
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**OPTIMAL NAVAL MOVEMENT SIMULATION WITH REINFORCEMENT LEARNING AI AGENTS**

by

Joseph R. Coble

June 2023

Thesis Advisor: Armon C. Barton
Co-Advisor: Christian J. Darken

**Approved for public release. Distribution is unlimited.**

*This project was funded in part by the NPS Naval Research Program.*

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | *Form Approved OMB No. 0704-0188* |
|---|---|



| **1. AGENCY USE ONLY** *(Leave blank)* | **2. REPORT DATE** June 2023 | **3. REPORT TYPE AND DATES COVERED** Master's thesis |
|---|---|---|

| **4. TITLE AND SUBTITLE** OPTIMAL NAVAL MOVEMENT SIMULATION WITH REINFORCEMENT LEARNING AI AGENTS | **5. FUNDING NUMBERS** NPS-23-N059-C |
|---|---|
| **6. AUTHOR(S)** Joseph R. Coble | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This project was funded in part by the NPS Naval Research Program.

| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release. Distribution is unlimited. | **12b. DISTRIBUTION CODE** A |
|---|---|

**13. ABSTRACT (maximum 200 words)**

As the U.S. Navy and its allies strive to ensure freedom of the seas, the need for effective strategies in naval engagements is paramount. Despite expectations, instances such as the USS John McCain and USS Fitzgerald have shown that identifying advantageous moves in every interaction can be challenging. Leveraging advancements in machine learning (ML) and artificial intelligence (AI), this study developed a simulation-based program that applies reinforcement learning (RL) to naval scenarios. The program, an adaptation of an existing land-based wargaming simulation, Atlatl, was designed to identify efficient movements for own forces in six scenarios. Evaluations of Deep Q-Network (DQN), Monte Carlo tree search (MCTS), and AlphaStar AI agents across various scenarios revealed that DQN and MCTS agents were able to identify superior strategies, with DQN demonstrating consistently high scores and outperforming human players in some scenarios. AlphaStar showed fewer promising results but provided insight into how it can be altered for better results in the future. These findings underscore the potential of AI as a decision aid in naval operations, contributing to enhanced decision-making in the U.S. Navy. Future research is recommended to further explore this potential.

| **14. SUBJECT TERMS** reinforcement learning, Monte Carlo tree search, MCTS, artificial intelligence, AI, machine learning, ML, Deep Q-Network, DQN | **15. NUMBER OF PAGES** 95 |
|---|---|
| | **16. PRICE CODE** |

| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |
|---|---|---|---|

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release. Distribution is unlimited.**

# OPTIMAL NAVAL MOVEMENT SIMULATION WITH REINFORCEMENT LEARNING AI AGENTS

Joseph R. Coble
Lieutenant, United States Navy
BS, University of Central Florida, 2013

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
**June 2023**

Approved by: Armon C. Barton
Advisor

Christian J. Darken
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

As the U.S. Navy and its allies strive to ensure freedom of the seas, the need for effective strategies in naval engagements is paramount. Despite expectations, instances such as the USS John McCain and USS Fitzgerald have shown that identifying advantageous moves in every interaction can be challenging. Leveraging advancements in machine learning (ML) and artificial intelligence (AI), this study developed a simulation-based program that applies reinforcement learning (RL) to naval scenarios. The program, an adaptation of an existing land-based wargaming simulation, Atlatl, was designed to identify efficient movements for own forces in six scenarios. Evaluations of Deep Q-Network (DQN), Monte Carlo tree search (MCTS), and AlphaStar AI agents across various scenarios revealed that DQN and MCTS agents were able to identify superior strategies, with DQN demonstrating consistently high scores and outperforming human players in some scenarios. AlphaStar showed fewer promising results but provided insight into how it can be altered for better results in the future. These findings underscore the potential of AI as a decision aid in naval operations, contributing to enhanced decision-making in the U.S. Navy. Future research is recommended to further explore this potential.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF FIGURES

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| ABM | agent-based modeling |
| AI | Artificial Intelligence |
| CDCM | Coastal Defense Cruise Missile |
| CIC | Combat Information Center |
| DQN | Deep Q-Networks |
| DRL | Deep reinforcement learning |
| MCTS | Monte Carlo tree search |
| ML | Machine Learning |
| MOVES | Modeling Virtual Environments and Simulation |
| MPC | Model Predictive Control |
| NTSB | National Transportation Safety Board |
| OOD | Officer of the Deck |
| PEAS | Performance, Environment, Actuators, Sensors |
| PIM | Plan of Intended Movement |
| PPO | Proximal Policy Optimization |
| RL | reinforcement learning |
| SSOP | Sound Shipboard Operating Principles and Procedures |
| TRPO | Trust Region Policy Optimization |
| UCB | Upper Confidence Bound |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

Thank you to LT Seth Kyler, LT Kevin Tran, LT Ashley Dodd, and LT Michael Williams for helping me study and the friendship through my time at NPS.

My thanks go also to Lt. Col Scott Black, who saved me many hours by helping me troubleshoot my programs so I could stay on track.

Samuel McKeown, Whitney Hough, Samantha Markum, Luke Taylor, Olivia, and Isabelle Hack—I thank you for the laughs and stress relief.

I thank my advisors, Dr. Armon Barton and Dr. Chris Darken, who each allowed me to explore their knowledge and provide clarifying oversight when I was unable to see the forest through the trees.

My sisters, Kathryn Shepherd and Lauren Garner, and their families, Justin, Rhett, Sawyer, Barrett, Korey, Eli, Saylor, and Griffin have provided a solid bedrock of their support for me in this endeavor: thank you.

Thanks to Rodney and Jacqueline Franklin: your selfless assistance has been constant, helping Shyenne and me in every aspect of our lives.

To my parents, Robert and Jean Coble, who have been my lifelong cheerleaders, I give my gratitude and love. From as early as I can remember, you have fed my interest in computers and stood by me as I pursued this dream. Your belief in me has been a guiding force.

Finally, a profoundly heartfelt thanks to my wife, Shyenne. Since day one, she has been my greatest supporter. Her patience and understanding have allowed me to cloister myself in the library, buried in classwork, while she managed our world outside. Her strength, love, and unwavering faith in me have propelled me forward. Shyenne, I wouldn't be who I am, or where I am, without you.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

As the U.S. Navy and its allies maintain presences worldwide to ensure freedom of the seas, it is increasingly necessary to identify the optimal strategy in engagements with other vessels. The expectation is that the captains and officers of our warships can correctly identify the optimal moves of every interaction. This thesis aims to investigate the development of a scenario-based framework in a Naval-focused version of Atlatl, a wargaming simulation, that enables Artificial Intelligence (AI) agents to determine the most effective strategies and actions for specific situations. The objective is to establish a foundational tool capable of accommodating a wide range of variables, facilitating the exploration of novel approaches to address emerging challenges. Using this framework, U.S. Navy can test and evaluate potential solutions without incurring the risks and costs of real-world experimentation, ultimately improving decision-making and strategic planning in various operational contexts.

## A. NAVAL SCENARIOS

Naval interactions happen all the time, with very minimal margins of error being acceptable. The ability for a ship captain, or Officer of the Deck (OOD), to identify the optimal path, not just for their ship but the adversary as well, can easily result in lives lost or not. These decisions are often made by qualified officers who stand in for the captain when they are away. The amount of experience between all these decision-makers varies considerably based on how long they have been navigating ships and training. This part of the paper will describe historic Naval interactions and how the lack of support resulted in major casualties.

### 1. USS Fitzgerald (DDG-62) and ACX Crystal Collision

On June 17, 2017, the USS Fitzgerald Guided Missile Destroyer collided with the ACX Crystal Container vessel [1]. Figure 1 shows how a minor course change was all that was needed for these two vessels to collide with each other in the middle of the night. This accident resulted in the loss of seven service members on board. Throughout the investigation, the National Transportation Safety Board (NTSB) found that poor

communication and competence among the crew of the USS Fitzgerald were the key issues that led to the collision [1]. Two of the leading issues identified were insufficient training and crew fatigue. The OOD did not use the appropriate resources, such as the Combat Information Center (CIC), which is designed to provide the OOD situational backup, resulting in erroneous decisions that directly contributed to the collision. The second issue of crew fatigue was created by the condensed schedule the USS Fitzgerald had been on over the last few days, leading to "diminished alertness and degraded performance" [1].

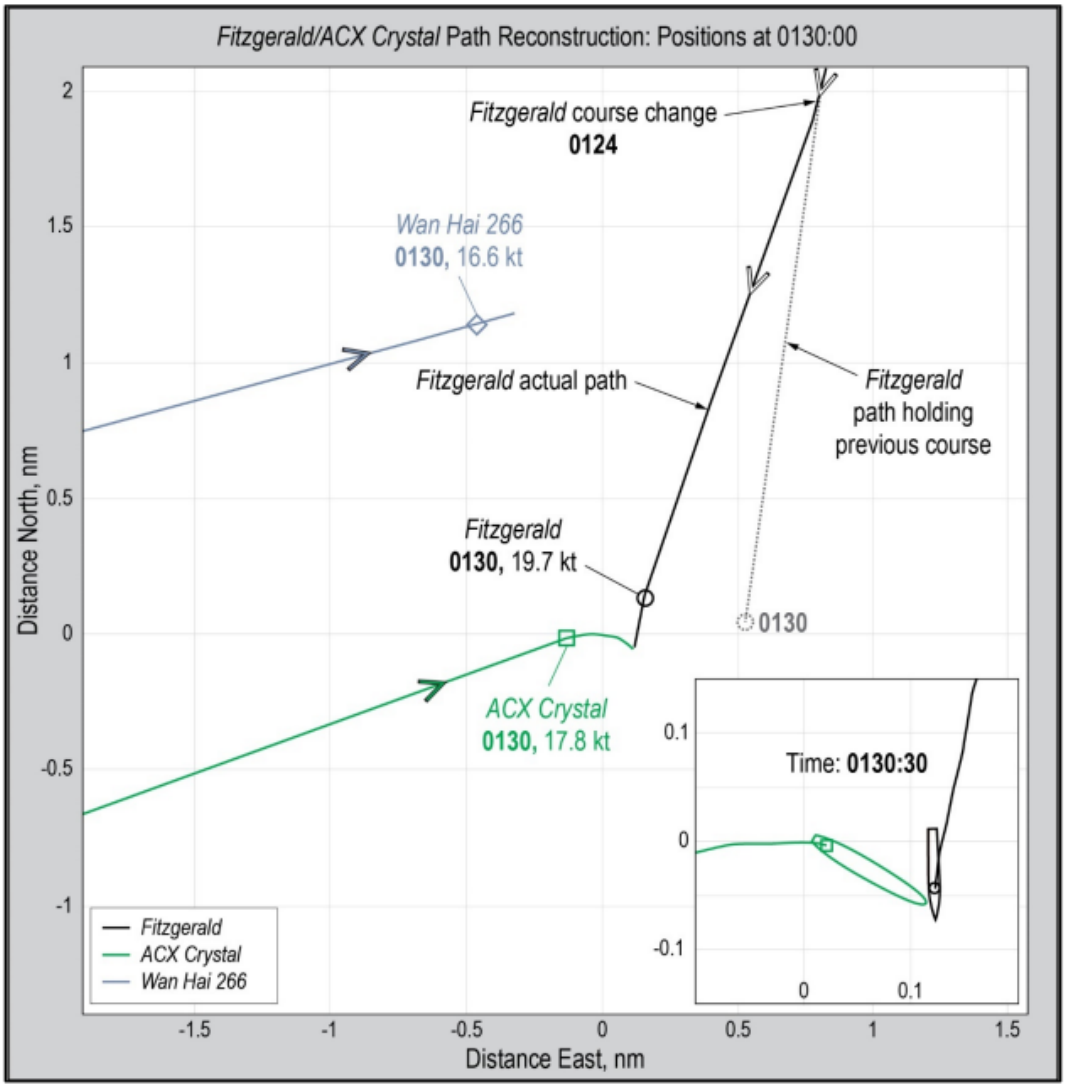


Figure 1. Collision of USS FITZGERALD. Source: [20].

## 2. USS John S. McCain (DDG-56) and Tanker Alnic MC Collision

On August 21, 2017, the USS John S. McCain Guided Missile Destroyer collided with the Tanker Alnic MC. This collision (see Figure 2) occurred in the Singapore Strait, one of the busiest straits in the world. In 2016, more than 80,000 vessels with over 300 gross tons sailed this strait [2]. This collision occurred differently than the USS Fitzgerald but had similar safety issues contributing to the result. The U.S. Navy's records state that leading up to the collision, the USS John S. McCain crew averaged about 4.9 hours of rest in 24 hours [2]. As indicated in the report, this fatigue and inadequate watch training led to poor decision-making and failure to conduct proper procedures in a restricted maneuver environment. In restricted maneuver environments, vessels cannot change their course and heading considerably. These poor decisions resulted in the loss of ten servicemen and women [2].

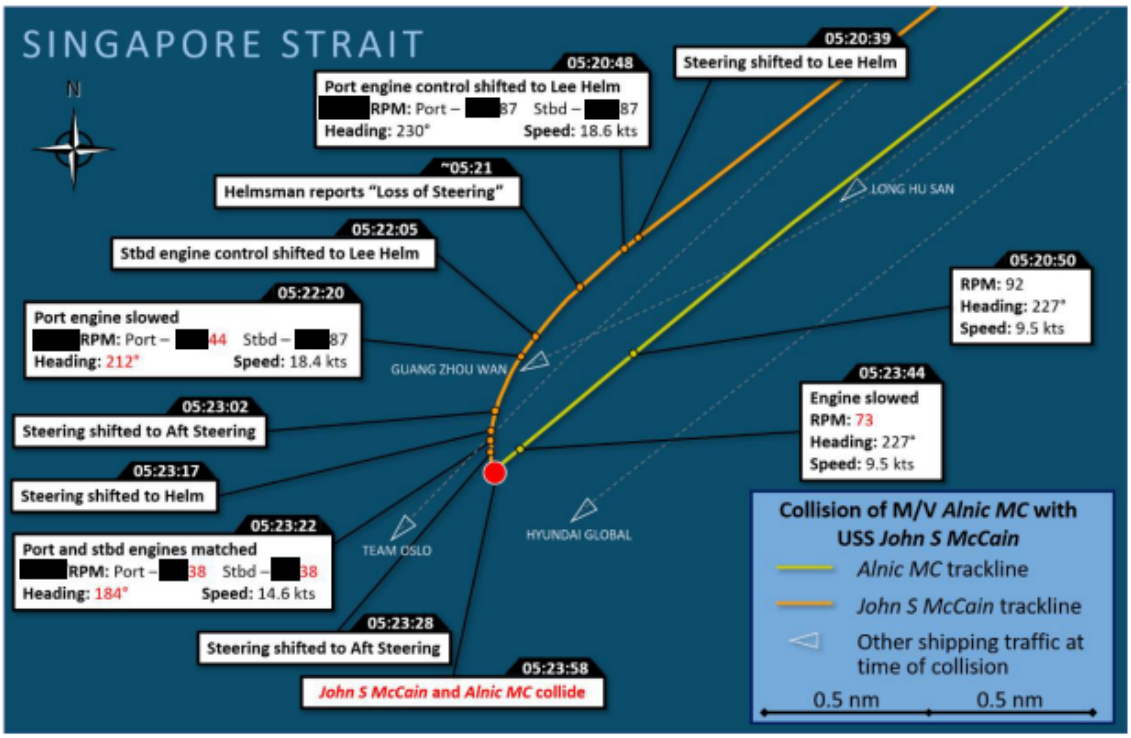


Figure 2. USS JOHN S. MCCAIN Collision. Source: [2].

### 3. Forceful Backup

In the U.S. Navy Sound Shipboard Operating Principles and Procedures (SSOP), one of the core principles is Forceful Backup. This is the concept of properly qualified personnel on station attentive to the ongoing scenario [3]. The ability of knowledgeable entities to anticipate future issues and present correcting actions or directing attention to these issues is key to how the Navy will optimize its actions while minimizing risk. The ability to have AI take the role of these expertly trained supervisors presents a backup that will not be compromised by fatigue or lack of judgment. This is not to remove the human in the loop of decision-making aboard Naval vessels but to augment their decision-making with solid informational backup. This concept of AI-supported decision-making is not limited to Naval vessel navigation but can be expanded to wargaming scenarios where the commanders at all levels can get the support of engagements with other entities, friends, or adversarial.

## B. WARGAMING

The utilization of wargaming in the U.S. Navy has been used for over 100 years to inexpensively test theories and lessons outside of war environments [4]. The Naval War College's wargaming department is dedicated to conducting these games to provide insight into how future battles will be played out [5]. The desire from leadership service-wide to implement AI into these wargaming scenarios has been increasing as the technology advances and decreases in cost. These wargaming scenarios are often built as a battle between two teams. These battles provide a better environment for AI agent development due to their zero-sum relationship, where one side's victory is the other side's loss, which provides more distinct feedback in the learning process. This thesis uses these types of battle scenarios to show that in the highest risk scenarios, AI can provide forceful backup to the commanders of the ships. The ability to replace or augment these human actors with AI agents presents an opportunity to create tools that could be used in scenarios, such as congested area navigation or combat, to enhance decision-making across the fleet. This thesis will take the wargaming concepts and

implement AI agents into given scenarios to determine if optimal outcomes are achievable.

## C. PROBLEM STATEMENT

Over the years, there have been multiple iterations of wargaming that implement different types of AI to achieve the end state of optimal performance. The issue is that these wargaming scenarios have been predominately done with land forces and broad scope. This thesis looks at the use of computer based wargaming and the ability for RL to create AI agents that can find the optimal outcome of a given scenario. This ability taken from an aggression-based simulation can be then used to solve non-aggressive simulations such as navigation backup and demonstrate the usability of AI in the future of all naval interactions.

## D. SCOPE

This thesis takes the baseline setup of Atlatl and modifies it to work with Naval units in six specific scenarios. Once these six scenarios are created, they will be tested with different AI agents to identify if the optimal movements can be achieved. The AI agents that will be used will consist of rules-based, model-based, and values-based methods. The scenarios will be limited to Destroyer ships and artillery units used as coastal defense units.

## E. BENEFITS OF STUDY

The successful development of the scenario-based framework proposed in this thesis promises significant benefits for U.S. Navy research in AI-assisted decision-making. The framework will enable more effective decision-making and strategic planning by empowering AI agents to identify optimal strategies in various situations. Additionally, the adaptability of the tool to incorporate diverse variables ensures its relevance in addressing an array of emerging challenges. The military can identify the most effective course with minimal cost in testing potential solutions without real-life risk. This research also will lead the military closer to producing a second set of eyes that

can provide unyielding support to decision-makers, from where to send troops in a battle to appropriately navigating a congested waterway.

## F. CHAPTER ORGANIZATION

### 1. Chapter I: Introduction

Chapter I introduces the thesis and addresses the initial reasons for its creation. It then defines the problem and scope of the thesis. Finally, it provides the benefits of the study and organizational layout for the entire thesis.

### 2. Chapter II: Background

This chapter details the history of wargaming and the introduction of AI. It examines the types of wargaming that are used. It also explains an AI and RL and how this technology is used in wargaming.

### 3. Chapter III: Methodology and Implementation

This chapter explains the program used and how it was modified to address naval-centric issues. The chapter also details how the scenarios were created. Finally, this chapter explains the implementation of the AI agents and tests them in the scenarios.

### 4. Chapter IV: Results

Chapter IV presents the findings of the work in Chapter III. The results are expressed in training evaluations and scoring of the agents against specific adversary agents.

### 5. Chapter V: Conclusions and Future Work

Chapter V reviews the results in Chapter IV and the conclusions drawn from it. This chapter then presents future work areas that can expand this area.

# II. BACKGROUND

This chapter will review previous research on Artificial Intelligence (AI) and wargaming. The chapter will first describe what AI is and how the different types of AI that are used today regarding the use of models and model-free agents. This will focus more on reinforcement learning (RL) and its types. The next section of this chapter breaks down wargaming and the types of scenarios that are used to support different outcomes. The final part of this chapter looks at the integration of AI and wargaming and how it has produced significant results in both respective fields.

## A. ARTIFICIAL INTELLIGENCE

AI has revolutionized how wargaming and naval simulations are conducted by allowing the human aspect to either play against a computer or have the computer play against itself. This section will introduce AI and show the different types used to improve wargaming and simulations.

### 1. Intelligent Agents

In "Artificial Intelligence: A Modern Approach (Third Edition)" by Russell and Norvig [6], Intelligent agents are described as computational entities designed to perceive their environment, process the gathered information, and subsequently take actions to achieve their predetermined objectives. These agents use their perceptual input of the environment, called their percept or observation, to assist in the agents' next action. These individual percepts can be put together into a percept or observation sequence that contains all the agent's perceptions of the environment it ever had. The authors categorize agents into four types:

1. Simple reflex agents: These types of agents use their current percept to decide their following action [6]. They do not use any previous information or learn from the environment outside what their immediate percept provides them. This is known as the condition-action rule.

2. Model-based reflex agents: These agents use their percept sequence to assist them in understanding how things work in the environment. This combined with their current percept allows them to make decisions that consider how their action will affect the environment around them.

3. Goal-based agents: These agents use desired objectives to help influence their actions in the environment [6]. They can take the objective and their understanding of their environment to choose actions that will enable them to get to their objective.

4. Utility-based agents: These agents build off the goal-based agents but have a utility function that provides a performance measure to assist in identifying the desirability of different outcomes. This utility function allows the agent to make decisions by evaluating and comparing the expected utility of various actions.

### 2. Problem-Solving Agents

Problem-solving agents are a specific type of intelligent agent that focuses on finding the best solution for a given problem by searching through a space of possible actions or states [6]. These agents employ algorithms and heuristics to navigate complex environments to arrive at their objective. Some of the more common algorithms seen with problem solving agents are breadth-first search and depth-first search. These agents capitalize on the memory they store of the environment around them to identify the most efficient path to achieve their goals.

### 3. Environments

The environments in which agents operate play a critical role in determining their behavior and performance. Characterized by their states, environments are the surroundings that the agent perceives at any given time while it is running. In the book by Russell and Norvig [6], it utilizes the PEAS (Performance, Environment, Actuators, Sensors) framework to describe and analyze agent environments:

1. Performance: This refers to the measures used to evaluate an agent's success in achieving its objectives within a given environment.
2. Environment: This encompasses the external context in which the agent operates, including other agents, objects, and the overall conditions that affect the agent's actions and performance.
3. Actuators: These are moves available to the agent which result in actions taken that could directly influence their environment and follow-on state.
4. Sensors: These are the inputs that allow agents to get an observation of their environment, gathering information for their decision-making processes.

By examining agents and their environments using the PEAS framework, it is possible to gain insights into the interactions between the agents and their surroundings.

## 4. Games

Games and game trees play a crucial role in understanding AI agents' decision-making processes. Games represent well-structured competitive environments in which agents engage other agents or humans, with zero-sum games being the most common type seen in AI [6]. These adversarial engagements result from agents having opposing utility functions, punishing poor scoring actions while promoting beneficial ones. Game trees, on the other hand, are tree-like data structures that illustrate potential action sequences and their associated outcomes in a game [6]. Each node represents a game state, while branches indicate potential moves leading to subsequent states. AI agents employ game trees to explore and evaluate possible actions, ultimately selecting the most optimal course of action to maximize their chances of success or minimize their opponent's. This methodology is employed in various AI techniques, such as minimax and alpha-beta pruning, which gameplaying agents use to effectively navigate complex decision spaces.

### *a. Minimax*

Minimax is a strategy employed by AI agents in adversarial games, where the objective is to minimize the worst-case loss while maximizing the best-case gain [6]. This method involves recursively evaluating the game tree, alternating between minimizing the opponent's score and maximizing the agent's score. The minimax algorithm seeks to predict the opponent's moves, assuming opponent plays optimally, allowing the agent to choose the most optimal course of action to ensure success.

### *b. Alpha-Beta Pruning*

Alpha-Beta Pruning is an optimization technique used to improve the efficiency of the minimax algorithm by pruning branches in the game tree that do not need to be explored [6]. This method reduces the number of nodes that need to be examined, allowing the AI agent to navigate the decision space more effectively. By setting upper and lower bounds for the possible values of a node (alpha and beta), unnecessary branches can be pruned, as they will not contribute to the final decision. This allows the AI agent to search deeper into the game tree and enhance its performance in competitive settings, such as wargaming and other applications.

## 5. Reinforcement Learning

In the context of reinforcement learning (RL), an agent serves as a computational entity that interacts with its environment by executing actions and examining the resulting outcomes, (see Figure 3). The environment encompasses the external setting in which the agent functions, while states signify the distinct conditions of the environment at any given moment. Actions pertain to the choices accessible to the agent, leading to alterations in the environment and subsequent state transitions. Rewards constitute the feedback signals acquired by the agent following an action, providing an indication of the action's desirability [7].

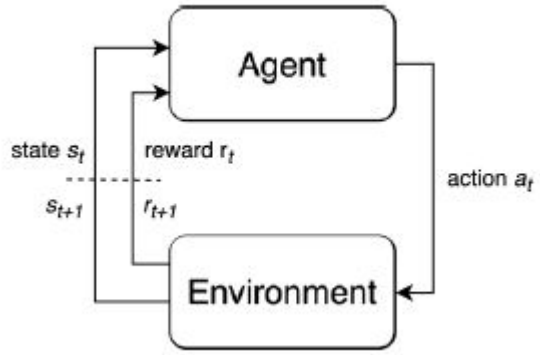


Figure 3. RL Control Loop Diagram. Source: [8]

### *a. Optimizing rewards*

A central aim of Reinforcement Learning (RL) is to maximize the cumulative rewards obtained by an agent during its interactions with the environment [9]. To achieve this, the agent must develop an effective policy that directs its actions across various states, enabling the agent to make well-informed decisions that result in the greatest cumulative rewards. The policy, essentially a mapping from states to actions, is honed through trial and error, as well as the evaluation of the consequences of previous actions. By continually refining its policy, the agent can learn to select actions that yield favorable outcomes, ultimately leading to the highest possible total rewards and successful achievement of its goals.

### *b. Exploration vs. Exploitation*

During this process, the agent faces the exploration versus exploitation dilemma, which requires finding a balance between trying new actions (exploration) and choosing the best-known action (exploitation) for a given state. Two efficient strategies to tackle this challenge are ε-greedy and upper confidence bound (UCB):

5. ε-greedy: This method opts for the best-known action with a probability of 1-ε, and a random action with a probability of ε. The parameter ε controls the exploration-exploitation trade-off, with higher ε values leading to

increased exploration. As the agent acquires more knowledge about the environment, ε can be reduced to prioritize exploitation.

6. Upper Confidence Bound (UCB): UCB computes an upper confidence bound for each action, factoring in both the estimated value of the action and the uncertainty surrounding the estimate. The agent selects the action with the highest upper confidence bound, balancing exploration and exploitation by considering both the potential value and the uncertainty of each action.

## B. DEEP REINFORCEMENT LEARNING ALGORITHMS

Deep reinforcement learning (DRL) algorithms, seen in Figure 4, leverage the power of deep neural networks to handle complex, high-dimensional state spaces in reinforcement learning. These algorithms have enabled significant advancements in AI applications, particularly in areas where traditional reinforcement learning methods struggle to perform. In "Foundations of Deep Reinforcement Learning" by Laura Graesser and Wah Loon Keng [7], DRL algorithms are categorized into four primary types, each with unique characteristics and applications.

### 1. Policy-Based Algorithms

Policy-based algorithms directly optimize the agent's policy, which dictates the actions the agent should take in a given state [7]. These algorithms, such as REINFORCE and Proximal Policy Optimization (PPO), focus on learning the optimal policy without explicitly estimating the value function. Policy-based methods can handle continuous action spaces and are particularly suited for problems where the optimal action is challenging to determine.

### 2. Value-Based Algorithms

Value-based algorithms, like Q-learning and Deep Q-Networks (DQN), seek to estimate the optimal value function, which represents the expected cumulative reward an agent can obtain from a particular state [7]. These algorithms employ deep neural

networks to approximate the value function, allowing the agent to determine the best action in a given state by selecting the action with the highest estimated value. Value-based methods are ideal for discrete action spaces and can often converge faster than policy-based approaches.

### 3. Model-Based Algorithms

Model-based algorithms construct an internal model of the environment, which they use to simulate the environment's dynamics and predict future states and rewards [7]. By planning within this learned model, the agent can improve its policy or value function. Model-based methods, such as Model Predictive Control (MPC) and Monte Carlo tree search (MCTS), offer sample efficiency and can be more robust to function approximation errors. However, learning an accurate model can be challenging, particularly in complex environments.

### 4. Combined Methods

Combined methods, seen in Figure 4, integrate elements from policy-based, value-based, and model-based approaches to harness their respective strengths and overcome their limitations [7]. Algorithms such as Actor-Critic and Trust Region Policy Optimization (TRPO) combine policy and value-based methods, while AlphaZero, developed by DeepMind in 2016 [10], incorporates model-based techniques into policy optimization. Combined methods can provide more stable learning and improved performance across various tasks and environments.

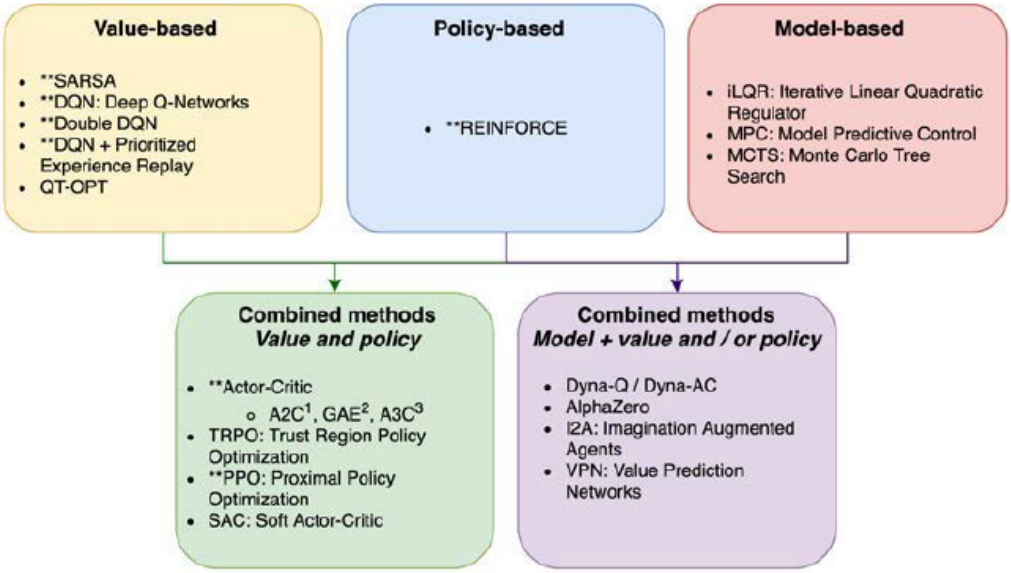


Figure 4. Deep RL Algorithm Families Diagram. Source: [7]

## 5. Q-Learning

Q-Learning is a widely used model-free reinforcement learning algorithm that aims to find the optimal policy for an agent to achieve its goals in a given environment [9]. It is based on estimating the action-value function, which represents the expected cumulative reward of taking a specific action in a given state and following a particular policy thereafter [7]. The algorithm iteratively updates the action-value function using the Bellman equation, eventually converging to the optimal action-value function. Q-Learning is particularly advantageous in situations where the agent has no prior knowledge of the environment and must learn through trial-and-error interactions.

## 6. Deep Q-Networks

Deep Q-Networks (DQNs) are an extension of Q-Learning that employ deep neural networks to approximate the action-value function [9]. By leveraging the expressive power of neural networks, DQNs can handle high-dimensional state spaces and complex problems that traditional Q-Learning cannot efficiently address [7]. Key innovations, such as experience replay and target networks, have been introduced to stabilize the learning process and improve the performance of DQNs [7]. DQNs have

demonstrated remarkable success in various domains, including playing Atari games at a human-level performance.

### 7. Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is a search algorithm that utilizes heuristics and integrates the fundamentals of tree search with Monte Carlo simulations, to facilitate decision-making in complex environments [11]. MCTS iteratively builds a search tree by performing random simulations and incrementally refining the tree based on the results of those simulations. The algorithm consists of four main steps: selection, expansion, simulation, and backpropagation (see Figure 5) [11]. MCTS has been widely applied to various domains, including games and planning, where it has demonstrated significant improvements over traditional search techniques.

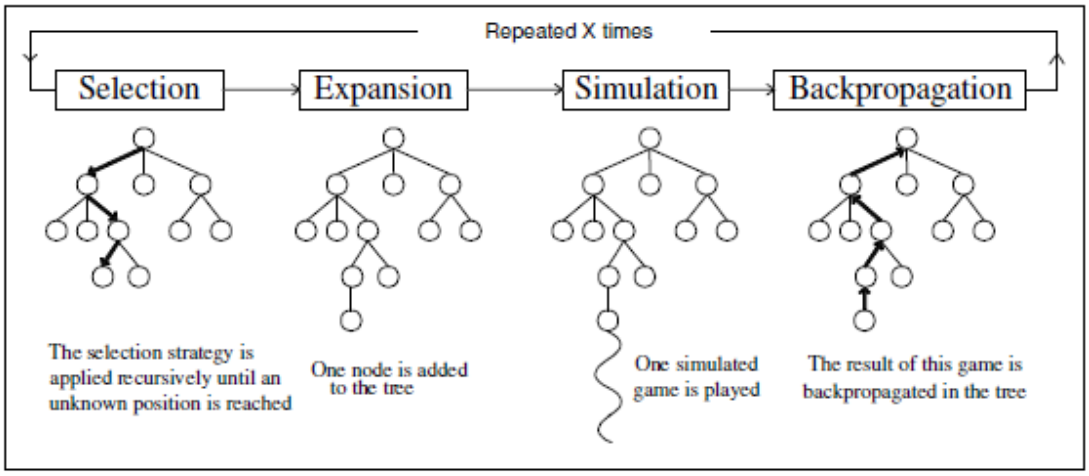


Figure 5. MCTS Loop Diagram. Source [11]

### 8. Alpha Zero

Alpha Zero is a groundbreaking reinforcement learning algorithm developed by DeepMind that achieved superhuman performance in the games of chess, shogi, and Go [10]. It represents a significant departure from traditional game-playing algorithms, as it learns solely through self-play without any reliance on human-generated data or domain-

specific knowledge. Alpha Zero employs a combination of deep neural networks, MCTS, and reinforcement learning to iteratively improve its gameplay. Its success demonstrates the potential of general-purpose learning algorithms to excel in complex and strategic tasks, offering new possibilities for AI research and applications.

## C. WARGAMING TYPES

Wargames play a crucial role in various objectives in defense and military organizations. The following classifications of wargaming, as detailed in the paper "AI and Wargaming" by James Goodman et al. [12], provide a comprehensive understanding of these objectives:

### 1. Planned Force Testing

This is the largest scale of wargaming conducted [12]. These are designed to incorporate multiple commands and how to approach an operation from beginning to end, including supporting functions. This type of testing will typically be representative of full-scale wars. An example of this would be a total U.S. war against some other superpower, requiring significant coordination and force planning. These are seen commonly with the large maps on tables with small 3D units across them.

### 2. Plan Variation Testing

This type of wargaming focuses on individual scenarios defined in more detail and can be repeated multiple times to determine optimal outcomes. These wargames can also identify unique situations that human decision-makers can recreate to explore unique actions and their outcomes [12]. The interesting part of this type of wargaming is that these specific scenarios do not need to be only combat-related but could also be simple interactions between vessels that are not adversarial. This could result in a better Plan of Intended Movement (PIM) generation and, eventually, real-time contact deconfliction.

### 3. Concept/Force Development

These wargames examine a suggested idea or alteration of current capabilities and how this idea or alteration could create a different outcome [12]. This type could also be

used to see how minimal force or equipment is needed to achieve an objective. The tradeoff in minimal forces and equipment could be more risk of loss of units or achieving the objective involved in the scenario.

#### 4. Procurement

The final type of wargaming is hyper-focused on how using different tools or equipment changes the scenario's outcome [12]. Regarding the impact of the scenario, these wargames are tightly controlled in all aspects besides the equipment and how it makes a difference. An example of change in equipment would be using the same scenario but switching out different missiles on the DDGs or other RADAR systems, allowing earlier detection of adversarial units.

### D. ARTIFICIAL INTELLIGENCE IN WARGAMING

In modern times, AI has taken a leading role in many different aspects of human life meant to facilitate advances in those spaces. These advances are seen in the use of AI in wargaming. This technology can upend the status quo of how the military conducts wargames could see advances in tactics development to force optimization on large-scale operations [4]. AI-driven wargames can provide commanders with valuable insights and data-driven recommendations, allowing them to make better-informed decisions during real-world operations [14]. In using AI in wargaming, the simulations can take a more realistic turn as they can represent both sides of the battle with high levels of realism. Some real-world examples of AI applications are seen in games such as Chess, Go, Shogi, and StarCraft II [10][11][14][15]. The AI "Deep Blue" was a supercomputer from IBM that could access all the moves ever performed by all the Grandmasters in the game, computing 200 million moves per second [16][17][18]. It eventually beat the world's greatest grandmaster, Garry Kasparov, in 1997 [16]. This was one of the most public displays of computer AI capabilities that helped highlight the field. The follow on AlphaGo and further on AlphaZero were able to take the concept of reinforcement learning (RL) under the umbrella of AI and create agents that achieved Grandmaster levels in Chess, Shogi and Go [10]. The next big step in RL was creating AlphaStar, an iteration of AlphaGo and AlphaZero that applied RL to the real-time strategy game

StarCraft II [14]. AlphaStar was a significant step since Starcraft II has a more dynamic environment and unit size/count. A near-infinite number of actions could be taken in StarCraft II, while the number of actions was finite in the previous three games. This implementation of AlphaStar resulted in creating an agent that was ranked as one of the top players in the world within months of initial training [14].

## E. SUMMARY SECTION

The use of AI in wargaming has presented a unique opportunity to take a multitude of scenarios and create a mechanism that can support the decisionmakers of our naval warships and beyond. This robust training process could present multiple courses of actions that allow these commanders to adjust their choices as events unfold. AI is the answer moving forward of how wargaming and planning will be conducted in the Navy. The following chapter will explain the implementation of the testing of AI usage in Naval scenarios.

# III. METHODOLOGY

This chapter discusses the simulation environment used and the creation of the scenarios and agents used to test the questions proposed in this thesis.

## A. SIMULATION ENVIRONMENT

The simulation environment selected for this testing was Atlatl, a program created by Dr. Christian Darken of the Modeling Virtual Environments and Simulation (MOVES) Institute at the Naval Postgraduate School [19]. The program's engine is based in Python, while much of the user interface is written in JavaScript and HTML. JavaScript and HTML allow the human player to see the board and make moves visually. The communication between the program and the AI is in JSON messages. JSON messages provide an easy way for the human to analyze what moves occurred and troubleshoot issues with the program. This environment simulates battles between two forces, seen as the blue team and the red team. The simulation is turn-based and uses a hexagonal board where each unit takes up one space. The program allows human versus human, a human versus an AI, and AI versus AI. Other naval units are in the Naval version of Atlatl but during selection of scenarios units were reduced to two types to reduce variability.

### 1. Terrain in Atlatl

The terrain used in Atlatl was designed to host land units on different types of land terrain (see Figure 6). In Figure 7, the hexes converted to naval-specific types are shown.

Each terrain affects the mobility and defense of the targets on them. Table 1 shows the mobility costs for each terrain type in the original Atlatl configuration while Table 2 shows the costs for the new naval scenarios created for this thesis. The mobility cost for the CDCM was infinite to represent fixed coastal batteries that could not readily move. The mobility cost of the Destroyers was set so they could move two hexes when in

normal deep water but more limited when they entered coastal areas to simulate the restriction of movement that ships deal with when they approach the shore.

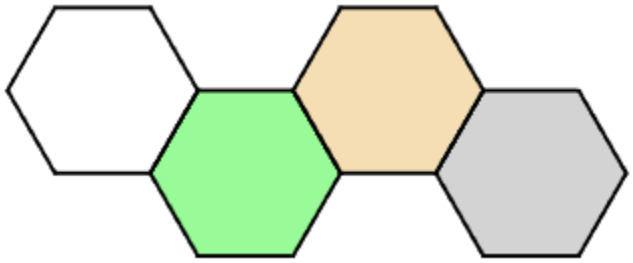

Figure 6. Terrains in Basic Atlatl: Clear, Marsh, Rough, Urban

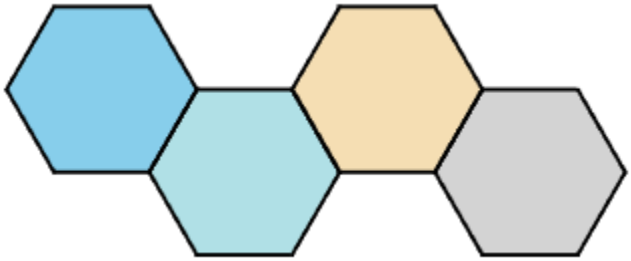

Figure 7. Terrains in Naval Atlatl: Water, Coast, Rough, Urban

Table 1. The Mobility Table for Basic Atlatl. Source [19].

| Mover | Terrain Entered | | | |
|---|---|---|---|---|
| | Clear | Rough | Marsh | Urban |
| Infantry | 100 | 100 | 100 | 100 |
| MechInf | 50 | 100 | 100 | 100 |
| Armor | 50 | 100 | 100 | 100 |
| Artillery | 50 | 100 | NA | 100 |

Table 2. Mobility Table for Naval Atlatl. Adapted from [19].

| | Water | Coast | Rough | Urban (City) |
|---|---|---|---|---|
| Destroyer | 50 | 100 | Infinite | 100 |
| CDCM | Infinite | Infinite | Infinite | Infinite |

The numbers used for the terrain damage multipliers were kept at 1 to standardize the interactions except for in the urban (city) terrain, shown in Table 3. The city terrain was designed to create an incentive to capture the city outside of just the points gained (see Table 4).

Table 3. Targets Damage Multiplier per Target Terrain in the Basic Atlatl. Source [19].

Target's Terrain

| Target | Clear | Rough | Marsh | Urban |
|---|---|---|---|---|
| Infantry | 1 | 0.5 | 1 | 0.5 |
| MechInf | 1 | 1 | 2 | 1 |
| Armor | 1 | 1 | 2 | 1 |
| Artillery | 1 | 1 | 2 | 1 |

Table 4. Damage Multiplier for Naval Atlatl. Adapted from [19].

| | Water | Coast | Rough | Urban (City) |
|---|---|---|---|---|
| Destroyer | 1 | 1 | 1 | .5 |
| CDCM | 1 | 1 | 1 | 1 |

### 2. Atlatl Units

Atlatl Units were initially designed as a land-based simulation using conventional land forces. The units, seen in Figure 8, represent the basic Atlatl units modeled on the U.S. MIL-STD-2525D. The blue and red coloring distinguish which team the unit is on.

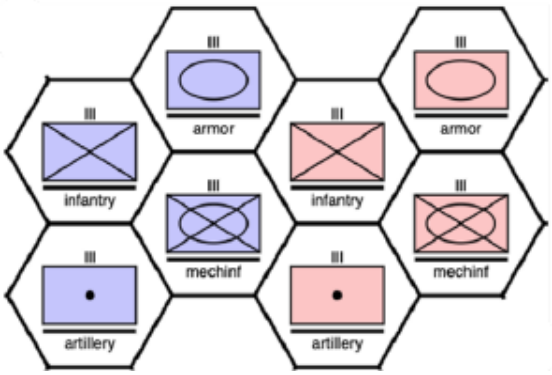


Figure 8. Standard Units Used in Atlatl Source: [20].

This thesis focused on naval assets, so the units had to be exchanged for units more commonly used in naval engagements. The units were given the NATO designations, with the Aircraft Carrier and Amphibious Assault ship being given a symbol resembling their silhouette from the front. In Figure 9, different units could be implemented into the naval simulations, but the destroyer and CDCM were the only two used for the scenarios used in this experiment. The number listed under the icon represents the health of the unit. In Atlatl, the units are considered disabled and unable to fight if they fall below 50% health; this could be modified to a higher or lower percentage, but this thesis stayed with the basic setting [19].

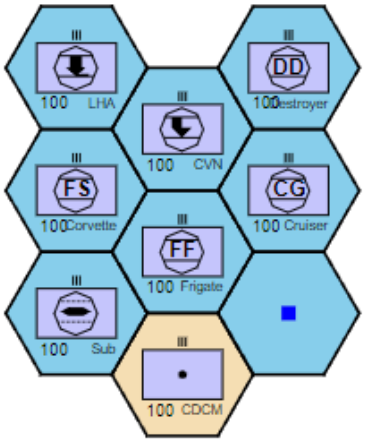


Figure 9. Adapted Naval Assets Sourced from the Atlatl Code Base.

The range that the destroyers could attack was two hexes long, while the CDCM was only given one hex. The CDCM was only used in one scenario (see Figure 16) which tested the AI's ability to recognize the limitations of the CDCM and maneuver around its range to get to the city.

### 3. Scoring in Atlatl

Scoring within Atlatl's framework, as described in Professor Darken's documentation of it, is shaped by combat results, territorial gains, and importantly, player choices, considering that scenario parameters are adjustable during scenario creation [19]. This flexibility permits customization to best reflect the goals of each scenario.

The game score is from the perspective of the Blue player. Scoring predominantly hinges on two facets: combat effectiveness and control of urban areas (or "cities"). In combat, each "strength point" loss inflicted on the Red opponent translates into a positive point for Blue. Blue also incurs a penalty of -1 point for each of its own strength points lost or rendered ineffective in combat. Each unit begins with an initial 100 strength points and is removed from the game once they drop below 50, with the remaining strength points going to the other team.

In addition to combat outcomes, the control of cities plays a significant role in shaping the player's score. At the start of each scenario, cities are not controlled by any faction. Control shifts only when a unit enters the city, with the controlling faction awarded a score of 24 divided by the total number of cities per city under control, awarded each phase. Importantly, once a city is occupied, it remains under the faction's control even if the unit vacates the location, until an opponent's unit occupies it.

This thesis adopted a slight modification to Atlatl's default scoring system: the penalty for Blue's loss of strength points was adjusted from -2 to -1. This alteration was chosen to ensure proportionality in the scoring mechanism.

### 4. Optimal Scoring Baseline

For the human optimal scoring testing, each scenario was played ten times by a researcher who has spent time with Atlatl and had a solid knowledge of how the game

and pass-agg worked. The best score out of those ten plays was selected for each scenario as the human optimal score baseline to test the AI agents against.

## 5. Scenarios

For this thesis, six simulated environments represented six real-life scenarios that naval units could see while conducting operations. Each has unique aspects that test the AI's ability to overcome or use the environments differently to obtain the optimal score. Each scenario will be introduced in this section, and the reason for this selection will be explained.

### *a. Scenario One: Middle Island*

The first scenario (see Figure 10), called Middle Island, was selected to see if the AI could maneuver around both sides of the island to attack the adversary from both sides simultaneously. The coastal waters hinder the destroyers' two-hex movement, which in an all-out assault will stagger the blue units' approach and send them into red range single file. This lack of formation would result in a loss for the blue forces, even with superior numbers. The optimal move found by the human player was to move all the units with outside of the range of one of the enemies and then two units move within range and destroy it the next turn. After that first unit is destroyed the friendly unit that was hurt does not advance with the other two on the final enemy. This move set is seen in Figure 11 with a final score of 100.

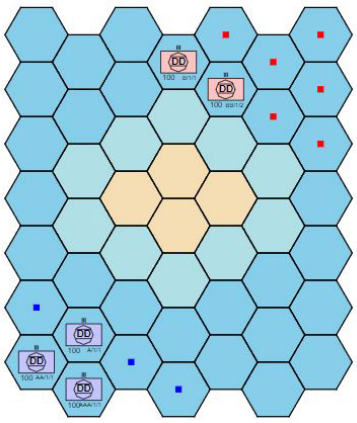

Figure 10. Scenario One Middle Island

**Scenario One Optimal Moves**

| Phase Number | Move Description |
|---|---|
| 0 | A/1/1: hex-1-5→hex-1-4 |
| 0 | AA/1/1: hex-0-6→hex-0-4 |
| 1 | AAA/1/1: hex-1-6→hex-2-5 |
| 2 | B/1/1: passed |
| 2 | BB/1/2: passed |
| 2 | AA/1/1: hex-0-4→hex-0-2 |
| 2 | A/1/1: hex-1-4→hex-0-3 |
| 3 | AAA/1/1: hex-2-5→hex-2-4 |
| 4 | B/1/1: passed |
| 4 | BB/1/2: passed |
| 4 | AA/1/1: hex-0-2→hex-0-0 |
| 4 | A/1/1: hex-0-3→hex-0-1 |
| 5 | AAA/1/1: hex-2-4→hex-1-3 |
| 6 | B/1/1: passed |
| 6 | BB/1/2: passed |
| 6 | AA/1/1: hex-0-0→hex-1-0 |
| 6 | A/1/1: hex-0-1→hex-1-1 |
| 7 | AAA/1/1: passed |
| 7 | B/1/1: passed |
| 8 | BB/1/2: passed |
| 8 | A/1/1: attacked B/1/1 |
| 8 | AA/1/1: attacked B/1/1 |
| 8 | B/1/1: hex-3-0→None |
| 8 | B/1/1: was destroyed |
| 9 | AAA/1/1: hex-1-3→hex-1-2 |
| 10 | BB/1/2: passed |
| 10 | A/1/1: hex-1-1→hex-3-0 |
| 10 | AAA/1/1: hex-1-2→hex-2-2 |
| 11 | AA/1/1: hex-1-0→hex-2-1 |
| 11 | BB/1/2: passed |
| 12 | A/1/1: attacked BB/1/2 |
| 12 | AA/1/1: attacked BB/1/2 |
| 12 | BB/1/2: hex-4-1→None |
| 12 | BB/1/2: was destroyed |

**Final Score**

Final score: 100.0

Figure 11. Optimal Moves for Scenario One

### *b. Scenario Two: Mainland and Island*

The second scenario (see Figure 12) was called Mainland and Island. It represents a close adversarial force that could quickly get to the island and require the blue forces to position themselves to attack while minimizing losses efficiently. The optimal human design here was to leave a unit within range of the city while the other units sailed around the island. This strategy would allow the unit in range of the city to soften up the red unit when it reached the city and allow faster recapture to minimize the loss of points from red controlling the city. This scenario was made as a ten-by-ten map to provide enough

distance between the forces and the city so the red team did not have an insurmountable lead before the blue team could move to their positions. If left as a seven-by-seven map, the reds' score from capturing the city very early was too high so that blue could never generate a positive score. The optimal moves are seen in Figure 13 with a final score of 534.

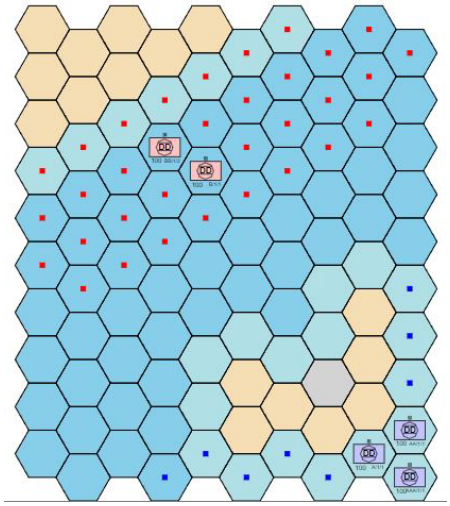

Figure 12. Scenario Two Mainland and Island

**Scenario Two Optimal Moves**

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | AA/1/1: hex-1-4→hex-3-3 | 6 | A/1/1: passed |
| 0 | AAA/1/1: hex-0-5→hex-2-4 | 6 | AAA/1/1: passed |
| 0 | A/1/1: passed | 7 | B/1/1: hex-5-1→hex-3-2 |
| 1 | B/1/1: passed | 7 | BB/1/2: hex-5-2→hex-3-3 |
| 1 | BB/1/2: passed | 8 | A/1/1: attacked B/1/1 |
| 2 | A/1/1: hex-0-4→hex-2-3 | 8 | AAA/1/1: attacked B/1/1 |
| 2 | AA/1/1: hex-3-3→hex-4-3 | 8 | B/1/1: hex-3-2→None |
| 2 | AAA/1/1: passed | 8 | B/1/1: was destroyed |
| 3 | B/1/1: passed | 9 | BB/1/2: passed |
| 3 | BB/1/2: passed | 10 | A/1/1: attacked BB/1/2 |
| 4 | A/1/1: passed | 10 | AAA/1/1: attacked BB/1/2 |
| 4 | AA/1/1: passed | 10 | BB/1/2: hex-3-3→None |
| 4 | AAA/1/1: passed | 10 | BB/1/2: was destroyed |
| 5 | AA/1/1: hex-4-3→None | 12 | AAA/1/1: hex-2-4→hex-2-5 |
| 5 | AA/1/1: was destroyed | 12 | A/1/1: passed |
| 5 | B/1/1: passed | 14 | AAA/1/1: hex-2-5→hex-2-6 |
| 5 | BB/1/2: hex-6-1→hex-5-2 | | |

**Final Score**

Final score: 534.0

Figure 13. Optimal Moves for Scenario Two

### *c. Scenario Three: Multi-Island*

The third scenario (see Figure 14) is called Multi-island, representing small island chains that adversarial forces have occupied. This scenario was selected to simulate the restricted movement of units near islands while facing a defensive enemy. The optimal human path for this scenario is to split the blue forces into three, circumvent the islands, and converge into the middle all at once to hit the red forces from each angle. The optimal moves are seen in Figure 15 with a final score of 100.

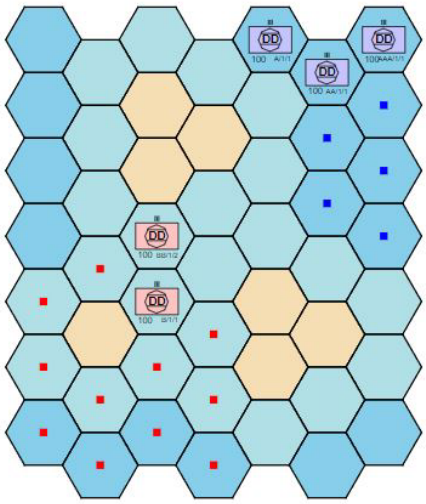

Figure 14. Scenario Three Multi-island

**Scenario Three Optimal Moves**

| Phase Number | Move Description |
|---|---|
| 0 | A/1/1: hex-4-0→hex-3-0 |
| 0 | AA/1/1: hex-5-0→hex-5-1 |
| 0 | AAA/1/1: hex-6-0→hex-6-2 |
| 1 | B/1/1: passed |
| 1 | BB/1/2: passed |
| 2 | A/1/1: hex-3-0→hex-2-0 |
| 2 | AAA/1/1: hex-6-2→hex-6-3 |
| 2 | AA/1/1: passed |
| 3 | B/1/1: passed |
| 3 | BB/1/2: passed |
| 4 | A/1/1: hex-2-0→hex-1-0 |
| 4 | AAA/1/1: hex-6-3→hex-6-4 |
| 4 | AA/1/1: hex-5-1→hex-5-2 |
| 5 | B/1/1: passed |
| 5 | BB/1/2: passed |
| 6 | A/1/1: hex-1-0→hex-0-1 |
| 6 | AAA/1/1: hex-6-4→hex-6-5 |
| 6 | AA/1/1: passed |
| 7 | B/1/1: passed |
| 7 | BB/1/2: passed |
| 8 | AAA/1/1: hex-6-5→hex-5-6 |
| 8 | A/1/1: passed |
| 8 | AA/1/1: passed |
| 9 | B/1/1: passed |
| 9 | BB/1/2: passed |
| 10 | AAA/1/1: hex-5-6→hex-4-6 |
| 10 | AA/1/1: hex-5-2→hex-4-2 |
| 10 | A/1/1: hex-0-1→hex-1-1 |
| 11 | BB/1/2: passed |
| 11 | B/1/1: passed |
| 12 | A/1/1: attacked BB/1/2 |
| 12 | AA/1/1: attacked BB/1/2 |
| 12 | BB/1/2: hex-2-3→None |
| 12 | BB/1/2: was destroyed |
| 12 | AAA/1/1: passed |
| 13 | B/1/1: passed |
| 14 | AAA/1/1: hex-4-6→hex-2-6 |
| 14 | A/1/1: hex-1-1→hex-1-2 |
| 14 | AA/1/1: passed |
| 15 | B/1/1: passed |
| 16 | A/1/1: attacked B/1/1 |
| 16 | AAA/1/1: attacked B/1/1 |
| 16 | B/1/1: hex-2-4→None |
| 16 | B/1/1: was destroyed |

**Final Score**

Final score: 100.0

Figure 15. Optimal Moves for Scenario Three

### *d. Scenario Four: Land Approach*

The fourth scenario (see Figure 16) called Land Approach, was a unique simulation because it utilizes a CDCM-type unit that cannot move but can fire at units one hex away. This scenario also presents a city hex quickly controlled by the red forces.

The optimal human strategy in this scenario is to quickly attack the city while staying out of range of the CDCM. This strategy forces the blue units to attack the destroyers from a distance and move up to the right side of the coast around the CDCM into the city, once the adversary ships are gone. The optimal moves are seen in Figure 17 with a final score of 172.

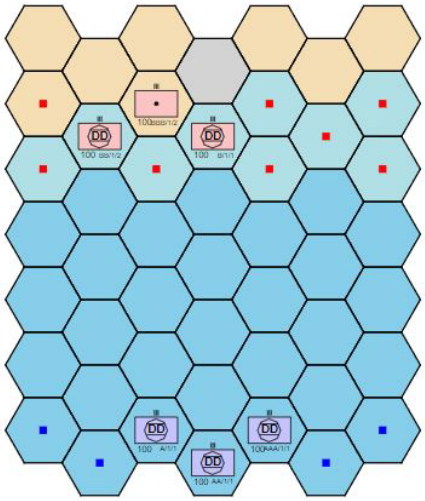

Figure 16. Scenario Four Land Approach

Scenario Four Optimal Moves

| Phase Number | Move Description |
|---|---|
| 0 | A/1/1: hex-2-6→hex-2-4 |
| 0 | AA/1/1: hex-3-6→hex-3-4 |
| 0 | AAA/1/1: hex-4-6→hex-4-4 |
| 1 | B/1/1: hex-3-1→hex-3-0 |
| 1 | BB/1/2: hex-1-1→hex-2-3 |
| 1 | BBB/1/2: passed |
| 2 | A/1/1: attacked BB/1/2 |
| 2 | AA/1/1: attacked BB/1/2 |
| 2 | BB/1/2: hex-2-3→None |
| 2 | BB/1/2: was destroyed |
| 2 | AAA/1/1: hex-4-4→hex-4-3 |
| 3 | B/1/1: passed |
| 3 | BBB/1/2: passed |
| 4 | AA/1/1: hex-3-4→hex-3-2 |
| 4 | A/1/1: hex-2-4→hex-2-3 |
| 4 | AAA/1/1: hex-4-3→hex-4-2 |
| 5 | B/1/1: passed |
| 5 | AAA/1/1: hex-4-2→None |
| 5 | AAA/1/1: was destroyed |
| 5 | BBB/1/2: passed |
| 6 | AA/1/1: attacked BBB/1/2 |
| 6 | BBB/1/2: was destroyed |
| 6 | A/1/1: hex-2-3→hex-2-2 |
| 7 | B/1/1: passed |
| 8 | AA/1/1: attacked B/1/1 |
| 8 | A/1/1: attacked B/1/1 |
| 9 | AA/1/1: hex-3-2→None |
| 9 | AA/1/1: was destroyed |
| 9 | B/1/1: passed |
| 10 | A/1/1: attacked B/1/1 |
| 10 | B/1/1: hex-3-0→None |
| 10 | B/1/1: was destroyed |
| 12 | A/1/1: hex-2-2→hex-3-1 |
| 14 | A/1/1: hex-3-1→hex-3-0 |

Final Score

Final score: 172.0

Figure 17. Optimal Moves for Scenario Four

### *e. Scenario Five: Chokepoint*

The fifth scenario (see Figure 18) is called chokepoint, which was selected to see if the AI could use the restriction of movement through the gap in the land to attack the enemy one by one as they come through. The optimal human movement here is to position the blue forces in a way that allows them to attack each red force as they come through. However, if they are allowed to get far enough get far enough through so that when the number of units on each side even out, the red forces are already through the chokepoint, and the blue team does not need to go through the chokepoint to get the remaining units. Going through the chokepoint after the red units would result in a losing score, as the red team would then be able to employ the same tactic that the blue had of attacking each unit as they went through, seen in Figure 19 with a final score of 250.

.

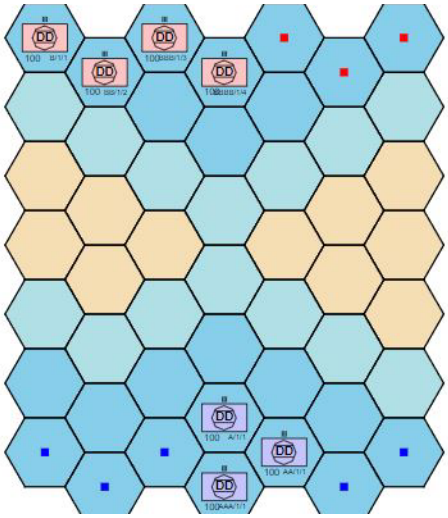

Figure 18. Scenario Five Chokepoint

**Scenario Five Optimal Moves**

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | A/1/1: hex-3-5→hex-1-5 | 9 | BB/1/2: hex-3-3→hex-4-5 |
| 0 | AA/1/1: hex-4-6→hex-5-5 | 9 | B/1/1: hex-3-2→hex-3-3 |
| 0 | AAA/1/1: passed | 9 | BBBB/1/4: hex-4-2→hex-3-2 |
| 1 | B/1/1: hex-0-0→hex-0-1 | 10 | A/1/1: passed |
| 1 | BB/1/2: hex-1-0→hex-1-1 | 10 | AA/1/1: passed |
| 1 | BBB/1/3: hex-2-0→hex-3-1 | 10 | AAA/1/1: passed |
| 1 | BBBB/1/4: hex-3-0→hex-4-1 | 11 | B/1/1: hex-3-3→hex-2-5 |
| 2 | A/1/1: passed | 11 | BB/1/2: passed |
| 2 | AA/1/1: passed | 11 | BBBB/1/4: hex-3-2→hex-3-3 |
| 2 | AAA/1/1: passed | 12 | AA/1/1: attacked BB/1/2 |
| 3 | BB/1/2: hex-1-1→hex-2-2 | 12 | AAA/1/1: attacked BB/1/2 |
| 3 | B/1/1: hex-0-1→hex-1-1 | 12 | BB/1/2: hex-4-5→None |
| 3 | BBB/1/3: hex-3-1→hex-3-2 | 12 | BB/1/2: was destroyed |
| 3 | BBBB/1/4: hex-4-1→hex-4-2 | 12 | A/1/1: attacked B/1/1 |
| 4 | A/1/1: passed | 13 | AAA/1/1: hex-3-6→None |
| 4 | AA/1/1: passed | 13 | AAA/1/1: was destroyed |
| 4 | AAA/1/1: passed | 13 | B/1/1: passed |
| 5 | BBB/1/3: hex-3-2→hex-3-3 | 13 | BBBB/1/4: passed |
| 5 | BB/1/2: hex-2-2→hex-3-2 | 14 | A/1/1: attacked B/1/1 |
| 5 | B/1/1: hex-1-1→hex-2-2 | 14 | B/1/1: hex-2-5→None |
| 5 | BBBB/1/4: passed | 14 | B/1/1: was destroyed |
| 6 | A/1/1: passed | 14 | AA/1/1: hex-5-5→hex-4-6 |
| 6 | AA/1/1: passed | 15 | BBBB/1/4: passed |
| 6 | AAA/1/1: passed | 16 | A/1/1: hex-1-5→hex-2-5 |
| 7 | BBB/1/3: hex-3-3→hex-4-5 | 16 | AA/1/1: hex-4-6→hex-4-5 |
| 7 | BB/1/2: hex-3-2→hex-3-3 | 17 | BBBB/1/4: passed |
| 7 | B/1/1: hex-2-2→hex-3-2 | 18 | AA/1/1: attacked BBBB/1/4 |
| 7 | BBBB/1/4: passed | 18 | A/1/1: attacked BBBB/1/4 |
| 8 | AAA/1/1: attacked BBB/1/3 | 18 | BBBB/1/4: hex-3-3→None |
| 8 | AA/1/1: attacked BBB/1/3 | 18 | BBBB/1/4: was destroyed |
| 8 | BBB/1/3: hex-4-5→None | | |
| 8 | BBB/1/3: was destroyed | | |
| 8 | A/1/1: passed | | |

**Final Score**

Final score: 250.0

Figure 19. Optimal Moves for Scenario Six

### *f. Scenario Six: Tight Channel*

The final scenario (see Figure 20) is called Tight Channel, which tested the AI's ability to efficiently use the space given while not going into the restricted maneuvering hexes of the coast. This scenario also tested the AI's ability to spread their forces into a different formation that would allow all units to attack at once, instead of having the two closest to the red forces waste a turn and potential lives by moving one hex closer to red so that the third unit could engage as well. The optimal human movement for this scenario is to use the three-hex wide channel while staying out of the coastal hexes and

converging into range three-on-one for both red units. The optimal path for Scenario Six is seen in Figure 21 with a final score of 100.

.

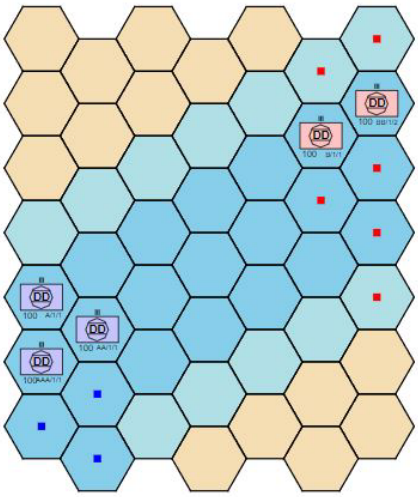

Figure 20. Scenario Six Tight Channel

**Scenario Six Optimal Moves**

| Phase Number | Move Description |
|---|---|
| 0 | A/1/1: hex-0-4→hex-2-3 |
| 0 | AA/1/1: hex-1-4→hex-3-3 |
| 0 | AAA/1/1: hex-0-5→hex-2-5 |
| 1 | B/1/1: passed |
| 1 | BB/1/2: passed |
| 2 | AAA/1/1: hex-2-5→hex-4-4 |
| 2 | A/1/1: hex-2-3→hex-3-2 |
| 2 | AA/1/1: hex-3-3→hex-4-3 |
| 3 | B/1/1: passed |
| 3 | BB/1/2: passed |
| 4 | AAA/1/1: hex-4-4→hex-5-3 |
| 4 | A/1/1: attacked B/1/1 |
| 4 | AA/1/1: attacked B/1/1 |
| 4 | B/1/1: hex-5-1→None |
| 5 | B/1/1: was destroyed |
| 5 | BB/1/2: passed |
| 6 | AA/1/1: hex-4-3→hex-4-2 |
| 6 | AAA/1/1: hex-5-3→hex-6-3 |
| 6 | A/1/1: hex-3-2→hex-4-3 |
| 7 | BB/1/2: passed |
| 8 | AAA/1/1: attacked BB/1/2 |
| 8 | AA/1/1: attacked BB/1/2 |
| 8 | BB/1/2: hex-6-1→None |
| 8 | BB/1/2: was destroyed |

**Final Score**

Final score: 100.0

Figure 21. Optimal Moves for Scenario Six

## 6. Agents

The agents used in this thesis consisted of rules-based, method-based, and value-based AIs. The stock rules-based AI called "pass-agg," created by Professor Darken [19], uses hard-coded decisions based on what the environment is presenting each time it is that AI moves. The core decisions revolve around how many units each team has and their collective strength. If the pass-agg agent has more units and strength, it will move toward the enemy and attack. If it has fewer units and strength, it will adopt a defensive stance and wait for the enemy to attack them. The agent's stance is checked each round, and if the environment changes, the agent will switch to the corresponding stance. This agent will also randomly attack any enemy in range by simply checking if the unit is in within range attack.

The second agent, "mctsnk" (n representing the iteration number), used in this thesis employs uses the method-based Monte Carlo tree search algorithm explained in the

previous chapter to test out all the potential moves to varying levels of iterations. The iteration limits used in this thesis were 30k, 50k. The consideration of the inclusion of 100k was added later to test if the scoring of these agents was affected by the limited depth the agent could go to in these scenarios. 100k iterations drastically increased the time needed to play the 20 games given to each scenario. For this reason, 100k was not tested.

The third agent was a value-based RL agent called "*gym13*," which uses the OpenAI Gym and Stable-Baselines3 to create a training environment that takes a blank slate agent and, through continuous play, learns how to play Atlatl effectively. The scoring function for this agent is derived from the remaining strength left of friendly forces, kills of the adversary, and a terminal bonus value. The terminal bonus value was implemented to encourage the agent to explore without significant loss of points that can result from getting within range of the adversary. OpenAI published the Gym environment framework to give the public a standardized RL method [39]. The Stable-Baselines3 was designed to work with OpenAI Gym, allowing users to implement different learning algorithms [21]. The learning algorithm used in this agent was DQN based on previous work conducted while using Atlatl in the MV4025 course with Professor Darken, which saw an overall higher score than other algorithms such as Proximal Policy Optimization (PPO) [22][23].

The final agent was the implementation of the AlphaZero RL algorithm inside Atlatl. This agent will be based on "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm" by Silver et al. [10], "Learning to Play Othello without Human Knowledge" by Thakoor et al., and GitHub [8]. These sources describe the foundations of the AlphaZero framework. The Thakoor et al. paper explains how AlphaZero uses a neural network and self-play method to create an agent that learns how to play "Othello." This AlphaZero framework, provided by the researchers in the paper as open source, enabled the creation of the Atlatl AlphaZero agent. Atlatl AlphaZero uses the MCTS approach in the self-play method to find the best estimated or terminal move. The resulting reward from the action picked in the MCTS then passes

back to the neural network along with the game winner to refine the policy that minimizes loss.

## B. SUMMARY

The ability to get the original Atlatl simulation to function in a naval environment represents the first step of this thesis to ensure that the testing of the agents and the scenarios could work. The agents were designed so that the earlier ones were used as cornerstones of the latter agents. The scenarios were designed to represent real-world situations the U.S. Navy could face. This thesis focuses on putting different types of AI agents in six standardized scenarios and demonstrating if the agents could identify the best path to achieving the maximum score. The next chapter will document the agents' results in the different scenarios.

# IV. RESULTS

The chapter shows the experiments' results and reviews the different models' performance in the six scenarios created. The chapter will then analyze if the models can be utilized in wargaming.

## A. PASS-AGG AGENT

The first agent, "pass-agg," was the simple rules-based agent that the rest of the agents were to be compared against. Once Atlatl was reconfigured for naval combat, pass-agg was tested against a passive stock agent that did not move. These scores were not included as it was designed to ensure pass-agg could function correctly in the naval variant. The following results, seen in Figure 22 and Table 5, were from simulations in each of the six scenarios where pass-agg played against itself.

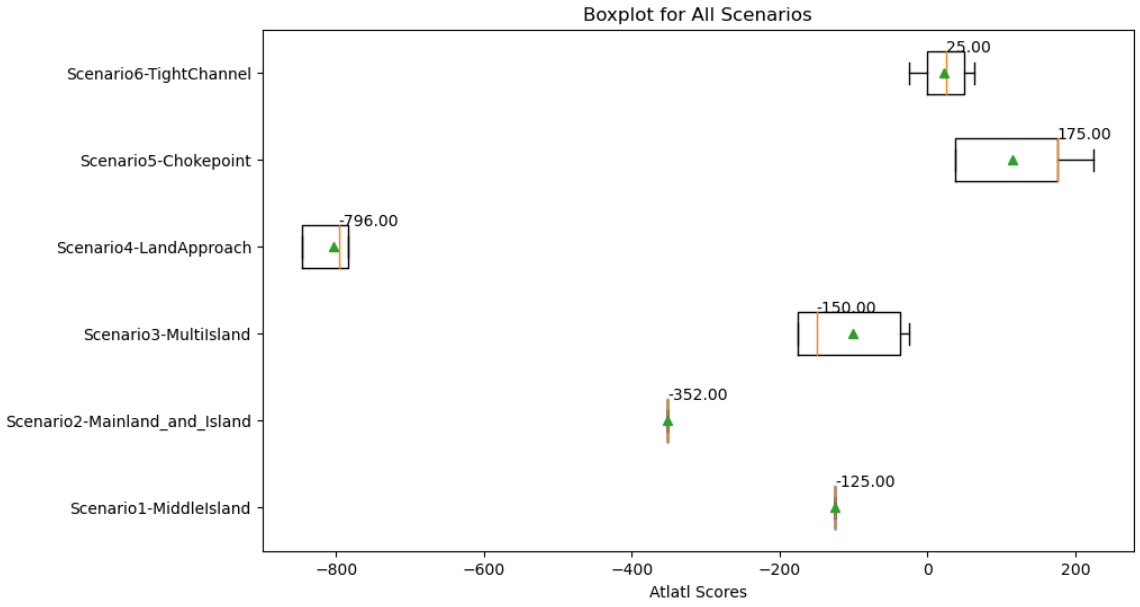


Figure 22. Boxplot of pass-agg vs. pass-agg in Each Scenario, N = 500

Table 5. Mean Scores for pass-agg vs. pass-agg on Each Scenario, N=500

| pass-agg vs. pass-agg | Mean Score | Standard Deviation |
|---|---|---|
| Scenario1-MiddleIsland | -125 | 0 |
| Scenario2-Mainland_and_Island | -352 | 0 |
| Scenario3-MultiIsland | -101.375 | 69.18 |
| Scenario4-LandApproach | -802.875 | 25.94 |
| Scenario5-Chokepoint | 115.5 | 79.2 |
| Scenario6-TightChannel | 22.5 | 25.09 |

### 1. Scenario One Results

Scenario one was consistently poor due to pass-agg's limited ability to understand the environment outside of where it can go and its fighting posture based on both sides' unit strength. The blue force pass-agg had more units than the red force, which resulted in blue sending all its units straight toward red. The middle island acted as a chokepoint with the board perimeter, and blue sent its units in one at a time. The actions of the blue force resulted in units lost and weakened, which changed both teams' stances, and red, now having more unit strength, was able to attack and destroy blues units.

### 2. Scenario Two Results

Scenario two's scores reflected the agent's inability to get around the island and attack the red forces before they captured the city. This inability resulted in the red team being able to collect 24 points per turn which the blue forces could never recover from. This reinforces that the pass-agg agent cannot make complex decisions and react in a scripted sense.

### 3. Scenario Three Results

In Scenario Three, pass-agg had more units but still suffered from sending its units in one at a time, generating a low score. The blue forces under pass-agg's control

would lose the strength advantage and then stop advancing towards red. Having gained the strength advantage, red would move to attack blue, so the standard deviation is larger than in the previous scenarios. Pass-agg's rule for attacking is that it selects a random unit in range, which contributes to the score's variability. If the units randomly concentrated their fire on one unit, that would result in a different score than if the attacks were spread across multiple units.

### 4. Scenario Four Results

Scenario four is the most unfavorable scenario for the blue forces. The red units start next to the city, easily allowing red to score 24 points in each phase. The blue forces need to move quickly up the right side of the board to minimize damage and take over the city. This is beyond pass-agg's simple algorithm, which reflects the large negative score in this scenario. The slight standard deviation results from the same random attack design explained in scenario three.

### 5. Scenario Five Results

Scenario Five is the most favorable for the pass-agg blue units, which have fewer units than the red forces, three to four. This results in the red forces pressing the attack and funneling into the chokepoint in the middle of the map. That chokepoint, along with the restricted movement of the coastal water, allows the blue units to destroy the red units as they advance. The deviation is that large because red stops advancing once they lose their strength advantage. Having gained the strength advantage, the blue forces make the same mistake red did by moving into the chokepoint one at a time. This, combined with pass-agg's random attack selection, provides less certainty on each round but still results in a relatively high mean for blue forces.

### 6. Scenario Six Results

In Scenario Six, the blue forces have the strength advantage but do not use their units appropriately. The simple advance and fire algorithm of pass-agg creates a situation where blue enters a two-on-two with the red units when it could push the third unit into the clear area in the channel and have all the units advance together. This self-limiting

situation evens the planning field, reflected in the score with a minor positive average with a standard deviation of about one unit being destroyed on blue or not.

## B. MONTE CARLO TREE SEARCH AGENT

The Monte Carlo tree search (MCTS) Agent "mctsnk" was run with various simulations per game. Based on the implementation of MCTS, the agent selects the best-valued node it could find in its random search. The number of simulations used in this thesis was 30k and 50k. This was designed to analyze the utility of simulations versus performance. These scores (see Figure 23 and Table 6) were then compared to the best score achieved by human play against pass-agg.

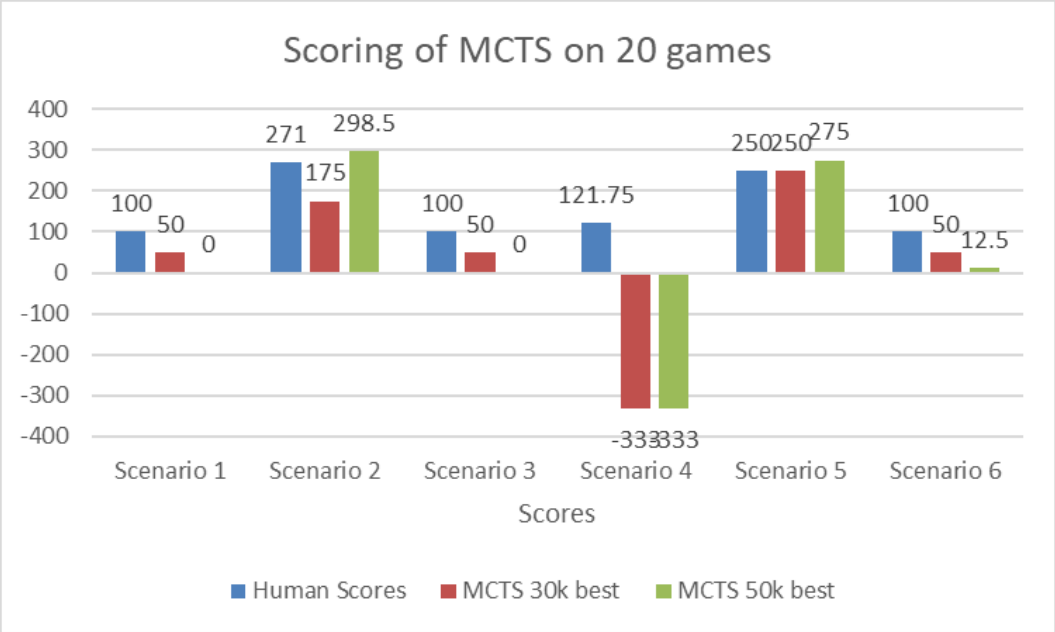


Figure 23. Best MCTS Simulations against pass-agg

Table 6. Scores of the MCTS Agent vs. pass-agg. N= 30k and 50k Iterations in 20 games

| 30k Iterations | Mean | Std | Max |
|---|---|---|---|
| Scenario 1 | -16.25 | 36.52 | 50 |
| Scenario 2 | -11.58 | 146.67 | 175 |
| Scenario 3 | -6.25 | 26.75 | 50 |
| Scenario 4 | -464.50 | 90.59 | -333 |
| Scenario 5 | 168.13 | 96.44 | 250 |
| Scenario 6 | -27.50 | 47.92 | 50 |
| 50k Iterations | Mean | Std | Max |
| Scenario 1 | -3.75 | 16.77 | 0 |
| Scenario 2 | -354.65 | 184.34 | 298.5 |
| Scenario 3 | -11.25 | 27.48 | 0 |
| Scenario 4 | -450.13 | 68.61 | -333 |
| Scenario 5 | 214.06 | 51.13 | 275 |
| Scenario 6 | -29.38 | 38.32 | 12.5 |

### 1. MCTS Agent Performance

The MCTS agent, while not yet at human-level performance in all scenarios, showed encouraging results in the scenarios carried out. The solutions it produced were generally a clear improvement over the pass-agg baseline, as shown in the figure above. However, the results varied quite significantly across the six scenarios.

#### *a. Scenario One and Three Results*

In Scenario One and Three, the MCTS agent was able to score better on the 30k iterations than the 50k iterations. This could be explained by the MCTS agent focusing too much on exploring new moves rather than exploiting known beneficial moves, leading to suboptimal actions being selected. Increasing the number of simulations may dilute the impact of good moves, reducing the agent's ability to effectively use its accumulated knowledge from earlier parts of the tree.

### *b. Scenario Two Results*

In contrast, the MCTS agent outperformed the human benchmark in Scenario Two with 50k simulations. This scenario highlighted the positives of more moves as they can discover better, more rare paths that result in higher scores. This is a good demonstration of the MCTS algorithm's ability to effectively explore and exploit the decision space. The additional simulations allowed the agent to build a more robust decision tree, leading to the achievement of a higher score.

### *c. Scenario Four Results*

Despite the seemingly daunting situation presented in Scenario Four, the MCTS agent managed to limit its losses. The agent's negative scores suggest the need for further optimization to effectively counter early advantageous positions of the enemy. The decision space exploration of the MCTS algorithm will have to be improved to find optimal strategies in such difficult situations.

### *d. Scenario Five Results*

In Scenario Five, the MCTS agent tied with the human score using 30k simulations and slightly improved with 50k simulations. This scenario's success is due to the agent's ability to manage limited resources and take advantage of the map's chokepoint, which restricted the enemy's movements.

### *e. Scenario Six Results*

Scenario Six was another scenario where an increased number of simulations resulted in a lower score. The MCTS agent's performance decreased from 50 to 12.5 when the simulations increased from 30k to 50k. This counter-intuitive result might be attributed to overfitting. As the number of simulations increases, the MCTS algorithm might have explored fewer probable scenarios, thus diluting the effectiveness of the decision-making process. This might have led the MCTS agent to select sub-optimal moves that were effective in rare cases but not in the average scenario.

Overall, these results provide promising insights into the utility of MCTS algorithms for wargaming decision-making and offer invaluable information for further optimization. It is crucial to understand that while increasing the number of simulations often leads to improved performance, as demonstrated in Scenario Two, there can be a point of diminishing returns or even negative impact, as shown in Scenario One, Three, and Six. Future work should focus on refining the balance between exploration and exploitation and the number of simulations, considering the specific context of each scenario.

## C. DEEP Q-NETWORK AGENT

The Deep Q-Network (DQN) agent was trained throughout 5 million timesteps, i.e., games, played against the pass-agg agent in each scenario. The reward function was modified, seen in section 4.C.2, to give more precise feedback to the DQN agent, reinforcing beneficial and discouraging detrimental strategies and enabling the agent to better understand the link between its actions and their outcomes. Similarly, the action space was adjusted, seen in section 4.C.3 to clarify the range of decisions available to the AI in each scenario, helping it to make more informed and effective choices. The results of the scores are seen in Figure 24 and Table 7.

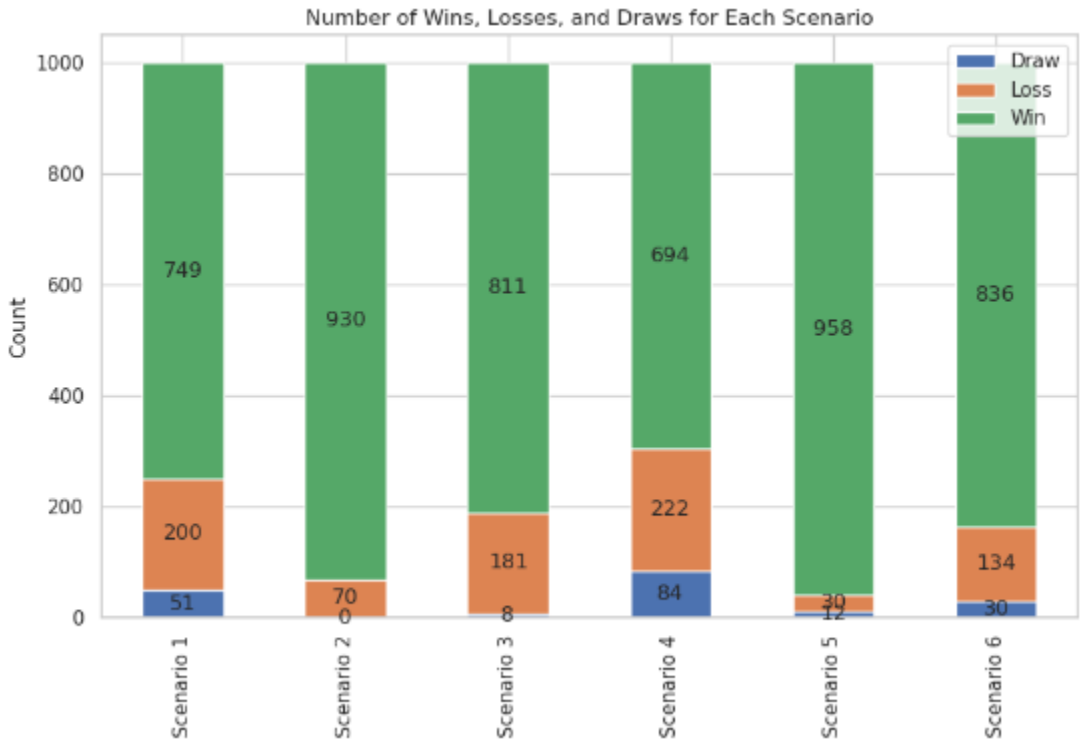


Figure 24. Wins/Losses/Ties of the DQN Agent vs. pass-agg. N=50

Table 7. Scores of the DQN Agent vs. pass-agg. N=1000

| DQN vs. Pass-agg | Games | Mean | Std | Max |
|---|---|---|---|---|
| Scenario 1 | 1000 | 12.23 | 78.19 | 50 |
| Scenario 2 | 1000 | 132.74 | 138.09 | 175 |
| Scenario 3 | 1000 | 25.04 | 57.85 | 50 |
| Scenario 4 | 1000 | 72.32 | 279.22 | 559 |
| Scenario 5 | 1000 | 231.14 | 82.26 | 300 |
| Scenario 6 | 1000 | 23.653 | 66.18 | 50 |

### 1. Deep Q-Network Agent Performance

The AI's strong performance across the varying game scenarios indicates that it has effectively learned and demonstrated its ability to conquer the complexity of the wargaming environment.

#### *a. Scenario One DQN Results*

In Scenario One, the DQN AI recorded an average score of 12.23, maxing out at 50. In comparison to the MCTS agent, it achieved the same highest score as MCTS but MCTS does not have an ability in it to keep its trees from game to game. This means that DQN, which had a saved model, could identify new paths, and use that knowledge in future games resulting in better scores.

#### *b. Scenario Two DQN Results*

In Scenario Two, the DQN AI agent achieved a mean score of 132.74, with a max of 175. In comparison, the MCTS Agent scored a max score of 298.5. This is explained by the DQN agent never reaching the city hex during its exploration. An increase in training timesteps or longer exploration time could result in the DQN agent finding the city and greatly increasing its score.

#### *c. Scenario Three DQN Results*

Scenario three again showcases DQN AI's robust performance with an average score of 25.04, reaching a max of 50. These results are explained in a similar way to Scenario One.

#### *d. Scenario Four DQN Results*

The fourth scenario posed a formidable challenge for all agents, but the DQN AI prevailed, securing an average score of 72.32 and a maximum of 559. Comparatively, the MCTS agent was unable to find the path DQN was able to use for the major victories in this scenario. The DQN agent was able to engage the red units before they took the city resulting in the red team's loss of points generated per turn while fighting blue. The

human optimal moves were seen as less optimal as well because it allowed red to capture the city before blue destroyed them and took it over.

#### *e. Scenario Five DQN Results*

In Scenario Five, the DQN AI achieved an average score of 231.14, with a max of 300, outperforming the MCTS agent and human scores. The DQN agent was able to pull more of the red units into the chokepoint before launching their attack resulting in a larger score.

#### *f. Scenario Six DQN Results*

In Scenario Six, the DQN AI upheld a strong average score of 23.65, peaking at 50, while the MCTS agent performance took a hit when increasing simulations from 30k to 50k.

### 2. Mean Rewards in Training

In the following Figures 25,26,27,28,29, and 30, the average mean reward per DQN evaluation is shown for each training. The dark lines on these graphs represent the actual performance measurements taken during training, while the lighter lines are the smoothed version of the same data, helping to visualize the overall trends by reducing noise from the raw data. Across all scenarios, the agent underwent a preliminary period of exploratory gameplay, spanning 250,000 timesteps, i.e., games versus pass-agg, before commencing its learning process. This exploratory phase was implemented to prompt the agent to recognize critical in-game interactions and areas of significance, subsequently facilitating its learning. The success of this approach is manifest in the sharp increase in scores observed between the 1.5 to 3.5 million timestep marks in the ensuing figures. This evident increase in performance underscores the agent's capacity to learn and adapt effectively within the dynamic parameters of the scenarios.

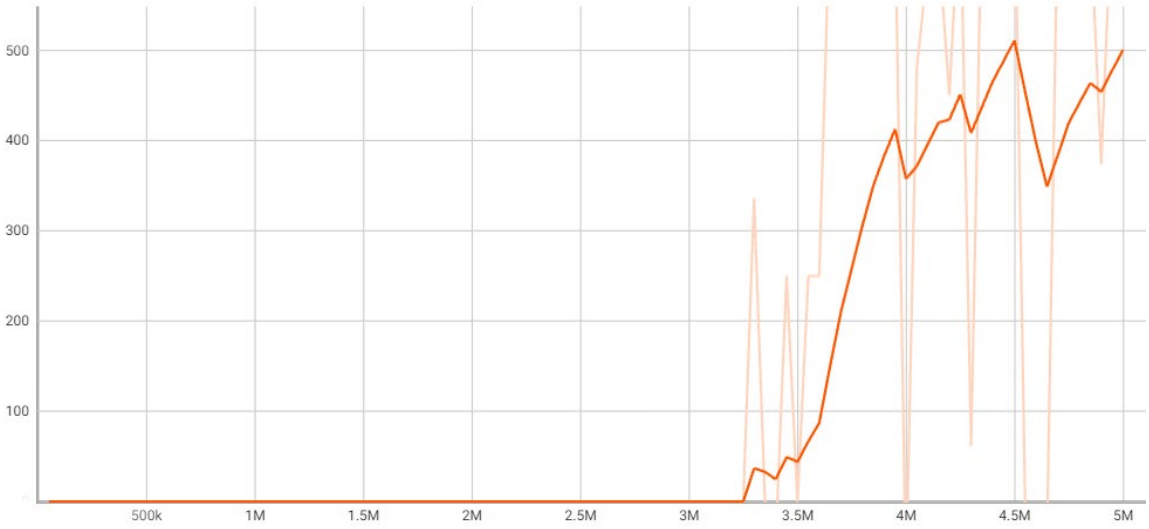


Figure 25. Mean Reward per Evaluation during Scenario One Training

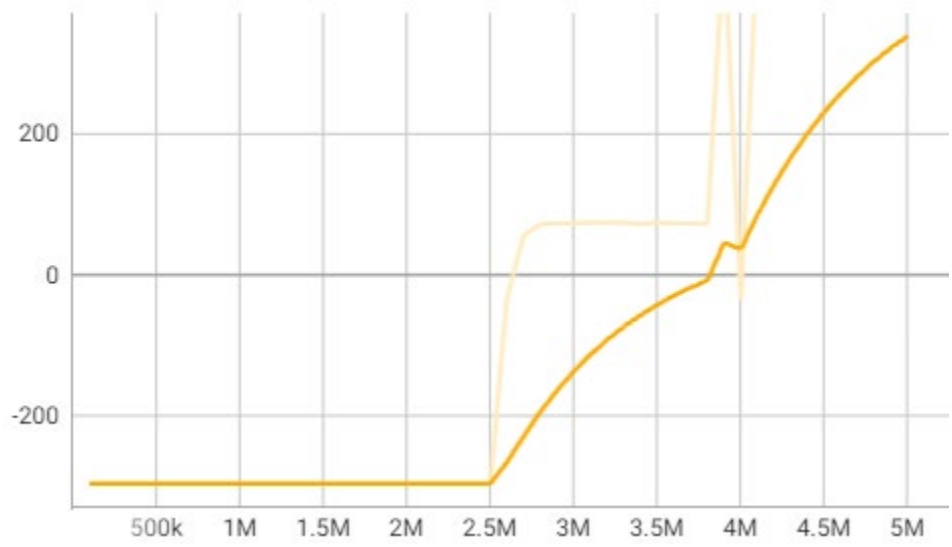


Figure 26. Mean Reward per Evaluation during DQN Scenario Two Training

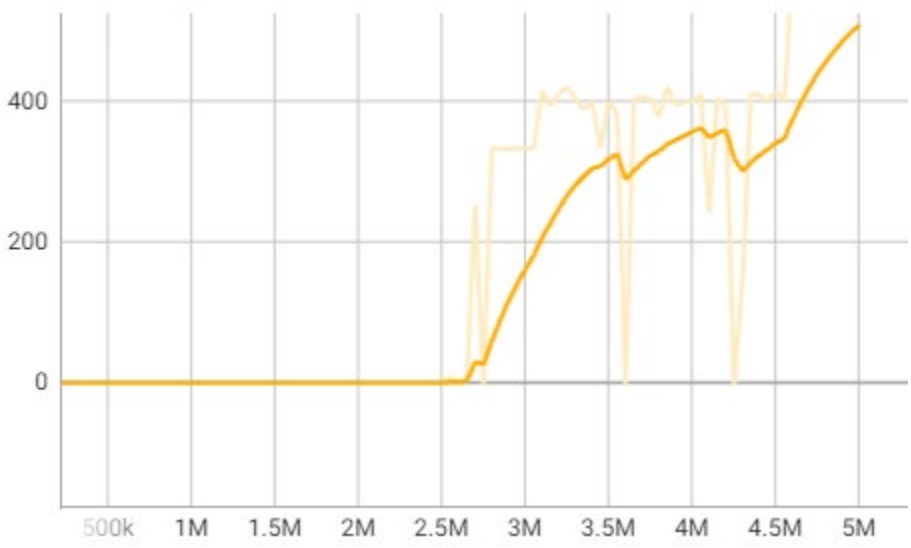


Figure 27. Mean Reward per Evaluation during DQN Scenario Three Training

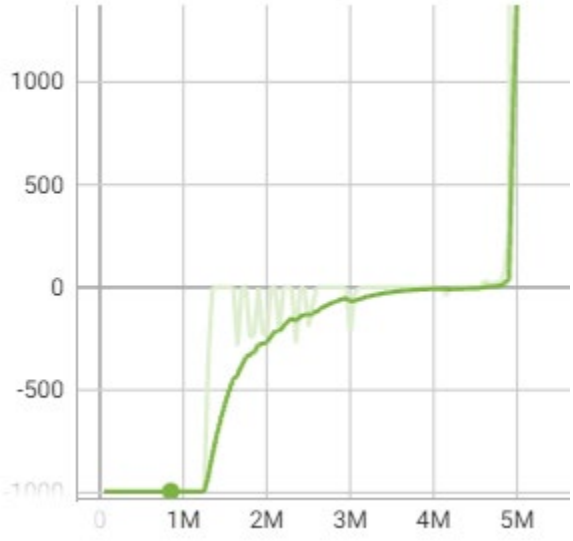


Figure 28. Mean Reward per Evaluation during DQN Scenario Four Training

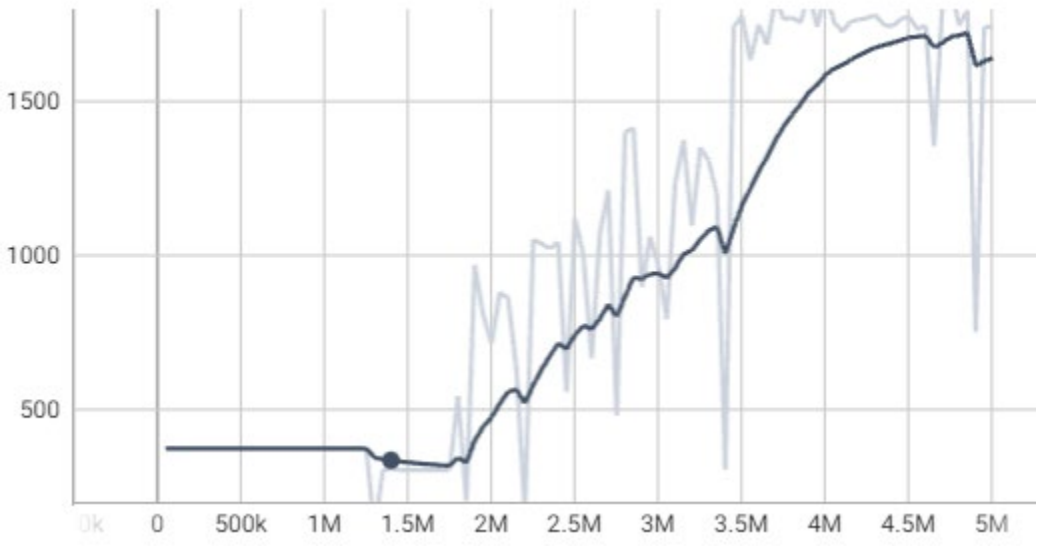


Figure 29. Mean Reward per Evaluation during DQN Scenario Five Training

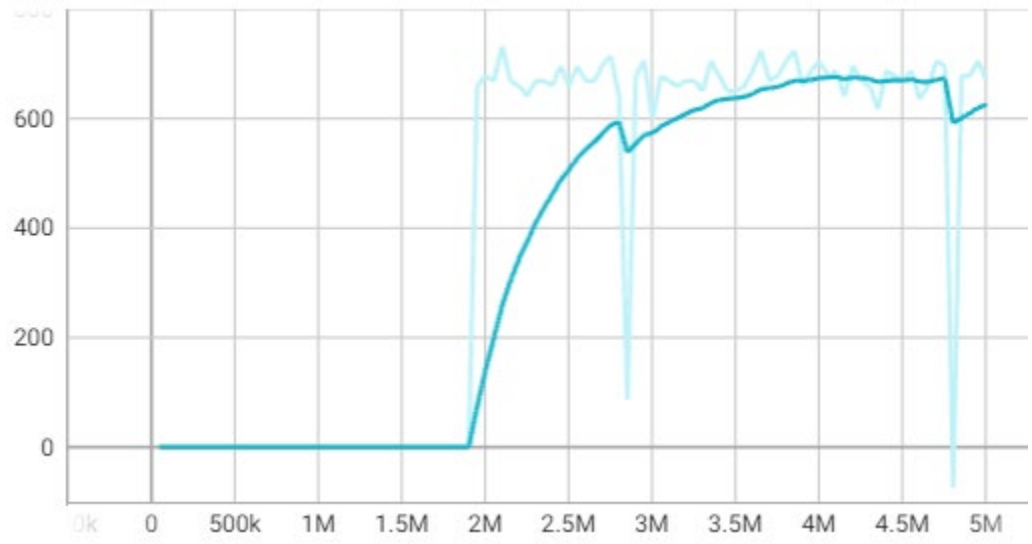


Figure 30. Mean Reward per Evaluation during DQN Scenario Six Training

### 3. Reward Function

The pseudocode for the reward function, seen in Figure 31, is built from the *BoronRewArt* function created by Jonathan Boron [24]. In the function, aside from the conservation of friendly units' reward code that Jonathan Boron created, the agent gained constant rewards throughout the simulation as the enemy strength decreased and if the city, if

applicable, was captured by friendly units. A penalty was also inflicted on the agent's score if the enemy captured the city. This penalty and a penalty inflicted on the total game score created a significant drop in the reward score. That is why the agent penalty for losing a city in the reward function is only ten percent of the positive bonus of capturing the city. Having immediate rewards following the capture of a city or damage to the enemy enhanced the agent's ability to more accurately identify which specific actions were most valuable in each scenario.

**Algorithm 1** Engineered Reward Calculation

```
1: cityOwner: dictionary mapping cities to their current owners
2: unitData: data containing unit information
3: raw_reward: the initial reward obtained by the AI agent
4: is_terminal: boolean indicating whether the game state is terminal or not
5: function EngineeredReward(cityOwner, unitData, raw_reward, is_terminal)
6:     if self.original_strength is None then
7:         self.original_strength ← totalStrength(self.own_faction, unitData)
8:     end if
9:     if self.original_enemy_strength is None then
10:        self.original_enemy_strength ← totalStrength(self.enemy_faction, unitData)
11:    end if
12:    current_strength ← totalStrength(self.own_faction, unitData)
13:    current_enemy_strength ← totalStrength(self.enemy_faction, unitData)
14:    reward_score ← 0
15:    For each city owned by enemy faction, subtract 5 from reward_score
16:    For each city owned by own faction, add 50 to reward_score
17:    raw_reward ← raw_reward + reward_score
18:    if current_enemy_strength is not zero then
19:        strength_bonus ← original_enemy_strength / current_enemy_strength
20:    else
21:        strength_bonus ← original_enemy_strength * 0.1
22:    end if
23:    raw_reward ← raw_reward * strength_bonus
24:    adjusted_reward ← raw_reward * (current_strength / original_strength)
25:    return adjusted_reward
26: end function
```

Figure 31. Pseudocode of Reward Function. Adapted from [24].

### 4. Action Space

In the original Atlatl game, most units could only shoot or move to the hex spaces around it. In the naval variant, the destroyers can shoot or move, in open water, up to two hex spaces. This tripled the action space to 18 from 6. This increase in action space

itself , from the standard version of Atlatl, did not lead to any specific increase in score in any scenario, seen in Figure 32 of Scenario One's DQN training reward average. The other part of the defined action space was that the action to move or shoot at one of the 18 hexes was covered by one action. In this naval variant, the action space was doubled to 36 +1 for the pass action. The second modification, seen in Figure 33, was splitting up the move feature and creating a shoot feature, *legalFireTargets*, providing the neural network with a feature focused on shooting to increase its understanding of the environment.

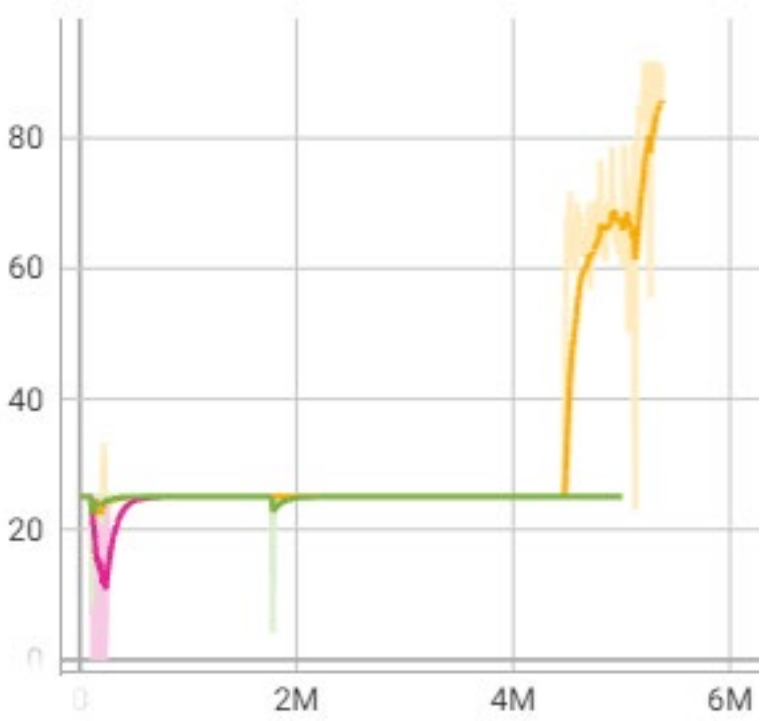


Figure 32. Mean Reward per Evaluation during Scenario One Training with an Action Space of 19

**Algorithm 2** Class AI13

```
Class AI13(AI):
    Function init(self, role, kwargs)
        AI.init(self, role, kwargs)
    Function observation(self)
        Define the feature vectors using mover, canMoveFeature,
        legalMoveFeatureFactory, legalFireTargetFeatureFactory,
        blueUnitFeature, redUnitFeature, unitTypeFeatureFactory,
        terrainFeatureFactory, constantFeatureFactory
    Function legalMoveHexes(self, mover)
        Create a dictionary to map each legal move target for mover to True
    Function legalFireTargets(self, mover)
        Create a dictionary to map each legal fire target for mover to True
    Function getNFeatures(self)
        return 13
```

Figure 33. Pseudocode of AI13 Class

## 5. DQN Best Move Sets

The best scoring move sets for the DQN agents in each scenario are shown in Figures 34,35,36,37,38,and 39 below. Although the DQN agents did not achieve the optimal human results in all the scenarios created, they did manage to find a better solution to Scenarios Four and Five. In the analysis of Scenario Four's replay, the DQN agent pushed directly toward the enemy units and immediately engaged them in the fire. This early engagement prevented the enemy from moving into the city, and after the DQN agent finished off the last enemy, it could move into the city and collect its points until the end of the game. In Scenario Five, the DQN agent spread out its units as the human move set did, but what caused the end score to be higher than the human score was the DQN agents allowing one of its units to take a single hit early in the simulation, lowering the collective strength of its units. After taking the hit, the unit backed away as the rest of the enemy units came through the chokepoint. This tactic helped keep the enemy units moving through the channel due to blue having a slightly lower overall strength vs. red which kept red on the offensive. In the human play, the last enemy unit would always stop just short of the channel due to their posture shift. This stopping by the enemy unit made the human player move their units at the enemy, which resulted in a 50-point decrease as the enemy was allowed to attack one more time. In Scenarios One, Three, and Six, the DQN agent found success in the one unit attacking the enemy, which

would, in turn, cause the enemy to attack the other remaining DQN agent units. The sacrificing of one unit was 50 points less effective than the human moves, which would attack with two, and once one of the units was hurt, it fell back and let the other two units attack with full strength. In Scenario Two, the agent adopted a defensive posture on the island's backside and destroyed the enemy forces as they entered the hexes outside the city. This defensive play led to a high score in the game, and the only reason it is much lower than the human score is that it never seemed to move into the city hex. This limited exploration or poor luck resulted in the agent never learning about how the city is essential. If the agent had more time to explore, it may have found the city, which would raise its score to around the human score level.

### Scenario One DQN Moves

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | A/1/1: hex-1-5→hex-0-4 | 11 | A/1/1: hex-1-1→None |
| 0 | AA/1/1: hex-0-6→hex-0-5 | 11 | A/1/1: was destroyed |
| 0 | AAA/1/1: hex-1-6→hex-1-4 | 11 | B/1/1: passed |
| 1 | B/1/1: passed | 11 | BB/1/2: hex-4-1→hex-3-1 |
| 1 | BB/1/2: passed | 12 | AA/1/1: passed |
| 2 | A/1/1: hex-0-4→hex-0-3 | 12 | AAA/1/1: passed |
| 2 | AA/1/1: passed | 13 | B/1/1: hex-3-0→hex-1-1 |
| 2 | AAA/1/1: passed | 13 | BB/1/2: hex-3-1→hex-2-2 |
| 3 | B/1/1: passed | 14 | AA/1/1: passed |
| 3 | BB/1/2: passed | 14 | AAA/1/1: passed |
| 4 | A/1/1: passed | 15 | B/1/1: hex-1-1→hex-0-3 |
| 4 | AA/1/1: passed | 15 | BB/1/2: hex-2-2→hex-1-2 |
| 4 | AAA/1/1: passed | 16 | AA/1/1: attacked B/1/1 |
| 5 | B/1/1: passed | 16 | AAA/1/1: attacked B/1/1 |
| 5 | BB/1/2: passed | 16 | B/1/1: hex-0-3→None |
| 6 | AA/1/1: hex-0-5→hex-0-4 | 16 | B/1/1: was destroyed |
| 6 | A/1/1: passed | 17 | BB/1/2: passed |
| 6 | AAA/1/1: passed | 18 | AA/1/1: attacked BB/1/2 |
| 7 | B/1/1: passed | 18 | AAA/1/1: attacked BB/1/2 |
| 7 | BB/1/2: passed | 18 | BB/1/2: hex-1-2→None |
| 8 | A/1/1: hex-0-3→hex-1-1 | 18 | BB/1/2: was destroyed |
| 8 | AA/1/1: passed | | |
| 8 | AAA/1/1: passed | | |
| 9 | B/1/1: passed | | |
| 9 | BB/1/2: passed | | |
| 10 | A/1/1: passed | | |
| 10 | AA/1/1: passed | | |
| 10 | AAA/1/1: passed | | |

### Final Score

Final score: 50.0

Figure 34. Best DQN Scoring Move Set for Scenario One

## Scenario Two DQN Moves

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | A/1/1: hex-8-9→hex-7-9 | 12 | AA/1/1: hex-9-5→hex-9-4 |
| 0 | AA/1/1: hex-9-8→hex-9-7 | 12 | AAA/1/1: hex-9-6→hex-9-5 |
| 0 | AAA/1/1: passed | 14 | AA/1/1: hex-9-4→hex-9-3 |
| 1 | B/1/1: hex-4-3→hex-5-4 | 14 | AAA/1/1: hex-9-5→hex-9-4 |
| 1 | BB/1/2: hex-3-2→hex-5-3 | 14 | A/1/1: passed |
| 2 | A/1/1: hex-7-9→hex-6-9 | 16 | A/1/1: hex-6-9→hex-5-9 |
| 2 | AAA/1/1: hex-9-9→hex-9-8 | 16 | AA/1/1: hex-9-3→hex-8-4 |
| 2 | AA/1/1: passed | 16 | AAA/1/1: hex-9-4→hex-8-5 |
| 3 | B/1/1: hex-5-4→hex-6-6 | 18 | AA/1/1: hex-8-4→hex-7-4 |
| 3 | BB/1/2: hex-5-3→hex-6-5 | 18 | AAA/1/1: hex-8-5→hex-7-5 |
| 4 | AA/1/1: hex-9-7→hex-9-6 | 18 | A/1/1: passed |
| 4 | A/1/1: passed | 20 | AA/1/1: hex-7-4→hex-6-5 |
| 4 | AAA/1/1: passed | 20 | A/1/1: passed |
| 5 | B/1/1: hex-6-6→hex-6-7 | 20 | AAA/1/1: passed |
| 5 | BB/1/2: hex-6-5→hex-6-6 | 22 | A/1/1: passed |
| 6 | A/1/1: attacked B/1/1 | 22 | AA/1/1: passed |
| 6 | AAA/1/1: hex-9-8→hex-9-7 | 22 | AAA/1/1: passed |
| 6 | AA/1/1: passed | 24 | AAA/1/1: hex-7-5→hex-7-4 |
| 7 | B/1/1: passed | 24 | A/1/1: passed |
| 7 | BB/1/2: hex-6-6→hex-7-6 | 24 | AA/1/1: passed |
| 8 | A/1/1: attacked B/1/1 | 26 | A/1/1: hex-5-9→hex-6-9 |
| 8 | B/1/1: hex-6-7→None | 26 | AA/1/1: hex-6-5→hex-6-4 |
| 8 | B/1/1: was destroyed | 26 | AAA/1/1: hex-7-4→hex-7-3 |
| 8 | AA/1/1: attacked BB/1/2 | 28 | AA/1/1: hex-6-4→hex-5-4 |
| 8 | AAA/1/1: attacked BB/1/2 | 28 | AAA/1/1: hex-7-3→hex-6-4 |
| 8 | BB/1/2: hex-7-6→None | 28 | A/1/1: passed |
| 9 | BB/1/2: was destroyed | | |
| 10 | AA/1/1: hex-9-6→hex-9-5 | | |
| 10 | AAA/1/1: hex-9-7→hex-9-6 | | |
| 10 | A/1/1: passed | | |

## Final Score

Final score: 175.0

Figure 35. Best DQN Scoring Move Set for Scenario Two

## Scenario Three DQN Moves

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | A/1/1: hex-4-0→hex-4-1 | 8 | A/1/1: attacked BB/1/2 |
| 0 | AA/1/1: hex-5-0→hex-5-2 | 8 | BB/1/2: hex-3-2→None |
| 0 | AAA/1/1: passed | 8 | BB/1/2: was destroyed |
| 1 | B/1/1: passed | 8 | AAA/1/1: attacked B/1/1 |
| 1 | BB/1/2: passed | 8 | B/1/1: hex-3-3→None |
| 2 | AA/1/1: hex-5-2→hex-4-3 | 8 | B/1/1: was destroyed |
| 2 | AAA/1/1: hex-6-0→hex-5-1 | | |
| 2 | A/1/1: passed | | |
| 3 | B/1/1: passed | | |
| 3 | AA/1/1: hex-4-3→None | | |
| 4 | AA/1/1: was destroyed | | |
| 3 | BB/1/2: passed | | |
| 4 | AAA/1/1: hex-5-1→hex-5-2 | | |
| 4 | A/1/1: passed | | |
| 5 | B/1/1: hex-2-4→hex-3-3 | | |
| 5 | BB/1/2: hex-2-3→hex-3-2 | | |
| 6 | A/1/1: attacked BB/1/2 | | |
| 6 | AAA/1/1: attacked B/1/1 | | |
| 7 | B/1/1: passed | | |
| 7 | BB/1/2: passed | | |

## Final Score

Final score: 50.0

Figure 36. Best DQN Scoring Move Set for Scenario Three

## Scenario Four DQN Moves

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | A/1/1: hex-2-6→hex-1-4 | 8 | A/1/1: hex-3-3→hex-3-2 |
| 0 | AA/1/1: hex-3-6→hex-3-4 | 8 | AA/1/1: hex-3-4→hex-2-3 |
| 0 | AAA/1/1: hex-4-6→hex-4-5 | 8 | AAA/1/1: hex-4-3→hex-4-2 |
| 1 | BB/1/2: hex-1-1→hex-2-2 | 9 | BBB/1/2: passed |
| 1 | B/1/1: passed | 10 | A/1/1: attacked BBB/1/2 |
| 1 | BBB/1/2: passed | 10 | BBB/1/2: hex-2-1→None |
| 2 | A/1/1: hex-1-4→hex-2-5 | 10 | BBB/1/2: was destroyed |
| 2 | AA/1/1: passed | 10 | AA/1/1: hex-2-3→hex-2-2 |
| 2 | AAA/1/1: passed | 10 | AAA/1/1: hex-4-2→hex-4-1 |
| 3 | BB/1/2: hex-2-2→hex-3-2 | 12 | A/1/1: hex-3-2→hex-3-1 |
| 3 | B/1/1: passed | 12 | AA/1/1: passed |
| 3 | BBB/1/2: passed | 12 | AAA/1/1: passed |
| 4 | A/1/1: hex-2-5→hex-3-3 | 14 | A/1/1: hex-3-1→hex-3-0 |
| 4 | AA/1/1: attacked BB/1/2 | | |
| 4 | AAA/1/1: hex-4-5→hex-4-3 | | |
| 5 | B/1/1: passed | | |
| 5 | BB/1/2: passed | | |
| 5 | BBB/1/2: passed | | |
| 6 | A/1/1: attacked B/1/1 | | |
| 6 | AA/1/1: attacked BB/1/2 | | |
| 6 | BB/1/2: hex-3-2→None | | |
| 6 | BB/1/2: was destroyed | | |
| 6 | AAA/1/1: attacked B/1/1 | | |
| 6 | B/1/1: hex-3-1→None | | |
| 7 | B/1/1: was destroyed | | |
| 7 | BBB/1/2: passed | | |

## Final Score

Final score: 559.0

Figure 37. Best DQN Scoring Move Set for Scenario Four

### Scenario Five DQN Moves

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | A/1/1: hex-3-5→hex-3-4 | 9 | BB/1/2: hex-3-3→hex-4-5 |
| 0 | AA/1/1: passed | 9 | B/1/1: hex-3-2→hex-3-3 |
| 0 | AAA/1/1: passed | 9 | BBB/1/3: hex-2-2→hex-3-2 |
| 1 | B/1/1: hex-0-0→hex-0-1 | 9 | BBBB/1/4: passed |
| 1 | BB/1/2: hex-1-0→hex-3-1 | 10 | AA/1/1: attacked BB/1/2 |
| 1 | BBB/1/3: hex-2-0→hex-2-1 | 10 | BB/1/2: hex-4-5→None |
| 1 | BBBB/1/4: hex-3-0→hex-4-1 | 10 | BB/1/2: was destroyed |
| 2 | A/1/1: passed | 10 | A/1/1: passed |
| 2 | AA/1/1: passed | 10 | AAA/1/1: passed |
| 2 | AAA/1/1: passed | 11 | B/1/1: hex-3-3→hex-4-5 |
| 3 | B/1/1: hex-0-1→hex-1-1 | 11 | BBB/1/3: hex-3-2→hex-3-3 |
| 3 | BB/1/2: hex-3-1→hex-3-2 | 11 | BBBB/1/4: hex-4-2→hex-3-2 |
| 3 | BBB/1/3: hex-2-1→hex-2-2 | 12 | AA/1/1: attacked B/1/1 |
| 3 | BBBB/1/4: hex-4-1→hex-4-2 | 12 | A/1/1: passed |
| 4 | A/1/1: attacked BB/1/2 | 12 | AAA/1/1: passed |
| 4 | AA/1/1: passed | 13 | B/1/1: passed |
| 4 | AAA/1/1: passed | 13 | BBB/1/3: hex-3-3→hex-3-5 |
| 5 | B/1/1: hex-1-1→hex-3-1 | 13 | BBBB/1/4: hex-3-2→hex-3-3 |
| 5 | BB/1/2: passed | 14 | A/1/1: attacked B/1/1 |
| 5 | BBB/1/3: passed | 14 | B/1/1: hex-4-5→None |
| 5 | BBBB/1/4: passed | 14 | B/1/1: was destroyed |
| 6 | A/1/1: hex-3-4→hex-3-5 | 14 | AA/1/1: attacked BBB/1/3 |
| 6 | AA/1/1: passed | 14 | AAA/1/1: attacked BBB/1/3 |
| 6 | AAA/1/1: passed | 14 | BBB/1/3: hex-3-5→None |
| 7 | BB/1/2: hex-3-2→hex-3-3 | 15 | BBB/1/3: was destroyed |
| 7 | B/1/1: hex-3-1→hex-3-2 | 15 | BBBB/1/4: passed |
| 7 | BBB/1/3: passed | 16 | A/1/1: hex-5-5→hex-4-5 |
| 7 | BBBB/1/4: passed | 16 | AA/1/1: hex-4-6→hex-3-4 |
| 8 | A/1/1: hex-3-5→hex-5-5 | 16 | AAA/1/1: hex-3-6→hex-2-5 |
| 8 | AA/1/1: passed | 17 | BBBB/1/4: passed |
| 8 | AAA/1/1: passed | 18 | AA/1/1: attacked BBBB/1/4 |
| 9 | BB/1/2: hex-3-3→hex-4-5 | 18 | AAA/1/1: attacked BBBB/1/4 |
| 9 | B/1/1: hex-3-2→hex-3-3 | 18 | BBBB/1/4: hex-3-3→None |
| 9 | BBB/1/3: hex-2-2→hex-3-2 | 18 | BBBB/1/4: was destroyed |
| 9 | BBBB/1/4: passed | | |

### Final Score

Final score: 300.0

Figure 38. Best DQN Scoring Move Set for Scenario Five

Scenario Six DQN Moves

| Phase Number | Move Description | Phase Number | Move Description |
|---|---|---|---|
| 0 | AA/1/1: hex-1-4→hex-3-3 | 6 | A/1/1: passed |
| 0 | AAA/1/1: hex-0-5→hex-2-4 | 6 | AAA/1/1: passed |
| 0 | A/1/1: passed | 7 | B/1/1: hex-5-1→hex-3-2 |
| 1 | B/1/1: passed | 7 | BB/1/2: hex-5-2→hex-3-3 |
| 1 | BB/1/2: passed | 8 | A/1/1: attacked B/1/1 |
| 2 | A/1/1: hex-0-4→hex-2-3 | 8 | AAA/1/1: attacked B/1/1 |
| 2 | AA/1/1: hex-3-3→hex-4-3 | 8 | B/1/1: hex-3-2→None |
| 2 | AAA/1/1: passed | 8 | B/1/1: was destroyed |
| 3 | B/1/1: passed | 9 | BB/1/2: passed |
| 3 | BB/1/2: passed | 10 | A/1/1: attacked BB/1/2 |
| 4 | A/1/1: passed | 10 | AAA/1/1: attacked BB/1/2 |
| 4 | AA/1/1: passed | 10 | BB/1/2: hex-3-3→None |
| 4 | AAA/1/1: passed | 10 | BB/1/2: was destroyed |
| 5 | AA/1/1: hex-4-3→None | 12 | AAA/1/1: hex-2-4→hex-2-5 |
| 5 | AA/1/1: was destroyed | 12 | A/1/1: passed |
| 5 | B/1/1: passed | 14 | AAA/1/1: hex-2-5→hex-2-6 |
| 5 | BB/1/2: hex-6-1→hex-5-2 | | |

Final Score

Final score: 50.0

Figure 39. Best DQN Scoring Move Set for Scenario Six

In summary, the DQN AI agent has successfully navigated the wargaming environment. Its consistent superior performance across all scenarios in comparison to both the MCTS agent and human benchmarks is a testament to its effective learning. The DQN AI agent's score maximization, and robust performance across various game instances underscore the advantages of integrating DQN AI into naval wargaming.

## D. ALPHAZERO AGENT

The AlphaZero agent was designed on the S. Thakoor et al. generic framework implementation of AlphaZero [8]. The implementation of their framework focuses on a self-play concept that uses the MCTS algorithm to improve policy improvement and a neural network for policy estimation.

### 1. AlphaZero Results

The AlphaZero agent versus pass-agg results, seen in Table 8, was not indicative of being able to find the best path optimally.

Table 8. Mean Scores for AlphaZero vs. pass-agg on Each Scenario, N=50

| AlphaZero vs. pass-agg | Mean Score | Standard Deviation |
|---|---|---|
| Scenario1-MiddleIsland | 0 | 0 |
| Scenario2-Mainland_and_Island | -834.5 | 15.18796702 |
| Scenario3-MultiIsland | 0 | 0 |
| Scenario4-LandApproach | -996.0 | 0 |
| Scenario5-Chokepoint | -298.75 | 5.59 |
| Scenario6-TightChannel | 0 | 0 |

AlphaZero struggled to optimize its play against the pass-agg algorithm in the given scenarios. The pass-agg algorithm adopts a reactive approach, adjusting its actions based on the strength of its units relative to the enemies. It will adopt a defensive stance when it has fewer units, waiting for the enemy to attack, and will aggressively attack when it has more units.

In scenarios where the pass-agg algorithm adopted a defensive stance (Scenarios One, Three, and Six), AlphaZero made no significant moves, leading to draws in these games. This might be due to AlphaZero's expectation of action from the opponent as part of its learned policy strategy. Suppose the opponent is passive and does not move at all. In that case, AlphaZero may struggle to learn an effective policy because there needs to be more variety in the opponent's actions to inform AlphaZero's policy updates. This underscores a limitation in AlphaZero's self-play approach: it is optimized for games where both players are actively making moves and struggles when facing a highly passive opponent.

In the other scenarios, the AlphaZero agent seemed to have learned to take offensive actions, such as shooting back at pass-agg once pass-agg units came within range, Scenario Five. However, these actions were not sufficient to win the game. For Scenarios Two and Four, the AlphaZero agent allowed the pass-agg agent to capture the

city, decreasing the total score until the end. AlphaZero's performance in these scenarios indicates that it has not learned an effective policy for these particular game situations. The slight standard deviation in Scenario Two suggests that AlphaZero's actions, such as firing on the enemy in range, needed to be more consistent.

### 2. Poor Results Analysis of AlphaZero

AlphaZero's policy and value networks did not assign a high enough expected reward to proactive movements toward the enemy units, leading to a more reactive and defensive strategy. This behavior may result from the self-play training method, where AlphaZero learns successful strategies and behaviors in games against itself.

Regarding the non-symmetrical nature of the forces in self-play, this could have posed a significant challenge for an algorithm like AlphaZero. One of the assumptions of AlphaZero's training approach is that games are symmetrical, meaning that the same rules and opportunities apply to both players. This assumption is inherent in games like Chess or Go, for which AlphaZero was initially designed [10]. In the scenarios used, the forces were not symmetrical. This was purposely designed to limit the situation of a draw being the optimal outcome. AlphaZero conducted self-play between two non-symmetrical forces, which may have developed policies and value estimates skewed by asymmetry. For instance, if one side had significantly stronger units, AlphaZero might learn that aggressive play is rewarded, but this policy would not necessarily translate well when the roles are reversed. This could explain the agent's inconsistent actions and poor performance in these scenarios. Figures 40,41,42,43, 44, and 45, show the *pi* and *v* values associated with the neural net training for each model trained on their scenario. The "Loss_pi" measures the difference between the predicted probabilities and the actual result of the game. "Loss_v" measures the difference between the predicted outcome and the actual result of the game [8]. In all graphs, both "Loss_pi" and "Loss_v" show a general decreasing trend, indicating that the model is learning and improving its predictions over time. The decrease is not smooth due to the inherent randomness in the training process, but the overall trend is clear. Including the city hex in scenarios Two

and Four resulted in "Loss_v" struggling to predict the score as the city capturing significantly alters the final game score.

The results underscore the importance of considering the characteristics of the training environment and the game dynamics when deploying reinforcement learning algorithms like AlphaZero. Additional adjustments to the learning algorithm or the training setup may be necessary when dealing with non-symmetrical scenarios.

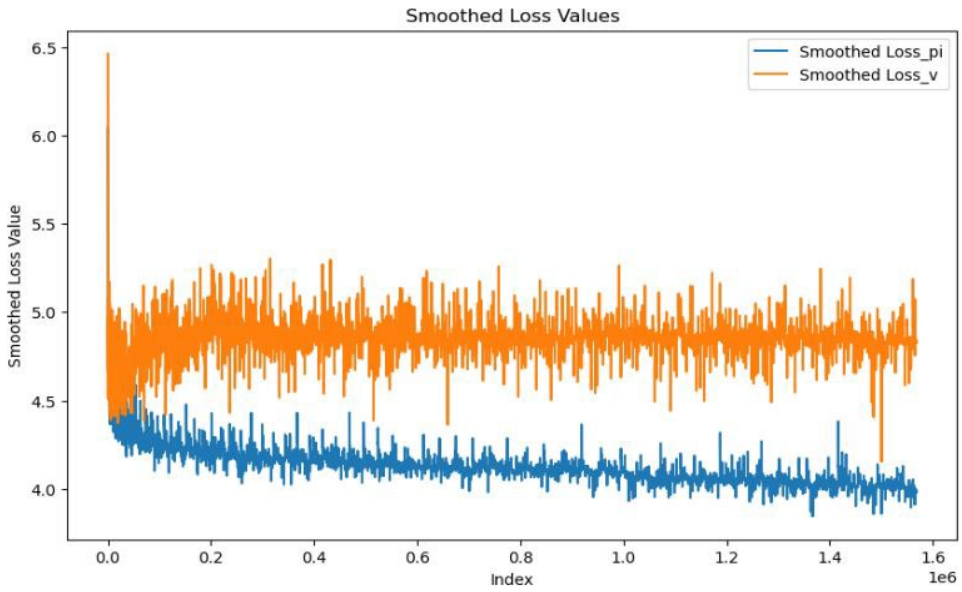


Figure 40. AlphaZero Scenario One *pi* and *v* Values

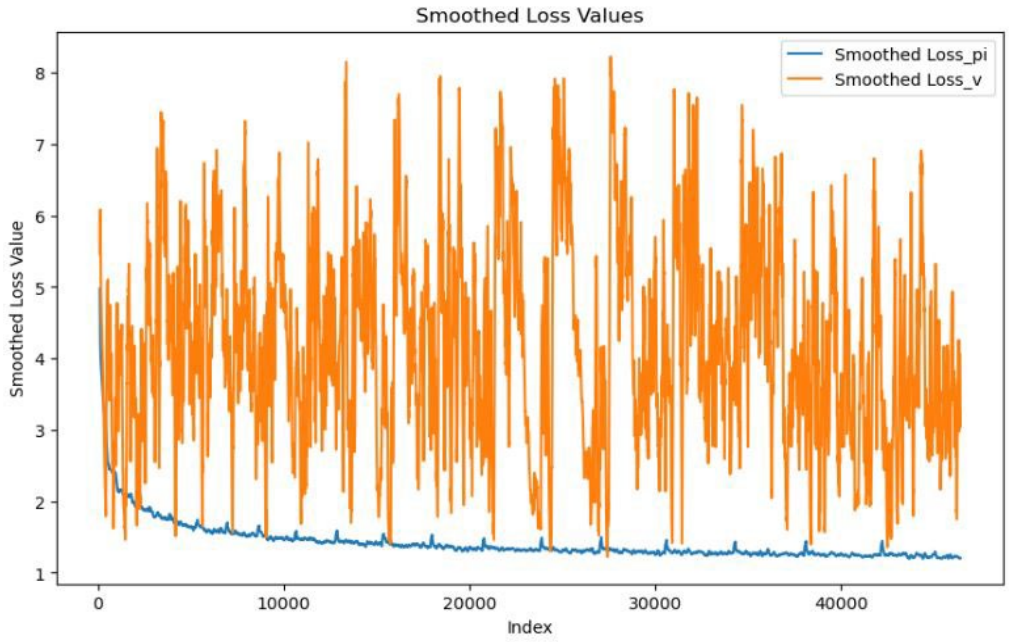


Figure 41. AlphaZero Scenario Two *pi* and *v* Values

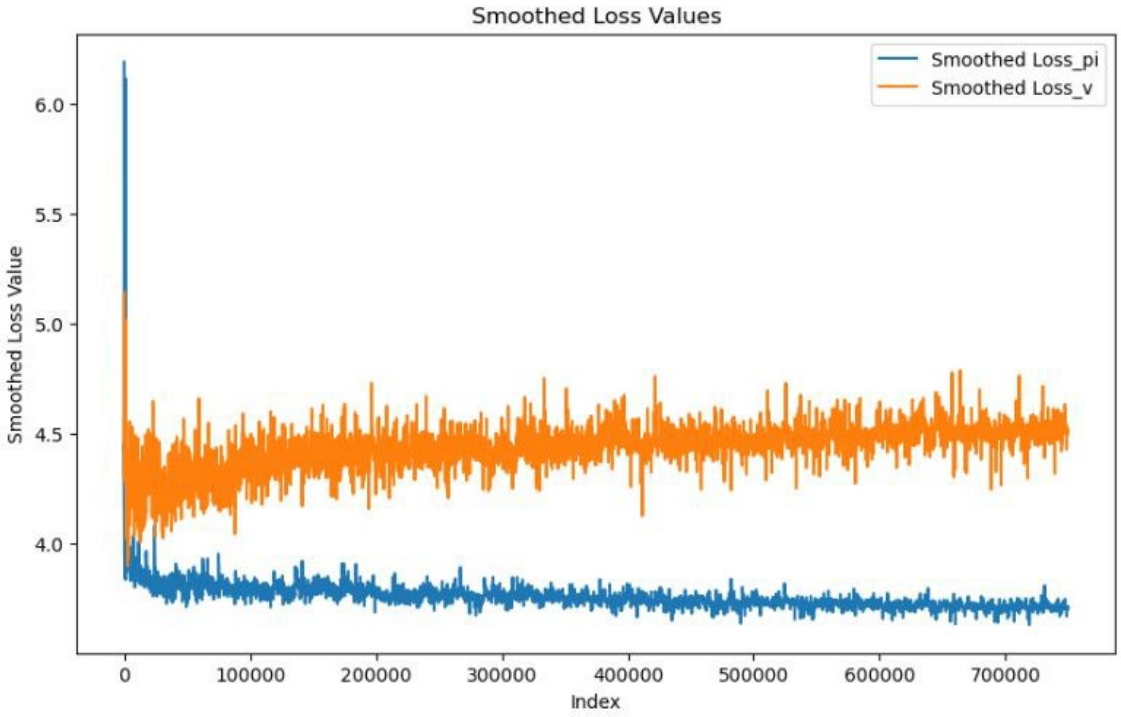


Figure 42. AlphaZero Scenario Three *pi* and *v* Values

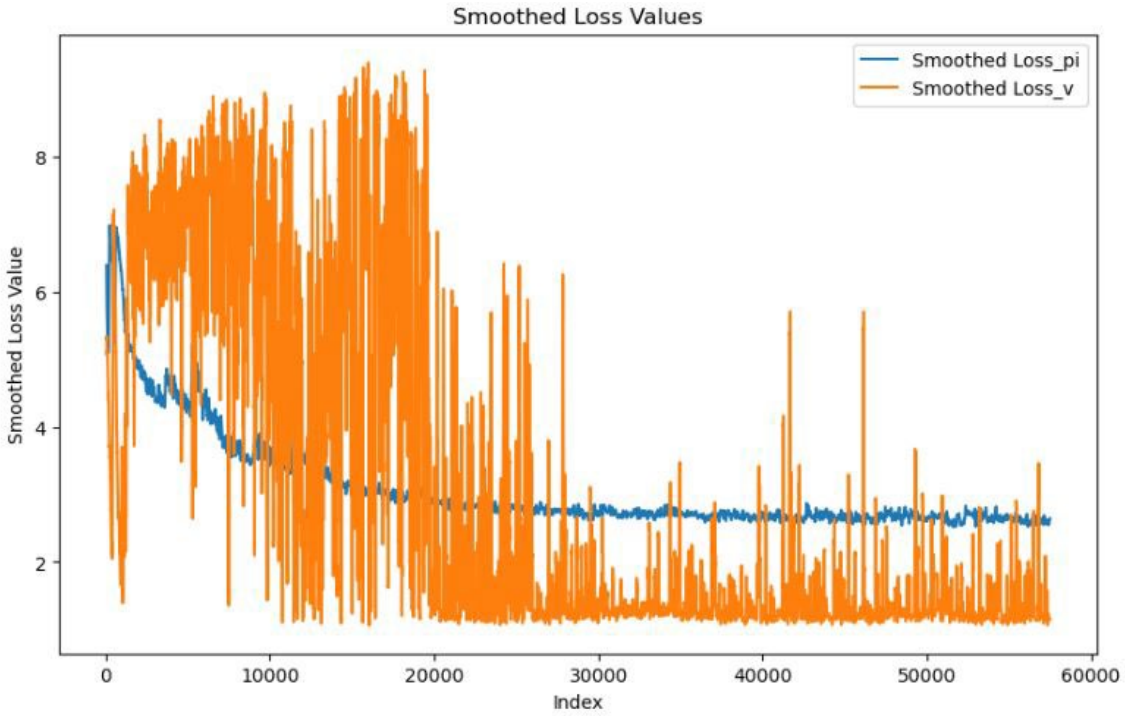


Figure 43. AlphaZero Scenario Four *pi* and *v* Values

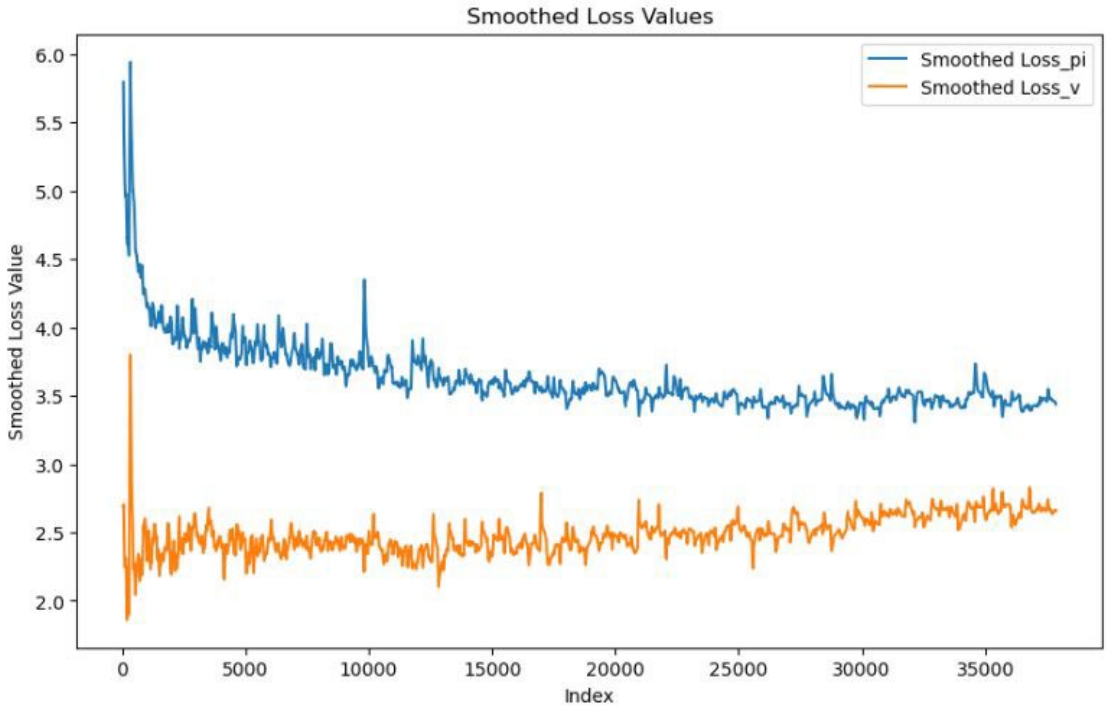


Figure 44. AlphaZero Scenario Five *pi* and *v* Values

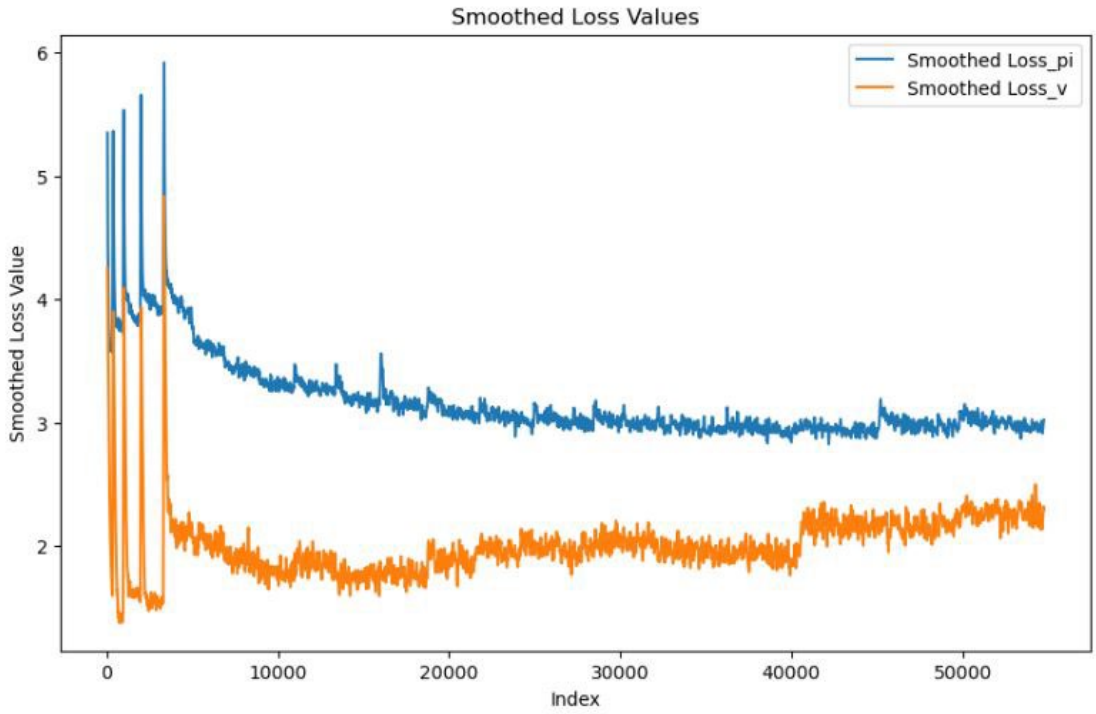


Figure 45. AlphaZero Scenario Six *pi* and *v* Values

## E. SUMMARY OF RESULTS

In the comprehensive analysis presented in this thesis, the DQN agent emerged as a consistently high performer across all scenarios (see Table 9), excluding the human agent. The remaining agents, pass-agg, MCTS, and AlphaZero struggled to score well consistently despite some scenarios where they had a positive mean score or high max score.

It is important to note that comparing mean scores to the optimal human scores presents a challenge, as the mean score encapsulates all performances, including both high and low scores, while the optimal score represents the best possible outcome assuming the human would never make a mistake in their moves.

Table 9. Summary of All Agent Mean and Max Scores on Each Scenarios

| Agents | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 | Scenario 6 |
|---|---|---|---|---|---|---|
| AlphaZero Mean | 0.00 | -834.50 | 0.00 | -996.00 | -298.75 | 0.00 |
| AlphaZero Std | 0.00 | 15.19 | 0.00 | 0.00 | 5.59 | 0.00 |
| AlphaZero Max | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| DQN Mean | 12.23 | 132.74 | 25.04 | 72.32 | 231.14 | 23.65 |
| DQN Std | 78.19 | 138.09 | 57.85 | 279.22 | 82.26 | 66.18 |
| DQN Max | 50.00 | 175.00 | 50.00 | 559.00 | 300.00 | 50.00 |
| MCTS 30k Max | 50.00 | 175.00 | 50.00 | -333.00 | 250.00 | 50.00 |
| MCTS 30k Mean | -16.25 | -11.58 | -6.25 | -464.50 | 168.13 | -27.50 |
| MCTS 30k Std | 36.52 | 146.67 | 26.75 | 90.59 | 96.44 | 47.92 |
| MCTS 50k Max | 0.00 | 298.50 | 0.00 | -333.00 | 275.00 | 12.50 |
| MCTS 50k Mean | -3.75 | -354.65 | -11.25 | -450.13 | 214.06 | -29.38 |
| MCTS 50k Std | 16.77 | 184.34 | 27.48 | 68.61 | 51.13 | 38.32 |
| pass_agg Mean | -125.00 | -352.00 | -101.38 | -802.88 | 115.50 | 22.50 |
| pass_agg Std | 0.00 | 0.00 | 69.18 | 25.94 | 79.20 | 25.09 |
| pass-agg Max | -125.00 | -352.00 | -24.00 | -783.50 | 225.00 | 62.50 |
| Human Scores | 100.00 | 271.00 | 100.00 | 121.75 | 250.00 | 100.00 |

# V. CONCLUSIONS AND FUTURE WORK

The research performed in this thesis has allowed for a comprehensive understanding of how AI agents can be applied in a naval-centric scenario-based framework. This thesis took an existing land-based wargaming simulation, Atlatl, and created a naval variant to test naval scenarios. Evaluating DQN and AlphaStar AI agents in different scenarios offered a rich understanding of their potential and limitations. This chapter concludes the research and suggests directions for future investigations.

## A. CONCLUSIONS

This thesis makes several key conclusions about utilizing DQN, Monte Carlo tree search (MCTS) and AlphaStar for Naval wargames based on the experiments performed with various Naval scenarios. Moreover, it provides evidence that integrating AI in wargaming is feasible when the appropriate processes are applied.

A significant finding from this thesis was the exceptional performance of the Deep Q-Network (DQN) agent in the scenario-based framework. The DQN agent demonstrated a promising capability to identify optimal strategies in different situations, outperforming human players in some of the scenarios presented. The increase in reward function and action space led to a more comprehensive understanding of the environment. DQN's robustness and adaptive nature allowed it to generalize and adapt to different operational contexts, thus making it an asset in the U.S. Navy's decision-making processes.

The Monte Carlo Tree Search (MCTS) agent, while not as consistently dominant as the DQN, showed itself to be a viable contender in certain scenarios. It excelled in complex strategic situations where exploration and exploitation of the decision space were key, even surpassing human-level performance in one instance. However, it also highlighted the delicate balance between exploration and exploitation, as an overemphasis on either led to a drop in performance. Despite this, the MCTS agent's ability to deliver superior results in certain scenarios suggests it has potential as a strategic tool in Naval wargames.

AlphaStar, while being a sophisticated AI agent with demonstrated successes in other domains, showed fewer promising results in the context of this research. Despite its capability to operate on highly complex tasks, its performance was suboptimal compared to the DQN agent. This suggests a potential need for further fine-tuning or customization of AlphaStar's learning structure to better adapt to the unsymmetrical environments that hamper its ability to learn to move optimally adequately.

## B. FUTURE WORK

### 1. Optimizing Training Process

This study has presented an initial exploration into the applicability of RL in creating AI agents for computer-based naval wargaming. While the results indicate a promising direction, training these models remains a substantial endeavor. Given the complexity of naval scenarios and the requirements of simulating large-scale wargames, optimizing the model training process for faster generation represents a significant area for future work.

Firstly, parallel computing could be further leveraged to accelerate the training process. Current models often train sequentially, which could be a potential bottleneck. Exploiting multi-threaded and distributed computing techniques may allow for the simultaneous processing of multiple training instances, drastically reducing the overall training time.

Additionally, future work could investigate advanced reinforcement learning techniques, such as asynchronous methods like Asynchronous one-step Q-learning [25], which operates and updates the AI agents' policies in batches concurrently. This would enable agents with different exploration variables to learn from multiple experiences simultaneously, significantly cutting down training durations while increasing robustness of the model.

Lastly, transfer learning, a technique that allows a model trained on one scenario to be re-purposed on a related scenario, could also be employed. This could accelerate the training process by allowing the reuse of already trained models in related scenarios, eliminating the need to train a model from scratch for every new scenario.

### 2. Modifications to the AlphaStar Algorithm

While the AlphaStar algorithm has proven to be a powerful tool in reinforcement learning, some issues were identified in its application in our asymmetric wargaming scenarios. Firstly, the structure of the arena play needs to be revised to handle asymmetric scenarios better. In the current setup, the weaker side tends to develop more conservative strategies, while the stronger side should ideally be more aggressive. This presents an inherent imbalance in the learning process. A potential solution is to develop a training setup that factors in the asymmetric nature of the sides. For instance, varying the strategies and play styles of the neural networks based on the strength of their side could lead to more balanced and effective learning.

Secondly, the way AlphaStar weighs its rewards in the context of naval wargames needs to be further investigated. It is vital to ensure that the rewards accurately reflect the objectives of the game and the behaviors that should be encouraged. The current reward system may not be fully adapted to naval warfare, leading to suboptimal decision-making by the AI agent.

Possible improvements might involve implementing a more dynamic and adaptable reward system. The reward system should be designed to incentivize the AI agent to take actions that result in higher scores more often. This could involve assigning higher rewards to successful aggressive moves when the agent is in a position of strength or summing all the scores in the arena play and using that to decide whether to keep the new model.

### 3. Summary

In closing, this thesis has laid the groundwork for the future integration of AI in naval warfare and operations, helping to illuminate a path toward more effective, responsive, and strategic decision-making tools. Our findings suggest that with further refinement and adaptation to naval contexts, reinforcement learning AI agents could significantly benefit training simulations and real-world naval operations. While much work is ahead, the results are promising and suggest a bright future for AI in supporting forceful backup to decision-making personnel in the U.S. Navy and beyond. This thesis

hopes to push the U.S. Navy to a safer more effective force with the assistance of sophisticated AI integrated at every level of its structure.

# LIST OF REFERENCES


[1] National Transportation Safety Board, "Collision between U.S. Navy Destroyer Fitzgerald and Philippine-flag container ship ACX Crystal, Sagami Nada Bay, off Izu Peninsula, Honshu Island, Japan, July 17, 2017," Washington, DC, USA, Rep. NTSB/MAR-20/02 PB2020-10100, 2017 [Online]. Available: https://maritimecyprus.com/wp-content/uploads/2020/09/Collision-Fitzgerald-ACX-Crystal_c.pdf

[2] National Transportation Safety Board, "Collision between U.S. Navy Destroyer John S McCain and Tanker Alnic MC," Washington, DC, USA, Rep. NTSB/ MAR-19/01 PB2019-100970 [Online]. Available: https://s3.documentcloud.org/ documents/6243999/MAR1901.pdf

[3] J. Winniefeld, "Safety is a matter of principles," U.S. Naval Institute, Feb. 01, 2018. https://www.usni.org/magazines/proceedings/2018/february/safety-matter-principles

[4] P. Dunn, *Sea battle games*. Hemel Hempstead: Model and Allied Publications, 1970

[5] Department of the Navy, "U.S. Naval War College." Accessed Apr. 17, 2023 [Online]. Available: https://usnwc.edu/

[6] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[7] L. Graesser and W. L. Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, 1st ed. Addison-Wesley Professional, 2019.

[8] S. Thakoor, S. Nair, and M. Jhunjhunwala, "Learning to play Othello without human knowledge." Stanford University, Final Project Report, 2016.

[9] A. Géron, *Hands-On Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems.* Sebastopol, CA: O'Reilly Media, 2017.

[10] D. Silver *et al.*, "Mastering Chess and Shogi by self-play with a general reinforcement learning algorithm." 2017 [Online]. Available: ArXiv. /abs/ 1712.01815

[11] G. Chaslot, "Monte-Carlo tree search," PH.D. dissertation, Universiteit Maastricht, Maastricht, Netherlands, 2010 [Online]. Available: https://project.dke.maastrichtuniversity.nl/games/files/phd/Chaslot_thesis.pdf

[12] J. Goodman, S. Risi, and S. Lucas, "AI and wargaming." arXiv, Sep. 25, 2020 [Online]. Available: http://arxiv.org/abs/2009.08922

[13] P. K. Davis and P. Bracken, "Artificial intelligence for wargaming and modeling," *Sage Journals*, EP-68860, Feb. 2022 [Online]. Available: https://www.rand.org/pubs/external_publications/EP68860.html

[14] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, Art. no. 7782, Nov. 2019, Available: doi: 10.1038/s41586-019-1724-z.

[15] Y. Li, "Deep reinforcement learning." *arXiv*, Oct. 15, 2018. Available: doi: 10.48550/arXiv.1810.06339

[16] C. Grosan and A. Abraham, "Rule-based expert systems," in *Intelligent Systems*, in Intelligent Systems Reference Library, vol. 17. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 149–185 [Online]. Available: doi: 10.1007/978-3-642-21004-4_7.

[17] M. Campbell, A. J. Hoane, and F. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, Jan. 2002, Available: doi: 10.1016/S0004-3702(01)00129-1.

[18] Lawrence Aung, "Deep blue: the history and engineering behind computer chess," Illumin Magazine, Mar. 04, 2010. Available: https://illumin.usc.edu/deep-blue-the-history-and-engineering-behind-computer-chess/

[19] C. Darken, "Atlatl." Monterey, CA, USA [Online]. Available: https://gitlab.nps.edu/cjdarken/atlatl

[20] P. R. Rood, "Scaling reinforcement learning through feudal multi-agent hierarchy," Thesis, Monterey, CA; Naval Postgraduate School, 2022 [Online]. Available: https://calhoun.nps.edu/handle/10945/71091

[21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[22] C. Darken, "Machine-learning tools," Monterey, CA, USA. [Online].

[23] OpenAI, "Proximal policy optimization – spinning up documentation," OpenAI Spinning Up, 2018. Available: https://spinningup.openai.com/en/latest/algorithms/ppo.html

[24] J. A. Boron, "Developing combat behavior through reinforcement learning," M.S. Thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2020 [Online]. Available: https://calhoun.nps.edu/handle/10945/65414

[25] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning – Volume 48*, in ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1928–1937.Available: https://doi.org/10.48550/arXiv.1602.01783

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California