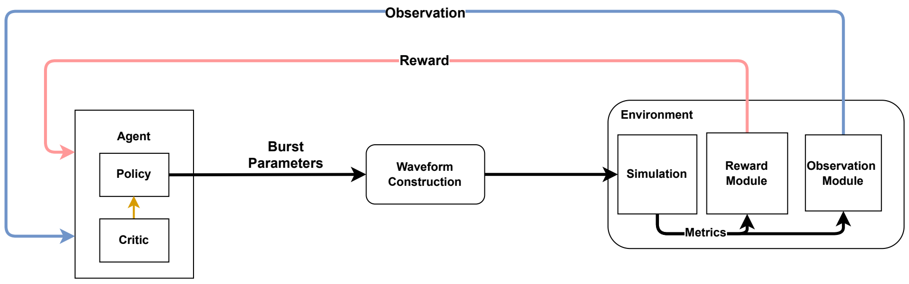


Figure 5.1: Single Agent

5.2. INDEPENDENT ACTOR-CRITIC (IAC)

The first actor-critic (AC) multi-agent extension, called Independent Actor-Critic (IAC), learns a decentralized policy and critic for each of the agents locally. The burst b budget is equally split between agents. At every timestep, each agent takes an action, and an observation is generated for the agent. Each agent has a different reward function, which defines its objective. The architectures of the agents and critics remain the same as for the SA model.

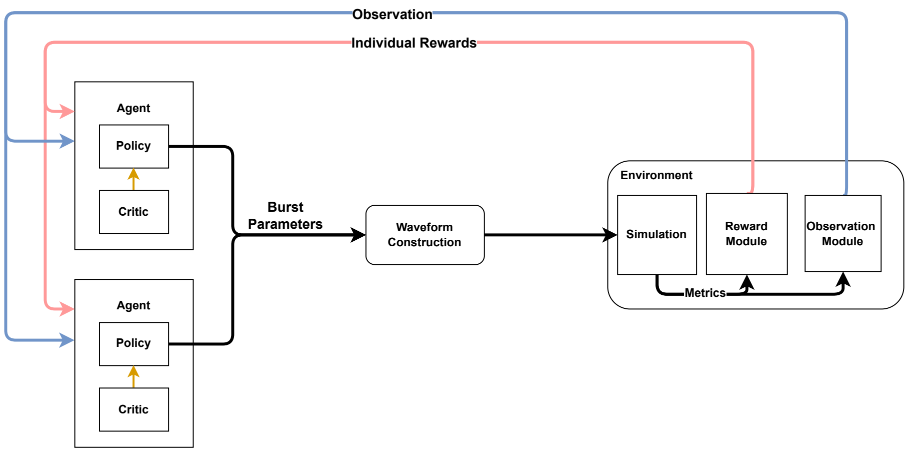


Figure 5.2: IAC

5.3. Independent Actor with Centralized Critic (IACC)

The second AC multi-agent extension is the Independent Actor with Centralized Critic. The centralized critic learns the joint value function, which is used to update each decentralized policy. We distinguish between two different architectures for the centralized critic, one based on an MLP, and the other on a GNN.

5.3.1. IACC (MLP)

The critic MLP has the same architecture as the critics in the previous models. The joint observation is fed directly into the network, which outputs the centralized value function.

5.3.2. IACC (GNN)

This model incorporates a GNN centralized critic. The main difference from the previous model is that the bursts of all agents form a graph, which is then processed by the GNN critic. The GNN uses two graph convolutional layers to compute the centralized value function. The performance of GNNs significantly depends on the graph-building method. We experiment with different methods and observe that connecting nodes when their dissimilarity value is over a certain threshold works best. Experiments with more complex architectures using hierarchical pooling and encoder-decoder modules did not bring any benefits.

5.4. Hyperparameter Tuning

In this study, hyperparameters were systematically tuned using grid search methods. The primary hyperparameters considered include the learning rate, batch size, mini-batch size, stochastic gradient descent (SGD) iterations, discount factor, exploration rate, and network architecture parameters.

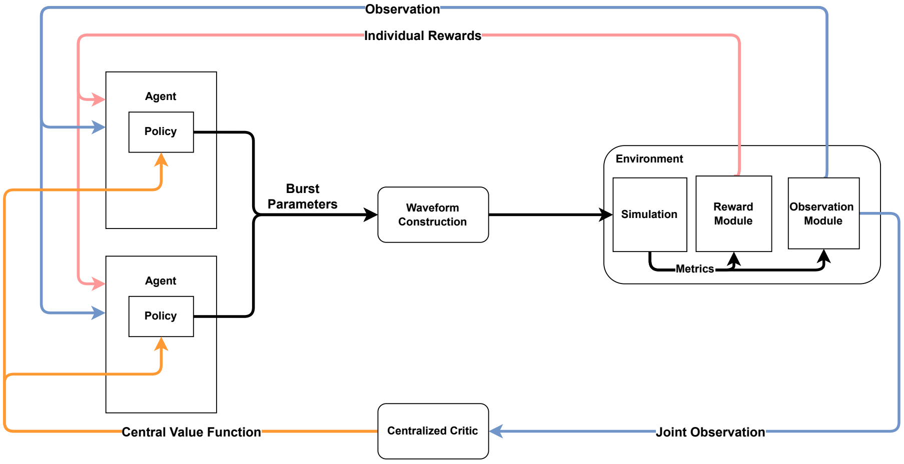


Figure 5.3: IACC (MLP)

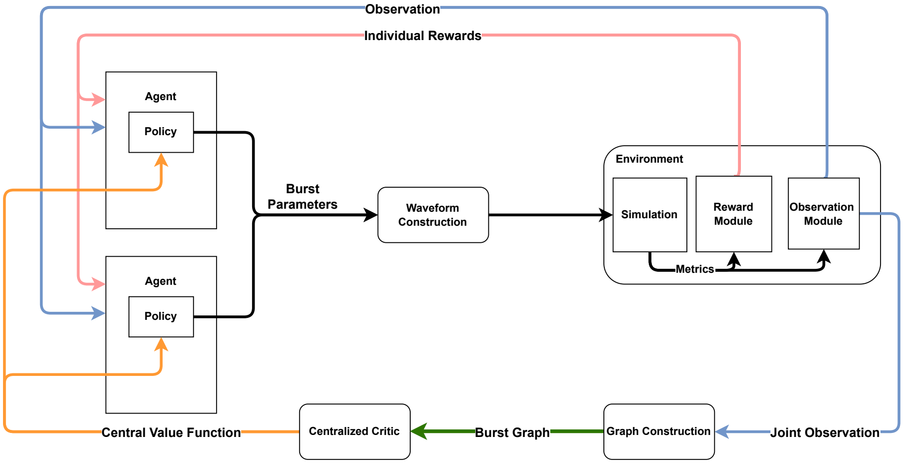


Figure 5.4: IACC (GNN)

Learning Rate

The learning rate, which determines the step size during model updates, was varied between 10^{-5} and 10^{-2}. Lower values contributed to stability, while higher values accelerated convergence but increased the risk of overshooting optimal solutions.

Discount Factor

The discount factor, which balances immediate and future rewards, was tested in the range of 0.90 to 0.99. Higher discount factors prioritized long-term rewards, whereas lower values placed more emphasis on short-term gains.

Exploration Rate

The exploration rate, which governs the trade-off between exploration and exploitation, was annealed from 1 to 0.01 over the training period using various decay schedules (e.g., linear, exponential). An initially high exploration rate facilitated the gathering of diverse experiences, while a reduced rate in later stages allowed for fine-tuning of the policy.

Batch Size, Minibatch Size, and SGD Iterations

- **Batch Size**: The number of episodes or experiences collected before performing a gradient update. Larger batch sizes provided more accurate gradient estimates but required greater memory and computational resources.

- **Minibatch Size**: The number of samples randomly selected from the batch for each step of SGD. Smaller minibatches resulted in more frequent updates and could help avoid local minima, though they introduced higher variance.

- **SGD Iterations**: The number of passes through the minibatch for model parameter updates. Increasing the number of SGD iterations enhanced learning stability but also raised computational costs.

We evaluated batch sizes of 256, 512, 1024, and 2048; minibatch sizes of 32, 64, 128, and 256; and 20, 30, or 40 SGD iterations.

Network Parameters

The architecture of the neural network, a critical factor in the learning capability of the reinforcement learning agent, was tested with 2 to 10 layers and units per layer varying from 64 to 512.

5.4.1. Tuning Results

Using grid search methods, the optimal set of hyperparameters was identified:

- **Learning Rate**: 3×10^{-4}

- **Discount Factor**: 0.98

- **Exploration Rate**: Exponentially decayed

- **Network Architecture**: 2 hidden layers with 256 units each

- **Batch Size**: 512
- **Minibatch Size**: 128
- **SGD Iterations**: 30

We find that shallow, wider architectures work best in this setting. Deeper networks did not bring any sort of improvement in performance and were sometimes detrimental. The underlying data distribution and the relationships between features can be captured well by a shallow network since there is no need for complex feature extraction. Furthermore, a shallow network is less prone to overfitting and is faster to train.

6
Evaluation

Chapter 6 of this thesis evaluates the proposed multi-agent reinforcement learning models in a radar tracking scenario 6.1. The evaluation focuses on several aspects, including Pareto optimality 6.2, optimization times 6.3, and sensitivity to environmental parameters 6.5. Various models, including Independent Actor-Critic and Independent Actor with Centralized Critic with both MLP and GNN architectures, are compared against traditional genetic algorithms and a single-agent baseline. Results are discussed in Section 6.6.

6.1. TRACKING SCENARIO

The reinforcement learning environment for training the agents is a radar tracking simulation designed to detect and follow a low-flying target over the sea. Radar tracking involves continuously monitoring a target's position, speed, and trajectory using radar systems. In this scenario, the wind direction is from the target toward the radar, with rain and waves further complicating the environment.

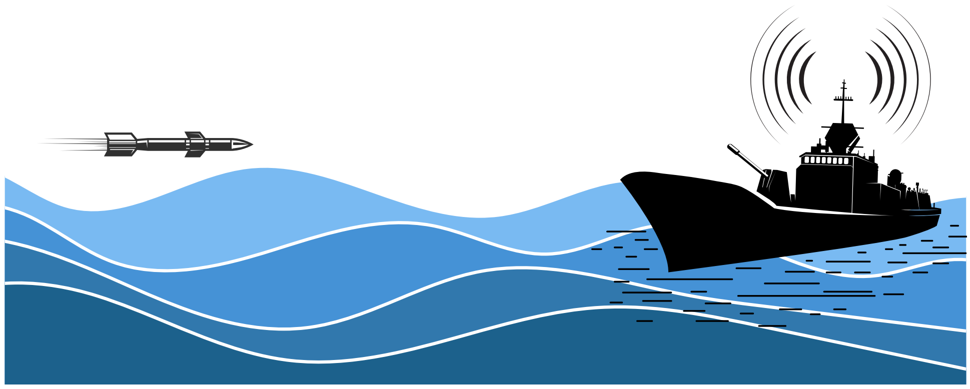


Figure 6.1: Tracking a low flying target over the sea

Tracking a low-flying target over the sea presents numerous challenges. The sea surface generates significant background noise, known as sea clutter, which can obscure radar returns from the target, making it difficult to distinguish the target from reflections off the sea. Additionally, multipath propagation causes radar signals to reflect off both the sea surface and the target, leading to signal interference and distortion.

Parameter	**Minimum**	**Maximum**
Wind Speed (m/s)	0	18
RCS (m^2)	0.1	5
Rainfall rate (mm/hr)	0	4

Table 6.1: Parameter Ranges

Environmental factors and the target itself can have a significant impact on radar performance. In our simulation, we will focus on the radar cross-section of the target, the wind speed, and the rainfall rate, with the purpose of creating diverse and challenging episodes for the agent to learn from. Radar Cross Section (RCS) is a measure of how much electromagnetic energy is reflected back to a radar system by a target. It effectively represents the "visibility" of the target to the radar. RCS is important because it influences the radar's ability to detect, track, and identify objects. Higher RCS values mean stronger return signals, making targets easier to detect and track, while lower RCS values indicate weaker signals, making targets harder to detect. Wind speed affects radar detection by influencing the movement and characteristics of the medium through which radar signals travel. High wind speeds can create turbulent conditions and increased sea clutter, particularly over bodies of water, which can obscure or distort radar returns from targets. Rainfall rate measures the intensity of precipitation, usually in millimeters per hour. It affects radar detection by causing signal attenuation and scattering, which weaken and distort radar returns. High rainfall rates can create significant clutter, masking the radar signals from actual targets and reducing detection accuracy. These parameters are sampled from a uniform distribution at the beginning of each episode. The ranges for each parameter can be seen in Table 6.1.

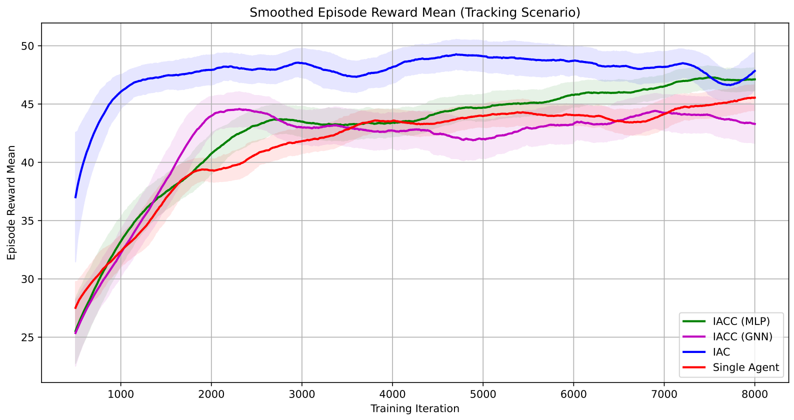


Figure 6.2: Episode Reward Mean during Training

The models are trained for 8000 iterations each. Figure 6.2 shows the smoothed mean episode reward over the training procedure. The IAC model converges the fastest and reaches the highest reward. The IACC (MLP) model converges slower but reaches a similar reward. The single agent and IACC (GNN) models reach slightly lower rewards, with the GNN model also having a larger variance than the other models. However, the reward signal is not the optimization target of the agent, and we cannot conclude anything about the general performance of the agents.

6.2. PARETO OPTIMALITY

Pareto optimality provides a framework for evaluating and comparing RL models based on multiple criteria. A solution is considered Pareto optimal if no other solution can improve one objective without degrading another. In this experiment, we simulate our main use case, where given an approximate range and velocity we would like to detect a target as accurately as possible. Each model receives an initial observation and can send up to 1000 consecutive waveforms to detect the target. We then construct a Pareto front based on the distribution of the waveforms, where we analyze the trade-off each model finds between the probability of detection and waveform duration ratio. The target is at a range of 25km, an altitude of 10 m, and moving with a velocity of 250 m/s. The RCS of the target is 0.5 m^2, the wind speed is 18 m/s and the rainfall rate is 4mm/hr.

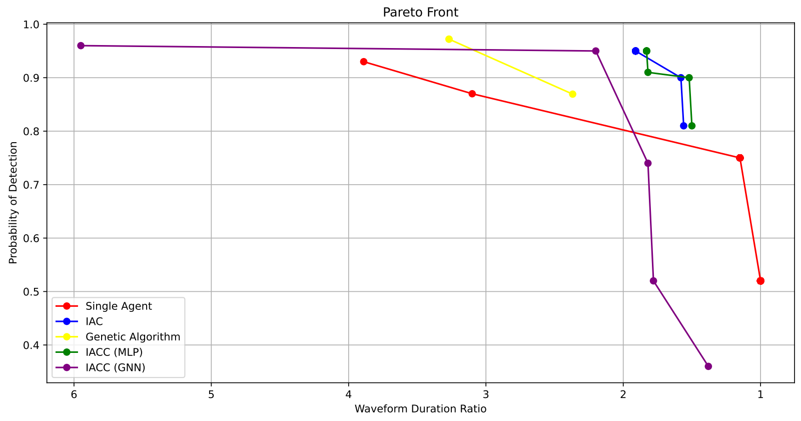


Figure 6.3: Pareto Frontier per Model

Figure 6.3 shows the Pareto front of each model. We also compare with the solution found by a genetic algorithm. The best-performing models are the IAC and the IACC (MLP). They find high-probability waveforms with durations below 2, 1 being optimal. The genetic algorithm finds the highest P_d waveform but does not manage to find waveforms with a duration shorter than 2. The IACC (GNN) model requires much longer waveforms to reach the same P_d as the other models, and we can see a very steep decline in PD once the waveforms become shorter. The single-agent model is somewhat better than the IACC (GNN) model when sending shorter waveforms, but it struggles to reach higher probabilities of detection. To sum up, the IACC (MLP) and IAC models find the best trade-offs between waveform duration and the probability of detection, reaching high P_ds, with short waveforms, while the other models struggle to find waveforms fulfilling both objectives simultaneously.

6.3. COMPARISON OF OPTIMIZATION TIMES

Similarly to the previous experiment, the agents have a maximum of 1000 function calls to find a sufficiently good waveform. The results are averaged over 100 repetitions to ensure a statistically significant result. We define a sufficiently good waveform as one having $P_d \geq 0.95$ and $T_{\Psi}^{ratio} \leq 2$. If the agent reaches 1000 function calls and has not found a good enough waveform, that iteration is counted as having taken 1000 calls, even though the agent has not managed to meet the objective. We define 4 scenarios with varying levels of difficulty, given by the values of the environmental and target parameters: RCS, wind speed, rainfall rate, and target altitude. The results are displayed in Table 6.2 and Figure 6.4.

For all scenario difficulties, both the Genetic Algorithm and the Single Agent baseline required more than 1000 function calls to achieve a good waveform. he IAC model shows significantly better performance compared to GA and SA across all difficulty levels. The

Model	**Scenario Difficulty**			
	Easy	**Medium**	**Hard**	**Very Hard**
GA	1000 <	1000 <	1000 <	1000 <
SA	1000 <	1000 <	1000 <	1000 <
IAC	**1.8 ± 1.4**	**1.6 ± 1.1**	**1.9 ± 1.5**	**32.5 ± 38.5**
IACC (MLP)	1.9 ± 1.5	1.7 ± 1.3	1.9 ± 1.6	38.0 ± 43.5
IACC (GNN)	3 ± 0	36.3 ± 120.4	805.6 ± 323.9	1000 <

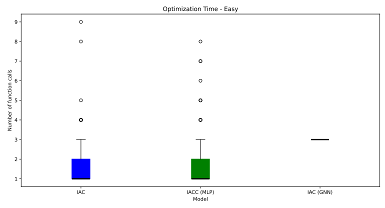
Table 6.2: Mean number of function calls (with standard deviation) needed to reach a sufficiently good waveform

number of function calls remains low and consistent for Easy, Medium, and Hard scenarios. The IACC (MLP) model also performs well with low and consistent mean function calls for Easy, Medium, and Hard scenarios. The IACC (GNN) model shows good performance for the Easy scenario but exhibits a dramatic increase in the mean number of function calls as the difficulty increases. For the Very Hard scenario, the number of function calls exceeds 1000, indicating that the IACC (GNN) struggles significantly as the difficulty increases. The results indicate that the performance of the agents does not scale linearly with the difficulty, as the number of function calls increases dramatically. The mean of the IACC (GNN) is also skewed in the Hard scenario as it fails to find a good waveform in around 70 % of the iterations. If those occurrences are removed, the mean number of function calls becomes 392.5 ± 275.6. The IAC and IACC (MLP) models never fail to reach the optimization target. IAC and IACC (MLP) are the most efficient models for reaching a good waveform across all scenarios, with low mean function calls and standard deviations across all scenarios. The IACC (GNN) model shows poor performance with increasing difficulty, suggesting it may not be suitable for more complex scenarios. The GA and SA models are not sufficiently good optimizers.

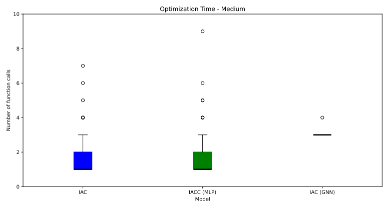
6.4. WAVEFORMS

To better understand the learned policies of the agents and the differences in performance, we need to compare the waveforms that each model chooses in different scenarios. In this experiment, we have plotted some of the best waveforms sent by each agent in scenarios varying from easy to very hard. The plots show the probability of detection across the range-doppler domain. Yellow denotes a high probability of detection, while blue is a probability of 0. The target is no longer visible after 30-40 km, depending on the scenario, as it goes over the horizon due to the low altitude. The plots can be seen in Figures 6.6, 6.7, 6.8, 6.9.

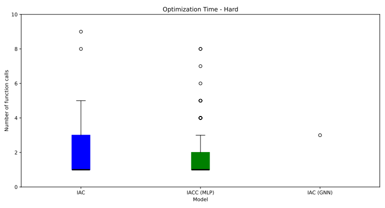
From the visualization, it is evident that the SA model focuses on sending bursts with short pulse durations, as the P_d at low ranges is very high. This is more visible especially when compared to the other models, which have low P_ds in the low-range areas, The agents do not need to track anything at such low ranges, and they are not trained for it. The actions of the SA model are a consequence of the reward function and the policy getting stuck in a local maxima where it learns to minimize the waveform duration ratio, instead of both objectives. The other 3 models achieve better coverage of the tracking area, in terms of mean probability of detection, or minimizing the blind range-doppler



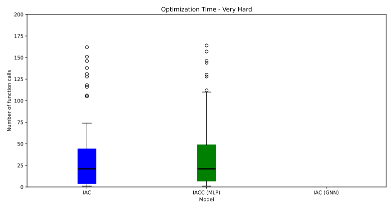
(a) Easy Scenario



(b) Medium Scenario



(c) Hard Scenario



(d) Very Hard Scenario

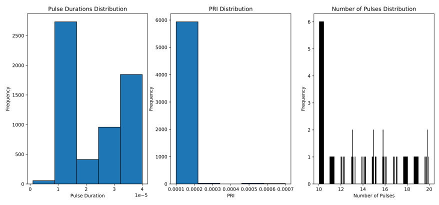
Figure 6.4: Boxplots for the different scenarios. The IACC (GNN) model is not visible in some of the plots so the other two models can be visualized.

zones. They appear to send identical waveforms, although a closer inspection will reveal that there are small differences between the models. The close resemblance of the waveforms suggests that the agents converge to similar policies, which is expected, given the similar architectures and reward functions.

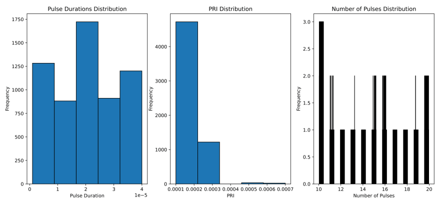
However, as we have seen so far, the performance of the agents differs by a significant margin. Although the waveforms are very similar, the agents' policies are not identical. It was already shown that the IACC (GNN) model is not able to find as short, high P_d waveforms as the IAC and IACC (MLP) models. The explanation for the similarity of the waveforms is that even though the IACC (GNN) waveform has a high mean P_d over the tracking area, the actual duration of the waveform is longer than the other 2 MARL models. If we analyze the action distribution of each agent in Figure 6.5, we can see that the GNN model favors a higher number of pulses and longer PRIs, which result in a higher probability of detection, but lengthen the duration of the waveform. On the other hand, the single agent baseline favors short PRIs, and a low number of pulses, resulting in a low probability of detection and short waveforms. The IAC and IACC(MLP) have almost identical action distributions for this scenario.

6.5. SENSITIVITY TO ENVIRONMENTAL PARAMETERS

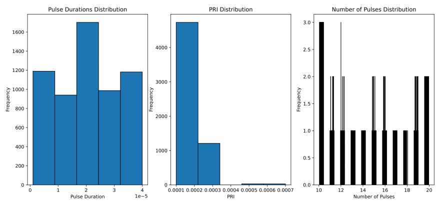
The aim of this experiment was to compare the average performance of different models and to observe how their performance varies with different environmental parameters. Each parameter range (as shown in Table 6.1) is divided into 20 discrete cells. For each cell, 1000 episodes are run, and the mean value of a metric is calculated over all iterations. The parameters that are not being varied in each figure remains fixed throughout



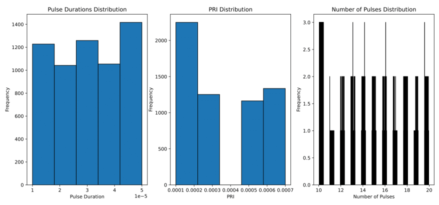
(a) Action Distribution for Single Agent Baseline



(b) Action Distribution for IAC

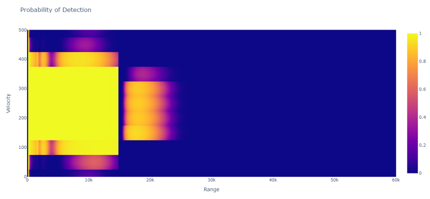


(c) Action Distribution for IACC (MLP)

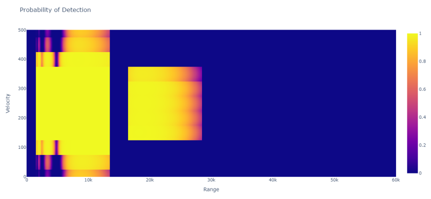


(d) Action Distribution for IACC (GNN)

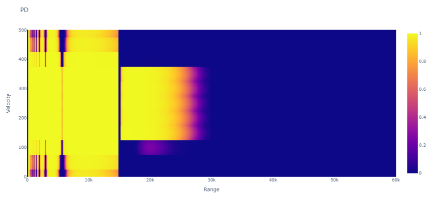
Figure 6.5: Main caption for all subfigures



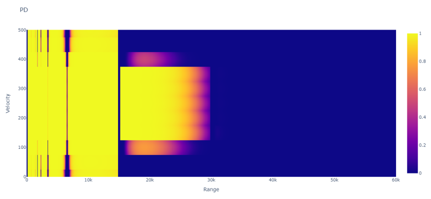
(a) Very Hard



(b) Hard

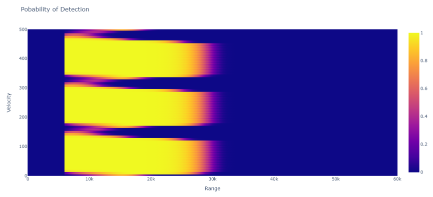


(c) Medium

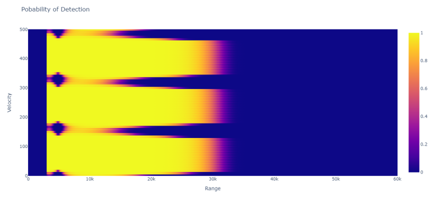


(d) Easy

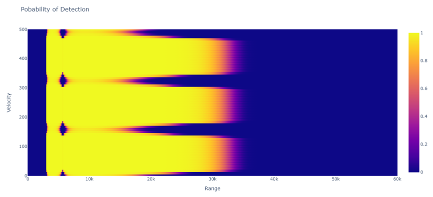
Figure 6.6: Range-Doppler probability of detection for the SA model across 4 scenarios



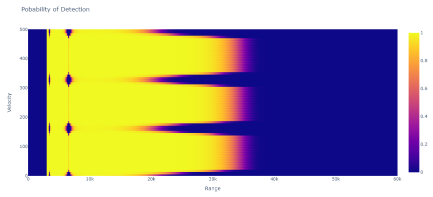
(a) Very Hard



(b) Hard

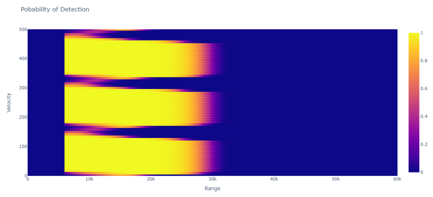


(c) Medium

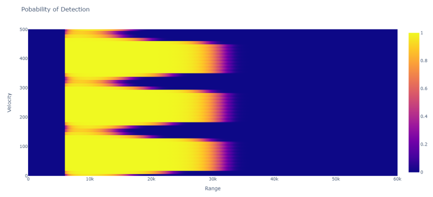


(d) Easy

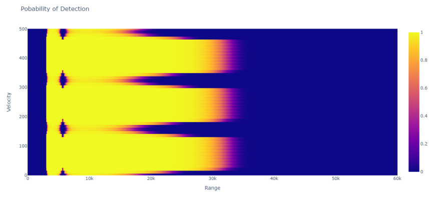
Figure 6.7: Range-Doppler probability of detection for the IAC model across 4 scenarios



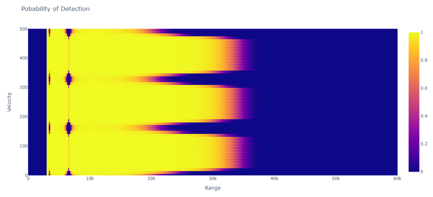
(a) Very Hard



(b) Hard

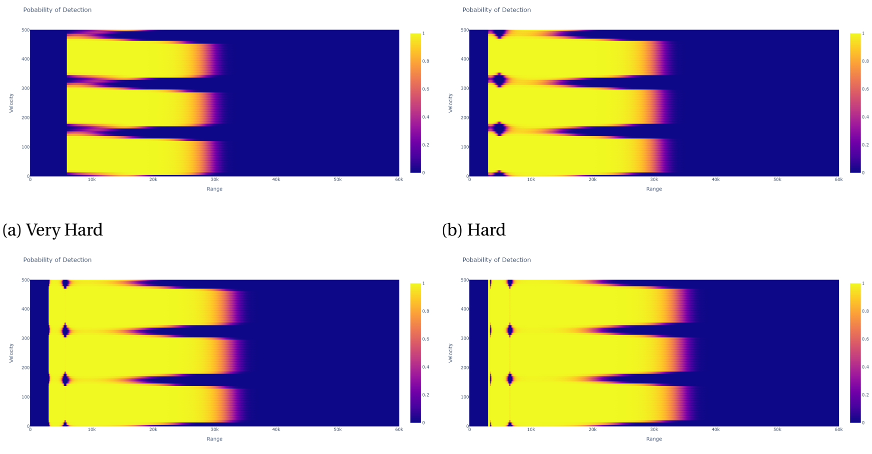


(c) Medium



(d) Easy

Figure 6.8: Range-Doppler probability of detection for the IACC (GNN) model across 4 scenarios

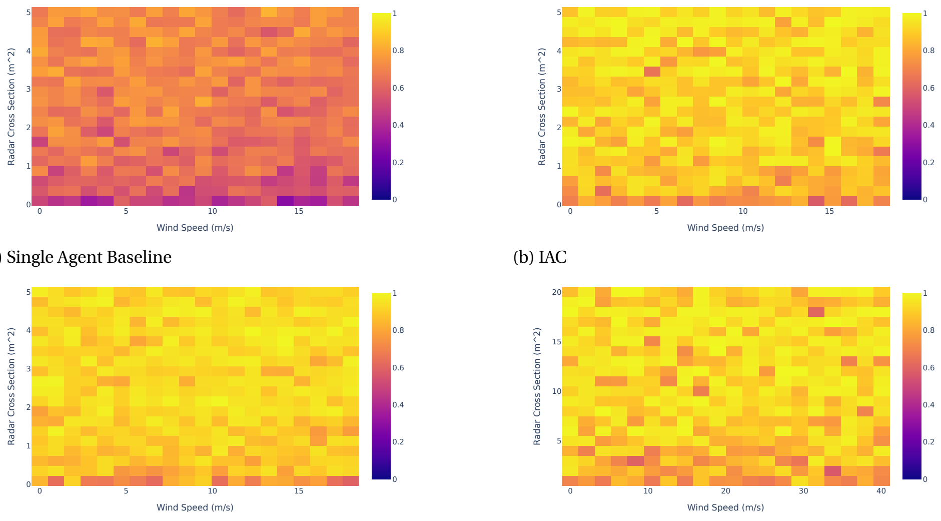


(a) Very Hard (b) Hard

(c) Medium (d) Easy

Figure 6.9: Range-Doppler probability of detection for the IACC (MLP) model across 4 scenarios

all the episodes. Heatmaps are used to visualize all pairwise combinations of parameters for each metric. These heatmaps can be found in Appendix A as well as in Figure A.5.



(a) Single Agent Baseline (b) IAC

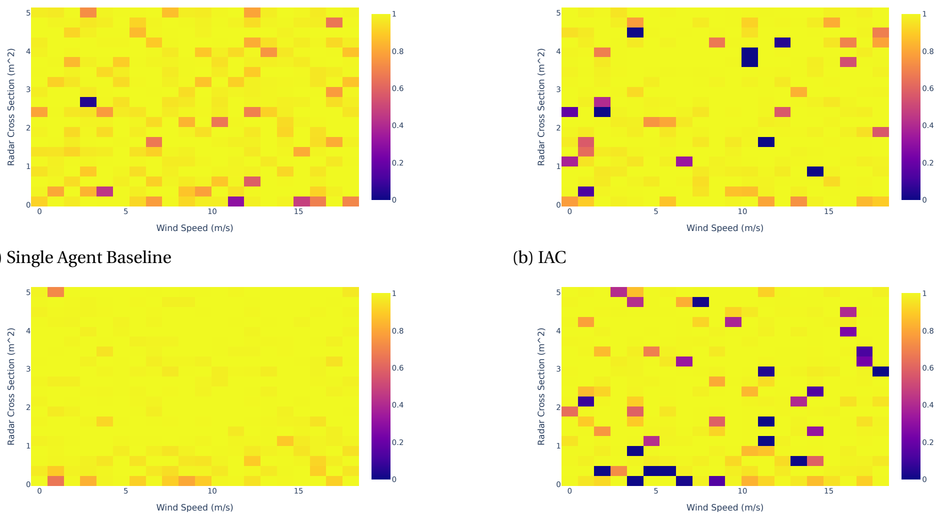
(c) IACC (MLP) (d) IACC (GNN)

Figure 6.10: Probability of detection for radar cross-section versus wind speed

From figures in Appendix A it can be seen that the IACC (MLP) model is the most robust to changes in the environment, maintaining a high average probability of detection as the RCS, rainfall rate and wind speed increase. The IAC and IACC (GNN) models offer similar performance, while the Single Agent Baseline performs the worst on average, and is also the most affected by environmental conditions. From figures in A we can see that the average waveform duration ratio is not affected by the changes in parameters. The IAC and IACC (GNN) models reach the lowest duration, closely followed by the IACC (MLP) model. The slightly longer waveform duration also explains the higher probability of detection. The Single Agent Baseline finds much longer waveforms on average.

In practice, however, we are not as interested in the average performance of the waveform, but in the best waveform the agent can output given a limited time budget. We will now investigate how the quality of the best waveform produced by the agent varies with the changing environment. Instead of averaging over 1000 episodes, we now select the best waveform out of the set of candidate waveforms produced by the agent in one episode (20 timesteps).

Figures in B show that the models perform well, and seem to randomly fail in some cells. The IACC model is again the most robust and fails to find a good waveform the least. Surprisingly, the Single Agent baseline is more consistent than the other two remaining models, by a small margin. Finally, the waveform duration ratio is mostly unaffected for the multi-agent models, with the Single Agent baseline seeing a significant increase.



(a) Single Agent Baseline

(b) IAC

(c) IACC (MLP)

(d) IACC (GNN)

Figure 6.11: Probability of detection for radar cross-section versus wind speed

6.6. DISCUSSION

The evaluation chapter has provided a comprehensive comparison of various MARL models in a radar-tracking scenario. Here, we discuss the key findings from our experiments and their implications.

The IAC model demonstrated the fastest convergence during training, attributed to the more stable policy gradients [20]. Stable policy gradients provide more consistent updates to the policy, reducing the likelihood of drastic changes from one update to the next. This consistency helps the policy to improve steadily rather than erratically, leading to smoother and more reliable learning trajectories. With lower variance, the policy gradients more accurately represent the true gradient of the expected reward. This accuracy ensures that each update moves the policy in the right direction, resulting in faster convergence to an optimal or near-optimal policy. Another minor finding is that the cooperation reward is key to the multi-agent models outperforming the single-agent baseline. Without it, unsurprisingly, the agents do not learn to coordinate.

The IACC's (MLP) and IAC's policies found the best trade-offs between waveform duration ratio and probability of detection, outputting waveforms that achieved both objectives simultaneously. In contrast, the single-agent baseline had trouble optimizing for both objectives simultaneously, instead prioritizing waveform duration ratio and neglecting the probability of detection. The reason for this is that reward signal The IACC (MLP) model proved to be the most robust to varying environmental conditions and the fastest and best performing when faced with a challenging test scenario. Since centralized critics are not limited to local observations, they can avoid the biases introduced by partial observability. Decentralized critics, on the other hand, base their estimates on limited local information, which can lead to biased or incomplete value estimates. Accurate value function estimates allow the centralized critic model to take optimal actions regardless of the difficulty of the task.

On the other hand, the IACC (GNN) model does not perform as well. It is unable to find sufficiently good waveforms in difficult scenarios and suffers from slow training times due to the graph-building process. There is likely no need to construct a radar-informed graph, as it hampers performance, and an MLP critic can learn the correlations between burst parameters on its own. The permutation invariance of the GNN also does not seem to provide any benefits. The IACC (GNN) might show increased benefits in larger coordination tasks, where the increased number of agents and the joint observation scaling affect the size of the MLP.

To summarize, we have seen that moving from a single-agent setting to a multi-agent setting is beneficial for radar waveform optimization. The IAC and IACC agents show greatly improved performance, and the IAC model also boasts faster convergence due to the stability in policy gradients. The IACC's (MLP) centralized critic makes it resilient to changes in the environment and to the non-stationarity induced by other agents.

7 Conclusion & Future Work

This chapter presents the conclusion of the work carried out in the rest of this thesis. In Section 7.1, a short summary is presented, together with the answer to the research question. Section 7.2 presents avenues for further research.

7.1. CONCLUSION

In this thesis, we explored the application of reinforcement learning to the problem of radar waveform optimization, with a particular focus on multi-agent reinforcement learning approaches.

Chapter 1 outlines the importance of radar technology in various applications such as aviation, maritime navigation, and defense. It highlights the challenges faced by radar systems, including interference, clutter, and the need for high resolution and accuracy. The chapter introduces cognitive radar as an advanced approach to overcoming these challenges through adaptability and intelligence, setting the stage for the exploration of reinforcement learning methods. We posed the following research question:

RQ) **How can multi-agent reinforcement learning (MARL) be effectively utilized to optimize radar waveforms?**

In particular, we focus on answering:

(a) How can the multi-objective radar waveform optimization problem be modeled as a MARL problem?

(b) How can radar domain knowledge be integrated to improve performance of the agents?

(c) What are the possible architectures that allow scalability in the number of radar waveform parameters?

Chapter 2 provided a comprehensive overview of radar technology, reinforcement learning, and graph machine learning. Fundamental concepts such as Pulse Doppler Radar, Markov Decision Processes, and graph convolutions were explained. The section on reinforcement learning covered important algorithms like Proximal Policy Optimization, IPPO and MAPPO, while the graph machine learning section introduced the key concepts and techniques used in GNNs.

In Chapter 3 we reviewed existing literature on radar waveform optimization, covering classical methods like mathematical optimizers and genetic algorithms, as well as more recent approaches using neural networks and reinforcement learning. The survey highlighted the strengths and limitations of these methods and identified gaps that our research aimed to address, particularly the potential of MARL in optimizing radar waveforms.

In Chapter 4, radar waveform design is framed as an optimization problem within the Decentralized Partially Observable Markov Decision Process framework. We define the radar environment, as well as the agents' observations and the actions they can take. We define the multiple objectives of radar waveform optimization as different reward functions for our agents. To address the challenges of instability and non-stationarity in multi-agent learning, the chapter introduces the concept of a centralized critic. This centralized critic has access to global state information and helps stabilize the learning process. Additionally, the use of Graph Neural Networks as a centralized critic is proposed to leverage the sparsity and structured nature of graph data, enhancing scalability and efficiency.

The architecture of various models used in the research is detailed in Chapter 5. It describes the Single Agent Baseline (SA), Independent Actor-Critic (IAC), and Independent Actor with Centralized Critic (IACC) models. The chapter also covers hyperparameter tuning and the results of tuning experiments.

Chapter 6 presents the training and testing of the proposed models in a radar tracking scenario. It discusses the performance of the models in terms of Pareto optimality and optimization times. The chapter also examines the sensitivity of the models to environmental parameters and provides a comprehensive discussion of the results.

The performance of the Independent Actor-Critic and Independent Actor with Centralized Critic models is compared against traditional genetic algorithms and the single agent baseline. The results indicate that both IAC and IACC models outperform the traditional methods in terms of probability of detection, waveform duration, and optimization speed. The IAC model shows robust performance across different environmental conditions, finding good waveforms while also being the fastest to converge due to its stable policy gradients.

The IACC model, specifically with the Multi-Layer Perceptron architecture, as well as the IAC model demonstrate superior performance in achieving optimal radar waveform parameters quickly. However, the Graph Neural Network based IACC model, while theoretically promising, did not provide the expected improvements and requires further refinement. The discussion emphasizes that while centralized critics improve robustness and coordination among agents, the choice of architecture significantly impacts the overall performance. The ability of centralized critics to give better value function estimates is thought to be the main cause of improvement in robustness over decentralized critics. However, decentralized critics are more scalable and converge faster.

Overall, the evaluation highlights the effectiveness of MARL approaches in optimizing radar waveforms, showcasing their potential for real-time applications in dynamic and uncertain environments. Not only do we show that RL approaches outperform heuristic methods such as genetic algorithms, but also that posing the problem as a multi-agent optimization problem results in a significant increase in performance. Furthermore, GNNs provide several advantages, the method of integrating domain knowledge in the centralized critic did not result in any improvement over the other MARL methods.

We hope that the insights derived from this work will help the field of radar waveform optimization toward embracing more machine learning techniques and build confidence in their ability to be deployed in practice. This thesis lays a solid foundation for the use of MARL and GNNs in the field of radar waveform optimization, demonstrating their potential to significantly enhance performance and adaptability. The findings encourage further exploration into more adaptive algorithms and the integration of additional environmental factors, ultimately contributing to the broader field of reinforcement learning and radar technology. We believe that continued innovation and interdisciplinary approaches will drive the development of more efficient and effective radar systems, capable of meeting the complex demands of modern applications.

7.2. Future Work

7.2.1. Increasing the number of agents

Due to the limited computational resources and time constraints, the number of agents in the system was kept low. However, there are many other figures of merit we can optimize for, such as autocorrelation, mean probability of detection over the tracking area, and so on. Increasing the number of agents would also make the benefits of the GNNs in terms of scalability more apparent, which could make the IACC (GNN) the only viable option, due to the poor scaling of the IACC (MLP) model. The IAC model could also be used as it scales linearly with the number of agents, but we would lose the benefits of having a centralized critic.

7.2.2. Variable burst budget

The current action space of the model only allows for a fixed burst budget. In realistic scenarios, the number of bursts is not fixed and influences the probability of detection by increasing the energy on the target, while simultaneously increasing the waveform duration. A way of implementing this would be to define the maximum number of bursts to be sent in practice, which is never more than 15-20, and then incorporate a one-dimensional mask of the same length in the action space of the agents. The mask is then used to select which bursts are sent as part of the waveform and which are not. Preliminary experiments suggest this would improve performance.

7.2.3. Improved integration of domain knowledge

Even though adding radar domain knowledge in the graph construction process did not yield any benefits, we still believe that with further refinement it could lead to promising results. In this work, we connected bursts that had dissimilar parameters, with the assumption that their PRFs and pulse lengths would be different enough to cover blind range-doppler areas. However, that is not guaranteed and one could compute the range-doppler diagrams defined by these parameters and only connect bursts where these zones are actually minimized by a sufficient amount. If we take into account the low number of bursts and agents in this work, the graph only amounts to 15-20 bursts per iteration. It is possible that once the number of agents increases, the added benefit of these connections will also become more apparent, as opposed to an MLP centralized critic which compares all bursts with each other.

Bibliography

[1] J. R. Boyd, "Destruction and creation," 1976. [Online]. Available: https://api.semanticscholar.org/CorpusID:9984185.

[2] C. A. Mohr, P. M. McCormick, C. Topliff, S. Blunt, and J. M. Baden, "Gradient-based optimization of pcfm radar waveforms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, pp. 935–956, 2021. DOI: 10.1109/TAES.2020.3037403.

[3] L. Patton and B. Rigling, "Phase retrieval for radar waveform optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, pp. 3287–3302, 2012. DOI: 10.1109/TAES.2012.6324705.

[4] E.-H. Kim, S.-B. Kim, S.-S. Han, S.-J. Shin, and S. Oh, "Design of polyphase codes using simulated annealing," *The Journal of Korean Institute of Electromagnetic Engineering and Science*, 2020. DOI: 10.5515/kjkiees.2020.31.4.383.

[5] W. Li and X. Liu, "Radar waveform design based on improved particle swarm optimization algorithm," vol. 12615, pp. 1261523 - 1261523–6, 2023. DOI: 10.1117/12.2673899.

[6] M. Xia, S. Chen, and X. Yang, "New optimization method based on neural networks for designing radar waveforms with good correlation properties," *IEEE Access*, vol. 9, pp. 91 314–91 323, 2021.

[7] C. E. Thornton, M. A. Kozy, R. Buehrer, A. Martone, and K. Sherbondy, "Deep reinforcement learning control for radar detection and tracking in congested spectral environments," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, pp. 1335–1349, 2020. DOI: 10.1109/TCCN.2020.3019605.

[8] M. Coutino and F. Uysal, "Reinforcement learning for radar waveform optimization," in *2023 IEEE Radar Conference (RadarConf23)*, IEEE, 2023, pp. 1–6.

[9] E. C. Farnett, G. H. Stevens, and M. Skolnik, "Pulse compression radar," *Radar handbook*, vol. 2, pp. 10–11, 1990.

[10] H. Thomas and T. Abram, "Stagger period selection for moving-target radars," in *Proceedings of the Institution of Electrical Engineers*, IET, vol. 123, 1976, pp. 195–199.

[11] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012. DOI: 10.1109/TSMCC.2012.2218595.

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

[13] C. Yu, A. Velu, E. Vinitsky, *et al.*, “The surprising effectiveness of ppo in cooperative multi-agent games,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[14] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *Advances in neural information processing systems*, vol. 30, 2017.

[15] A. Das, T. Gervet, J. Romoff, *et al.*, “Tarmac: Targeted multi-agent communication,” in *International Conference on machine learning*, PMLR, 2019, pp. 1538–1546.

[16] H.-R. Lee and T. Lee, “Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy,” in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 1089–1098.

[17] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, “Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 4213–4220.

[18] D. Simões, N. Lau, and L. P. Reis, “Multi-agent actor centralized-critic with communication,” *Neurocomputing*, vol. 390, pp. 40–56, 2020.

[19] W. J. Yun, S. Yi, and J. Kim, “Multi-agent deep reinforcement learning using attentive graph neural architectures for real-time strategy games,” *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2967–2972, 2021. DOI: 10.1109/SMC52423.2021.9658625.

[20] X. Lyu, Y. Xiao, B. Daley, and C. Amato, “Contrasting centralized and decentralized critics in multi-agent reinforcement learning,” *arXiv preprint arXiv:2102.04402*, 2021.

[21] C. S. de Witt, T. Gupta, D. Makoviichuk, *et al.*, *Is independent learning all you need in the starcraft multi-agent challenge?* 2020. arXiv: 2011.09533 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2011.09533.

[22] G. Weathers and E. Holliday, “Group-complementary array coding for radar clutter rejection,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, pp. 369–379, 1983. DOI: 10.1109/TAES.1983.309317.

[23] L. Xiu-you, X. Yonghua, D. Yun-long, and G. Jian, “Constant modulus waveform synthesis based on iterative convex optimization,” 2015. DOI: 10.1049/CP.2015.1069.

[24] A. Maio, Y. Huang, and M. Piezzo, “A doppler robust max-min approach to radar code design,” *IEEE Transactions on Signal Processing*, vol. 58, pp. 4943–4947, 2010. DOI: 10.1109/TSP.2010.2050317.

[25] J. Li, G. Liao, Y. Huang, Z. Zhang, and A. Nehorai, “Riemannian geometric optimization methods for joint design of transmit sequence and receive filter on mimo radar,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 5602–5616, 2020. DOI: 10.1109/TSP.2020.3022821.

[26] E. J. Hughes and C. M. Alabaster, "Medium prf radar prf optimisation using evolutionary algorithms," in *Proceedings of the 2003 IEEE Radar Conference (Cat. No. 03CH37474)*, IEEE, 2003, pp. 192–197.

[27] C. T. Capraro, I. Bradaric, G. T. Capraro, and T. K. Lue, "Using genetic algorithms for radar waveform selection," in *2008 IEEE Radar Conference*, 2008, pp. 1–6. DOI: 10.1109/RADAR.2008.4720947.

[28] B. Liu, Z. He, and Q. He, "Optimization of orthogonal discrete frequency-coding waveform based on modified genetic algorithm for mimo radar," in *2007 International Conference on Communications, Circuits and Systems*, 2007, pp. 966–970. DOI: 10.1109/ICCCAS.2007.4348208.

[29] G. Lellouch, A. K. Mishra, and M. Inggs, "Stepped ofdm radar technique to resolve range and doppler simultaneously," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 937–950, 2015. DOI: 10.1109/TAES.2014.130753.

[30] J. Hu, Z. Wei, Y. Li, H. Li, and J. Wu, "Designing unimodular waveform (s) for mimo radar by deep learning method," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 2, pp. 1184–1196, 2020.

[31] J. Boubin, A. M. Jones, and T. Bihl, "Neurowav: Toward real-time waveform design for vanets using neural networks," in *2019 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2019, pp. 1–4.

[32] M. Xia, S. Chen, and X. Yang, "New optimization method based on neural networks for designing radar waveforms with good correlation properties," *IEEE Access*, vol. 9, pp. 91 314–91 323, 2021.

[33] P. John-Baptiste, G. E. Smith, A. M. Jones, and T. Bihl, "Rapid waveform design through machine learning," in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, IEEE, 2019, pp. 659–663.

[34] R. W. Gerchberg, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, Jan. 1972. [Online]. Available: https://www.scinapse.io/papers/1484412996.

[35] Q. Wang, Y. Qiao, and L. Gao, "A cognitive radar waveform optimization approach based on deep reinforcement learning," *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, pp. 1–6, 2019. DOI: 10.1109/ICSIDP47821.2019.9173348.

[36] C. E. Thornton, R. Buehrer, H. S. Dhillon, and A. Martone, "Universal learning waveform selection strategies for adaptive target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, pp. 5798–5814, 2022. DOI: 10.1109/TAES.2022.3181554.

[37] G. Rens and J.-F. Raskin, "Learning non-markovian reward models in mdps," *arXiv preprint arXiv:2001.09293*, 2020.

[38] T. Komeda, M. Takagi, *et al.*, "Reinforcement learning in non-markovian environments using automatic discovery of subgoals," in *SICE Annual Conference 2007*, IEEE, 2007, pp. 2601–2605.

[39] G. E. Smith and T. J. Reininger, "Reinforcement learning for waveform design," in *2021 IEEE Radar Conference (RadarConf21)*, IEEE, 2021, pp. 1–6.

[40] L. Wang, S. Fortunati, M. Greco, and F. Gini, "Reinforcement learning-based waveform optimization for mimo multi-target detection," *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 1329–1333, 2018. DOI: 10.1109/ACSSC.2018.8645304.

[41] A. M. Ahmed, A. Ahmad, S. Fortunati, A. Sezgin, M. Greco, and F. Gini, "A reinforcement learning based approach for multitarget detection in massive mimo radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, pp. 2622–2636, 2020. DOI: 10.1109/TAES.2021.3061809.

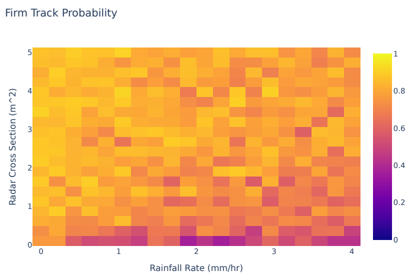
[42] Q. Yang, Z. Han, H. Wang, J. Dong, and Y. Zhao, "Radar waveform design based on multi-agent reinforcement learning," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 35, 2159035:1–2159035:17, 2021. DOI: 10.1142/S0218001421590357.

[43] S. Chen, J. Dong, P. Ha, Y. Li, and S. Labi, "Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, pp. 838–857, 2021. DOI: 10.1111/mice.12702.

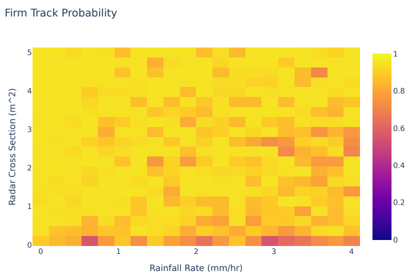
[44] M. Vincent, A. El Fallah Seghrouchni, V. Corruble, N. Bernardin, R. Kassab, and F. Barbaresco, "Multi-objective reinforcement learning in factored mdps with graph neural networks," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 2833–2835.

A

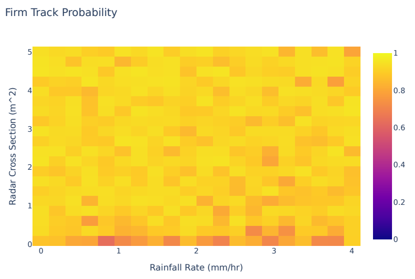
Sensitivity to environmental parameters - average waveform performance



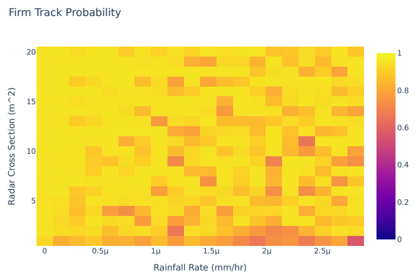
(a) Single Agent Baseline



(b) IAC

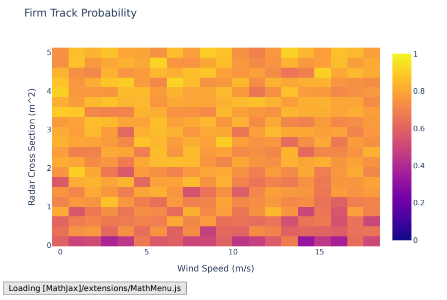


(c) IACC (MLP)

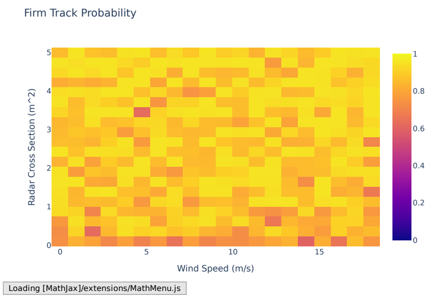


(d) IACC (GNN)

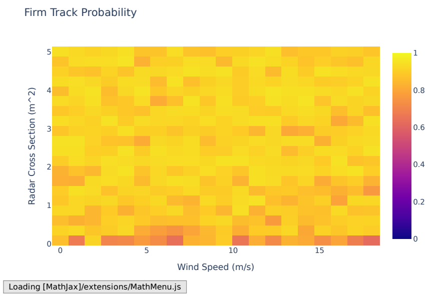
Figure A.1: Firm track probability for radar cross-section versus rainfall rate



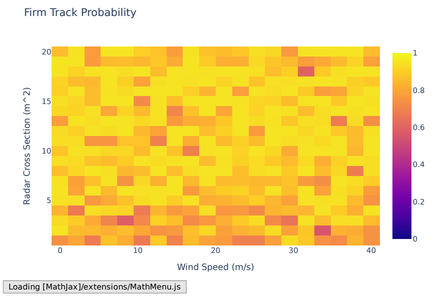
(a) Single Agent Baseline



(b) IAC

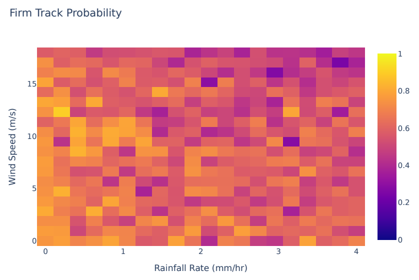


(c) IACC (MLP)

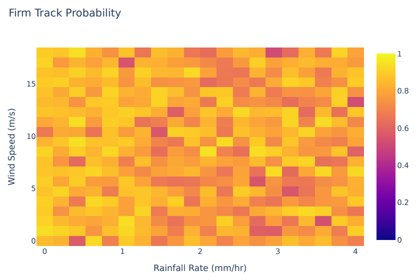


(d) IACC (GNN)

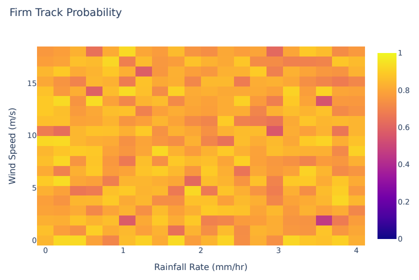
Figure A.2: Firm track probability for radar cross-section versus wind speed



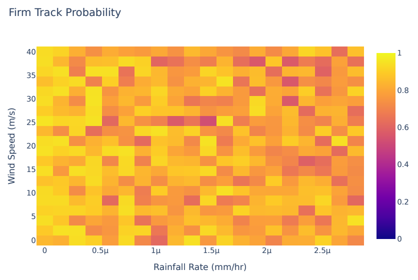
(a) Single Agent Baseline



(b) IAC

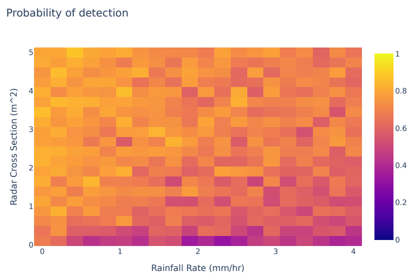


(c) IACC (MLP)

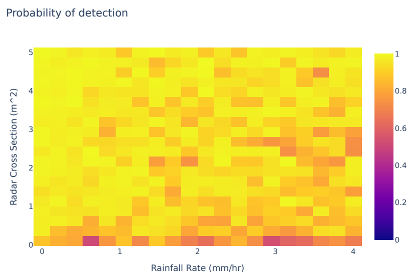


(d) IACC (GNN)

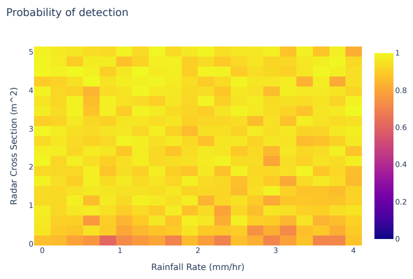
Figure A.3: Firm track probability for wind speed versus rainfall rate



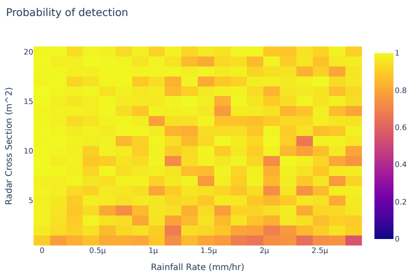
(a) Single Agent Baseline



(b) IAC

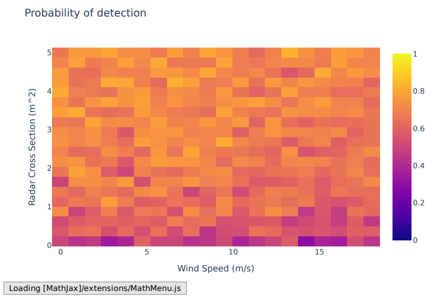


(c) IACC (MLP)

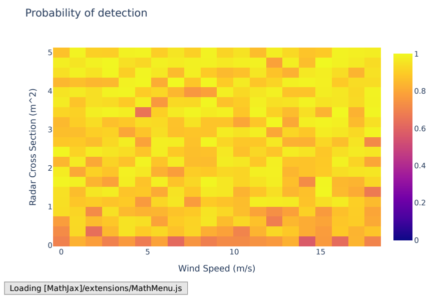


(d) IACC (GNN)

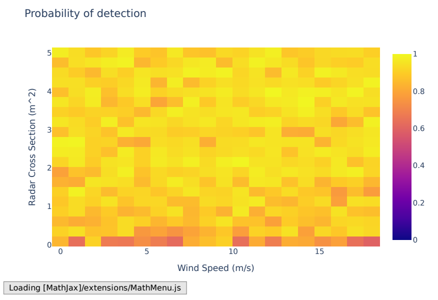
Figure A.4: Probability of detection for radar cross-section versus rainfall rate



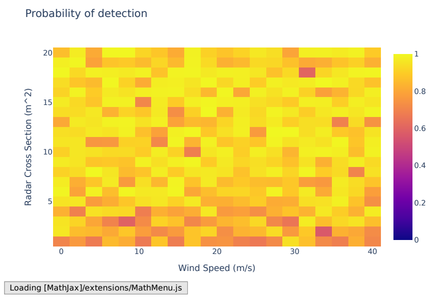
(a) Single Agent Baseline



(b) IAC

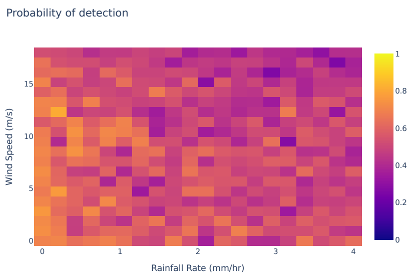


(c) IACC (MLP)

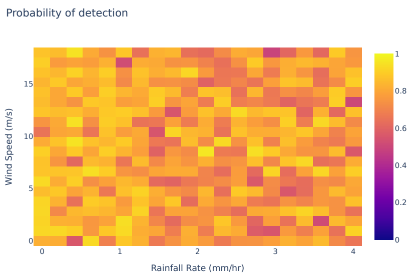


(d) IACC (GNN)

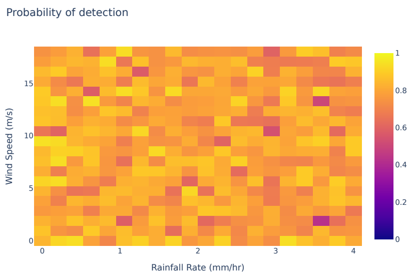
Figure A.5: Probability of detection for radar cross-section versus wind speed



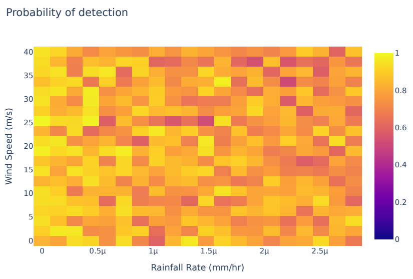
(a) Single Agent Baseline



(b) IAC

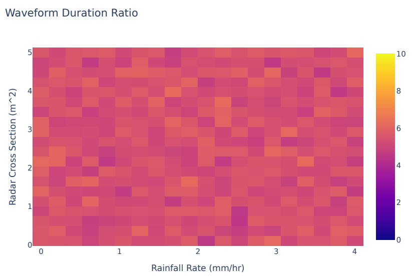


(c) IACC (MLP)

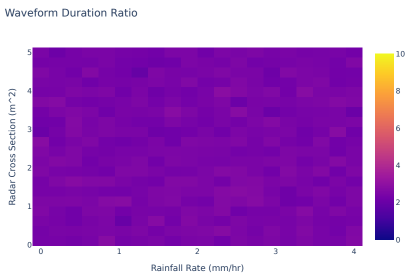


(d) IACC (GNN)

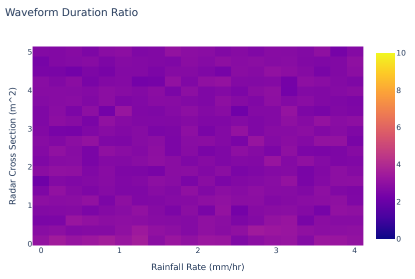
Figure A.6: Probability of detection for wind speed versus rainfall rate



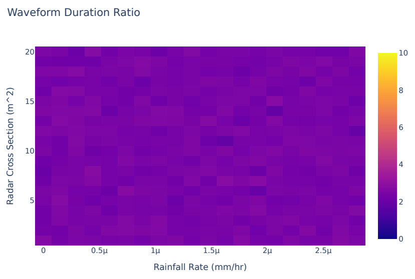
(a) Single Agent Baseline



(b) IAC

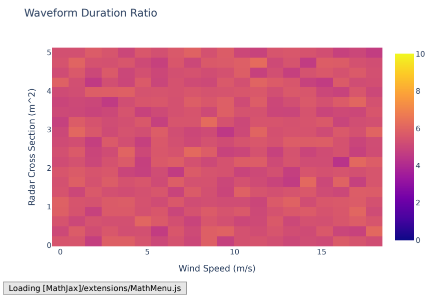


(c) IACC (MLP)

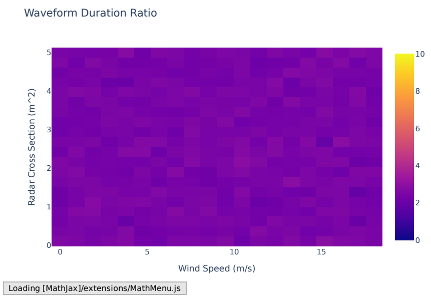


(d) IACC (GNN)

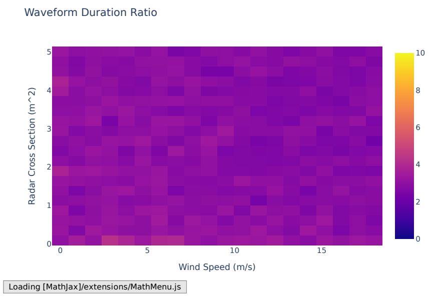
Figure A.7: Waveform duration ratio for radar cross-section versus rainfall rate



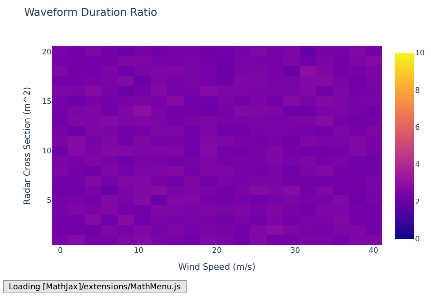
(a) Single Agent Baseline



(b) IAC

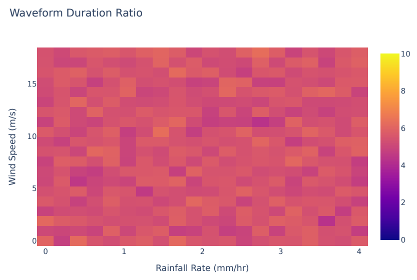


(c) IACC (MLP)

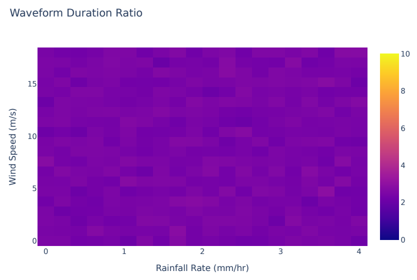


(d) IACC (GNN)

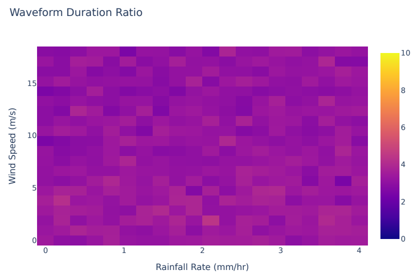
Figure A.8: Waveform duration ratio for radar cross-section versus wind speed



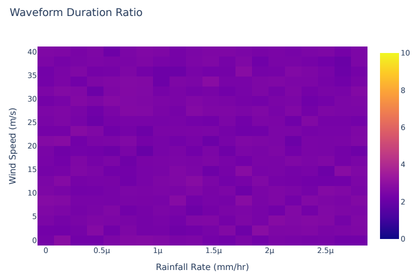
(a) Single Agent Baseline



(b) IAC



(c) IACC (MLP)

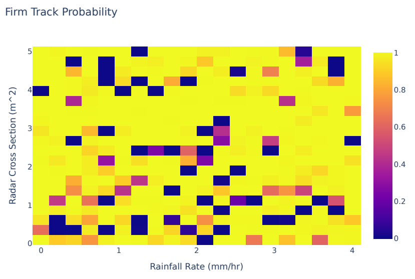


(d) IACC (GNN)

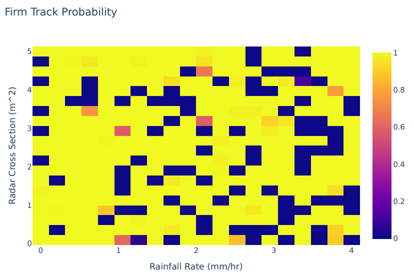
Figure A.9: Waveform duration ratio for wind speed versus rainfall rate

B

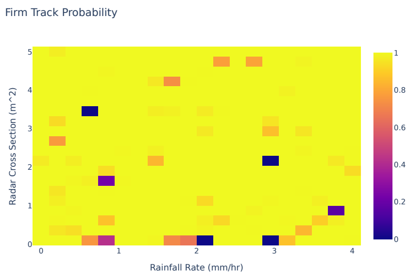
Sensitivity to environmental parameters - best waveform performance



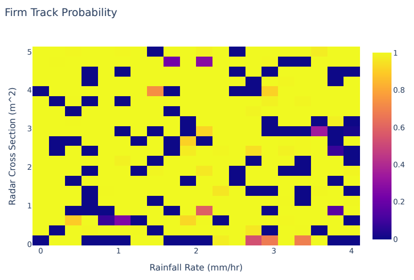
(a) Single Agent Baseline



(b) Multi-Agent Baseline

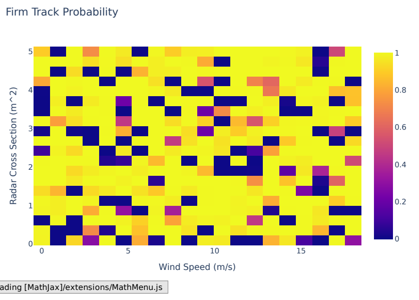


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)

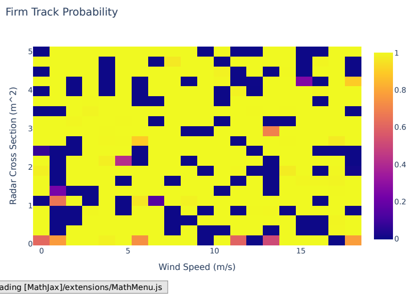


(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

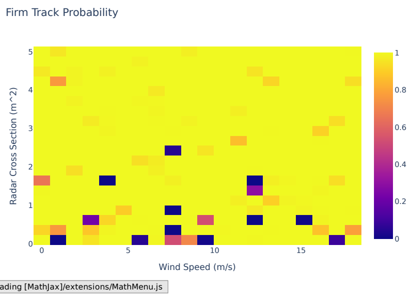
Figure B.1: Firm track probability for radar cross-section versus rainfall rate



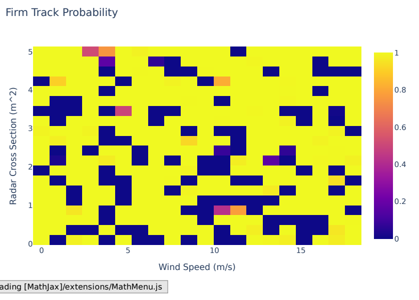
(a) Single Agent Baseline



(b) Multi-Agent Baseline

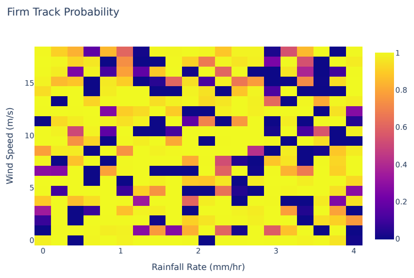


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)

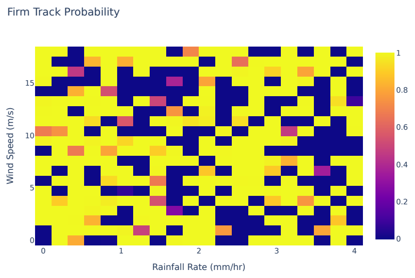


(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

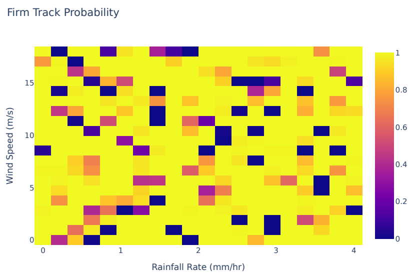
Figure B.2: Firm track probability for radar cross-section versus wind speed



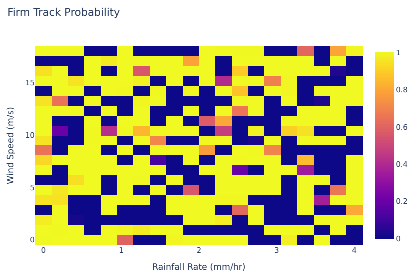
(a) Single Agent Baseline



(b) Multi-Agent Baseline

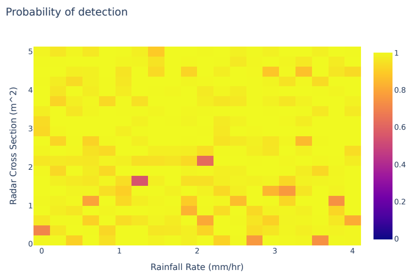


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)

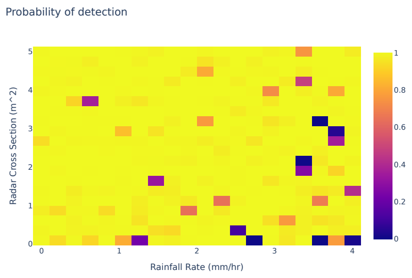


(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

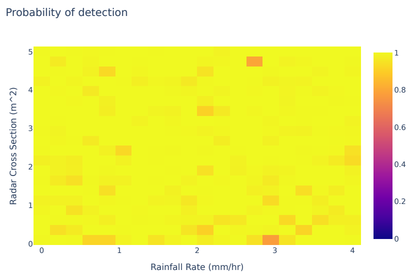
Figure B.3: Firm track probability for wind speed versus rainfall rate



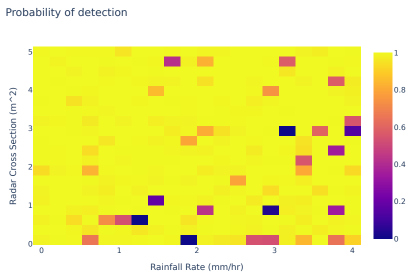
(a) Single Agent Baseline



(b) Multi-Agent Baseline

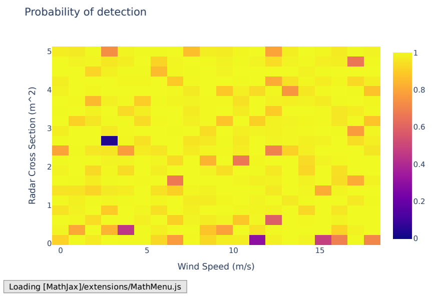


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)



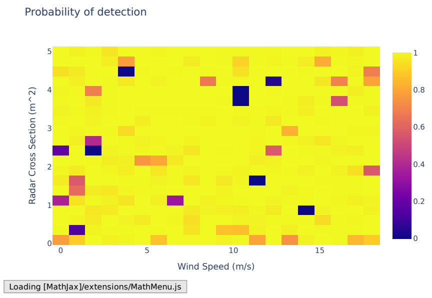
(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

Figure B.4: Probability of detection for radar cross-section versus rainfall rate



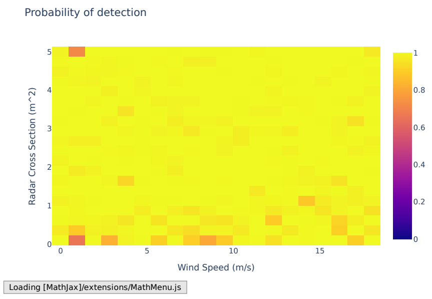
Loading [MathJax]/extensions/MathMenu.js

(a) Single Agent Baseline



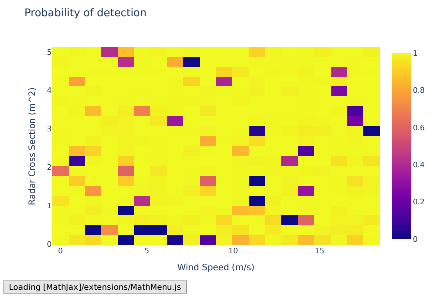
Loading [MathJax]/extensions/MathMenu.js

(b) Multi-Agent Baseline



Loading [MathJax]/extensions/MathMenu.js

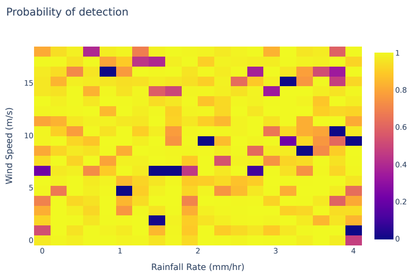
(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)



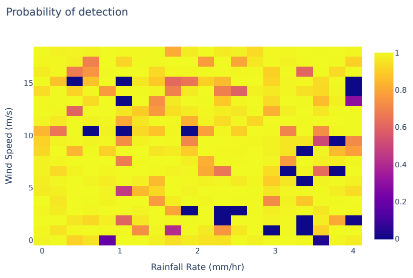
Loading [MathJax]/extensions/MathMenu.js

(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

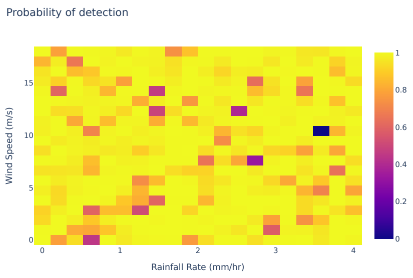
Figure B.5: Probability of detection for radar cross-section versus wind speed



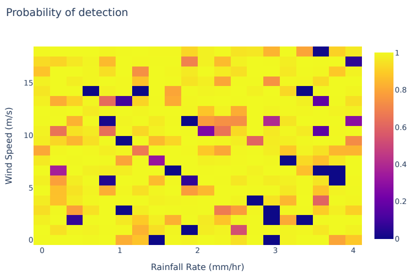
(a) Single Agent Baseline



(b) Multi-Agent Baseline

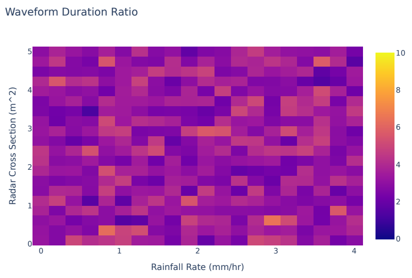


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)

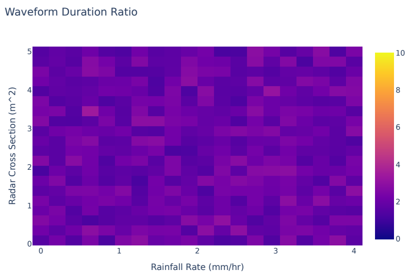


(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

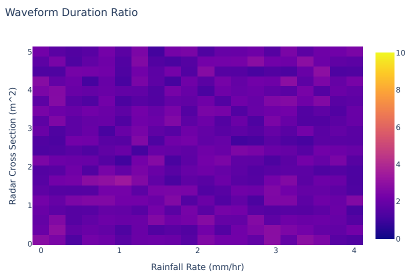
Figure B.6: Probability of detection for wind speed versus rainfall rate



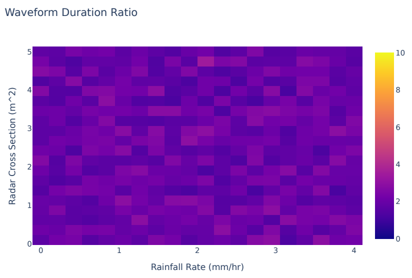
(a) Single Agent Baseline



(b) Multi-Agent Baseline

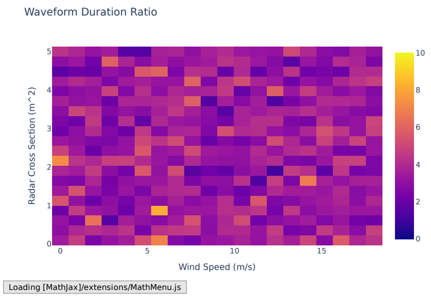


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)

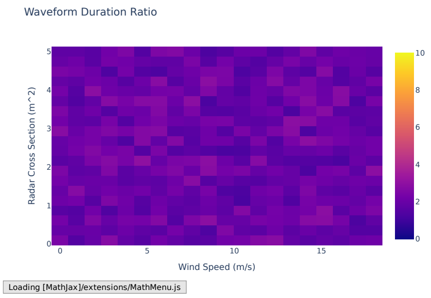


(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

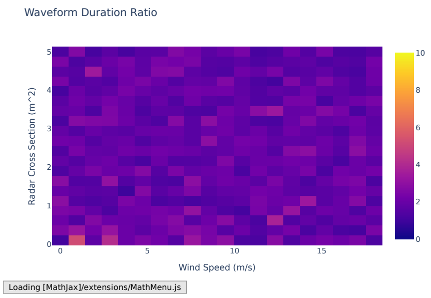
Figure B.7: Waveform duration ratio for radar cross-section versus rainfall rate



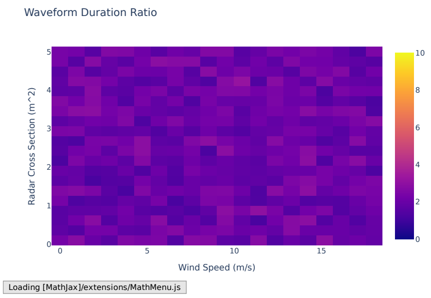
(a) Single Agent Baseline



(b) Multi-Agent Baseline

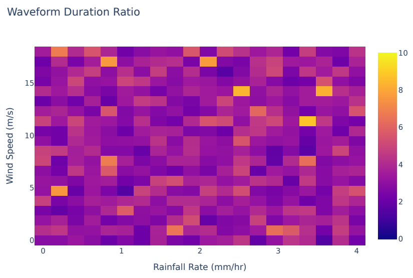


(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)

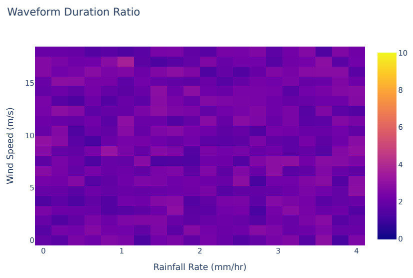


(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

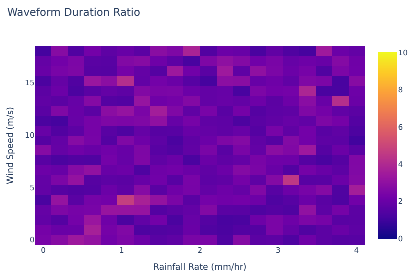
Figure B.8: Waveform duration ratio for radar cross-section versus wind speed



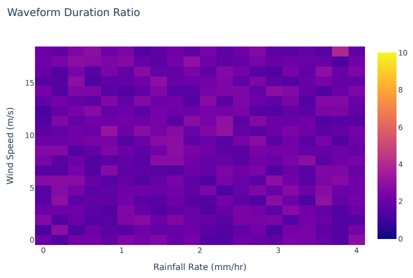
(a) Single Agent Baseline



(b) Multi-Agent Baseline



(c) Multi-Agent Baseline with a Centralized Critic (Fully Connected Network)



(d) Multi-Agent Model with a GNN Centralized Critic (Threshold Graph)

Figure B.9: Waveform duration ratio for wind speed versus rainfall rate