6-2020

Monte Carlo Tree Search Applied to a Modified Pursuit/Evasion Scotland Yard Game with Rendezvous Spaceflight Operation Applications

Joshua A. Daughtery

MONTE CARLO TREE SEARCH APPLIED TO A MODIFIED PURSUIT/EVASION SCOTLAND YARD GAME WITH RENDEZVOUS SPACEFLIGHT OPERATION APPLICATIONS

THESIS

Joshua A. Daugherty, Master Sergeant, USAF

AFIT-ENG-MS-20-M-000

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-20-M-000

MONTE CARLO TREE SEARCH APPLIED TO A MODIFIED PURSUIT/EVASION SCOTLAND YARD GAME WITH RENDEZVOUS SPACEFLIGHT OPERATION APPLICATIONS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyberspace Operations

Joshua A. Daugherty, BS

Master Sergeant, USAF

June 2020

AFIT-ENG-MS-20-M-000

MONTE CARLO TREE SEARCH APPLIED TO A MODIFIED PURSUIT/EVASION SCOTLAND YARD GAME WITH RENDEZVOUS SPACEFLIGHT OPERATION APPLICATIONS

Joshua A. Daugherty, BS

Master Sergeant, USAF

Committee Membership:

Dr. Kenneth M. Hopkinson
Chair

Dr. Richard G. Cobb
Member

Maj Joan A. Betances, PhD
Member

AFIT-ENG-MS-20-M-000

Abstract

This thesis takes the Scotland Yard board game and modifies its rules to mimic important aspects of space in order to facilitate the creation of artificial intelligence for space asset pursuit/evasion scenarios. Space has become a physical warfighting domain. To combat threats, an understanding of the tactics, techniques, and procedures must be captured and studied. Games and simulations are effective tools to capture data lacking historical context. Artificial intelligence and machine learning models can use simulations to develop proper defensive and offensive tactics, techniques, and procedures capable of protecting systems against potential threats. Monte Carlo Tree Search is a bandit-based reinforcement learning model known for using limited domain knowledge to push favorable results. Monte Carlo agents have been used in a multitude of imperfect domain knowledge games. One such game was in which Monte Carlo agents were produced and studied in an imperfect domain game for pursuit-evasion tactics is Scotland Yard. This thesis continues the Monte Carlo agents previously produced by Mark Winands and Pim Nijssen and applied to Scotland Yard. In the research presented here, the rules for Scotland Yard are analyzed and presented in an expansion that partially accounts for spaceflight dynamics in order to study the agents within a simplified model, while having some foundation for use within space environments. Results show promise for the use of Monte-Carlo agents in pursuit/evasion autonomous space scenarios while also illuminating some major challenges for future work in more realistic three-dimensional space environments.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Kenneth Hopkinson, for his guidance and support throughout the course of this thesis effort. There were several times I thought I was not going to be able to make this accomplishment, and the insight and experience helped push me through those dark times. I would also like to thank my family for the days, nights and weekends lost while I had to squirrel away and work through these struggles. Finally, I want to thank the faculty and committee for providing the patience and resources to ensure I did not fail.

Joshua A. Daugherty

Table of Contents

Page

Page

Page

List of Figures

Page

List of Tables

Page

List of Acronym

Acronym	Definition
AI	Artificial Intelligence
DLNN	Deep Learning Neural Network
ECI	Earth-Centered Inertial
GEO	Geosynchronous Earth Orbit
HCW	Hill-Clohessy-Wiltshire
LEO	Low-Altitude Earth Orbit
MC	Monte-Carlo
MCD	Maximize Closest Distance
MCTS	Monte-Carlo Tree Search
ML	Machine Learning
RL	Reinforcement Learning
RPO	Rendezvous Proximity Operation
TTP	Tactics, Techniques and Procedures
UCT	Upper Confidence Bounds Applied to Trees

MONTE CARLO TREE SEARCH APPLIED TO A MODIFIED PURSUIT/EVASION SCOTLAND YARD GAME WITH RENDEZVOUS SPACEFLIGHT OPERATION APPLICATIONS

I. Introduction

1.1 Overview

Space is rapidly evolving as a critical warfighting domain, as recognized by the recent creation of the Space Force. As the number of satellites continues to grow and their controls become both more autonomous and more sophisticated, the need for better pursuer/evader mechanisms becomes critical to effectively operate and maneuver in space. This is true both for the ubiquitous presence of space junk as well as the possibility of the future need to pursue, evade, and rendezvous between satellites and other space vehicles. This thesis develops a two-dimensional pursuer-evader platform, based on the Scotland Yard game, to test and evolve artificial intelligence and other forms of automation using a simplified set of operating rules to mimic some of the key aspects of space dynamics. The Scotland Yard game was chosen as an effective Monte Carlo Tree Search model had been developed and could be modified within the environment to show how the agent adapts to experimental design changes that partially account for spaceflight dynamics, a foundational step toward an autonomous space defense system. A Monte Carlo algorithm is chosen as a proof of concept in this game environment. The results of this effort shows promise for further development. They also illuminate some of the challenges that remain in future work as development shifts to more realistic three-dimensional cases.

1.2 Motivation

Cyberspace, the application of software to enhance operations, maintenance, and security, has been a key component in numerous defensive domains with the most recent being the addition of space. It is fair to say that computer-based automation and control has been a key component of space operations from the launch of the first spacecraft to the manned and unmanned space systems in orbit today. As information technology has modernized and modularized space systems, more nations have developed and ran their own space programs. Advancements in cyberspace have also enabled enhanced security ranging from better cryptography, artificial intelligence to monitor and secure telecommand structures in orbiting satellites, and a variety of other enhancements.[1, 2] Additionally, artificial intelligence (AI) and machine learning (ML) models have been vital in improving and optimizing space system mission performance.[3, 4] Deep Learning Neural Network (DLNN) models of open-looped and closed-loop controls were used to determine the best maneuvers for rendezvous proximity operation (RPO) missions which include space station docking procedures and close proximity maneuvers of geosynchronous-belt inspection.[5] As the space domain is now an official warfighting domain and the United States creation of a new Space Force military branch to contend with adversarial threats, cyberspace is vital component to achieving and maintaining space superiority.

Many questions exist as to how traditional tactics, techniques, and procedures of hostile warfighting applications project in the space domain. Without historical data to complete concrete methods of tactics, techniques and procedures (TTP), simulations provide the best model to project and predict adversarial behaviors given a mission and

circumstantial set of scenarios. Hypothetical situations include, but are not limited to, destroying enemy intelligence gathering, communication, or navigation satellite networks, seizing high value assets from space, and other conventional warfare tactics typically employed in the air domain. These scenarios have expanded problems, such as how a one-on-one dogfight would differ from many spacecraft of two nations battling head-to-head. Other considerations include a concentrated effort of defending a high value space asset against multiple attackers.

AI is a tool to assist answering these complex problems. ML models can quickly simulate scenarios using game theory mechanics and train over time to find an effective to optimal solution for the problem at hand. Cyberspace tools, such as AI, are necessary to leverage superiority in land, sea, air, and space operations. This research focuses on a foundational reinforcement learning (RL) model with a vision toward an autonomous defense, counter-offense system to protect high value space systems. RL was the chosen model for this research as there currently lacks historical data to model the AI to train with. RL learns by playing itself in a virtual state and providing a choice based upon the outcomes of the virtual simulation.

1.3 Research Overview

Given a problem of two spacecraft operating in close proximity with imperfect domain knowledge, this research will demonstrate that a Monte Carlo Tree Search (MCTS) algorithm is an effective ML model. The goal of this research is to provide the MCTS foundation using Scotland Yard as a simplified two-dimensional platform to introduce scenarios of one-on-one to many-on-many simulations.

This research begins an effort toward the creation of an autonomous defensive/counter-offensive system capable of operation with imperfect domain knowledge as a tool to protect high value space systems. This research looks at using a Monte-Carlo Tree Search (MCTS) model to train a system under a given set of conditions to pursue or evade. An evader's objective is to evade capture from a pursuer. Likewise, a pursuer's objective is to capture an evader. Evader position is only given at specific time-state durations making the mechanics MCTS operates in an imperfect domain knowledge. This research is focused on Winands and Nijssen's MCTS implementation and will operate on the Scotland Yard gameboard they used to create their model.[6]

By keeping this stage of research to the MCTS developed by Winands and Nijssen to the Scotland Yard gameboard, we can directly compare how MCTS model performance differs when the model needs to account for some of the spaceflight dynamics principles. While the win rate of the MCTS model is the primary means to measure effectiveness, other factors analyzed in this research include average distance between pursuers and evaders, the amount of time for pursuers to capture the evader and the consideration. These are important factors to carry forward in future iterations as the MCTS models moves into a full three-dimensional simulation where additional factors are applied to the model. The performance metrics mentioned in this paragraph will be defined in Chapter 3.

Given the above discussion, the hypothesis of this research is that the MCTS model created by Nijssen and Winands for the game of Scotland Yard can be employed as an

effective RL model to account for a number of spacecraft running a pursuit-evasion differential game in close proximity.

1.3.1 Research Questions

To support the hypothesis, the following research questions are posed and answered:

1. How can a MCTS model be used to provide a one-on-one to many-on-many pursuit-evasion framework of proximal spacecraft?

2. How can the MCTS algorithm be modularized to support the varying frameworks between one-on-one and many-on-many scenarios?

3. How does the model perform under the following specific circumstances: one pursuer versus one evader operating in a classically constrained gameboard, one pursuer versus one evader opening the gameboard such that all locations are accessible, and five pursuers vs one evader in the classically constrained gameboard?

1.3.2 Research Tasks

The following tasks will be performed to address the corresponding research questions:

1. Create a MCTS algorithm in Scotland Yard using the works of Winands and Nijssen as a model.

2. Modify Scotland Yard program to simulate spaceflight dynamics by programming varying transportation cost between nodes between time states. Modification will also update all routes to taxi routes.

3. Create three experiments, test conditions to measure MCTS performance: One pursuer versus one evader where a win is recorded if the pursuer captures the evader with movement confined to available routes on a classical gameboard, five pursuers versus one evader with same win condition, and one pursuer versus one evader with the same win condition, this time opening the gameboard such that all routes are available between turns.

4. Analyze win rate against Winands and Nijssen's implementation to determine MCTS effectiveness.

5. Analyze and report residual factors for consideration in future work. Residual factors include average node distance between pursuers and evaders from initialization of the game and each round until the game ends, average time required for pursuer wins recorded by the number of turns in each game, and fuel (ticket) consumption during gameplay.

1.3.3 Scope and Assumptions

This research takes the MCTS implementation of Winands and Nijssen in Scotland Yard and applies some of the spaceflight dynamics principles when transitioning from one position to another. The surrogate model based upon Winands and Nijssen provides valid and useful results transferrable to space

applications. Three dimensional models are out of scope and will be considered in future work.

The main principle this MCTS model uses the Scotland Yard environment is a simplified model of the Hill-Clohessy-Wiltshire orbital relative motion dynamic, in that satellite nodal positions rotate as Earth completes its orbit around the Sun, therefore, carrying a varying cost to transition to nodes on different time states.[7] Graph traversal was simplified so the model can operate on a common consumption cost (fuel) that would happen in a space environment. With this research limited to the Scotland Yard gameboard, these principles have been simplified and therefore are not a perfect mathematical correlation to spaceflight but are assumed sufficient to mimic the actual behavior.

Other factors considered, but not implemented in this research was the control objective function for differential pursuit-evasion scenarios and opening the traversal graph to all game nodes between time states. The control objective function was considered an out of scope factor due to not being able to fully integrate the three-dimensional control within a two-dimensional gameboard with limited nodes. The decision to keep original graph traversal was to maintain balance on the limited nodes on the Scotland Yard gameboard as compared to satellite nodal position which are boundless. While graph connectivity was maintained to original game mechanics, traversal routes were all changed to taxi routes so that the correlation between two-dimensional and three-dimensional simulation is more comparable to energy consumption between the two models.

Finally, this research is assumed to use imperfect domain knowledge as information about the evader's location is only known at certain time intervals to the pursuers and not known during the full duration of the game.

1.4 Thesis Outline

Chapter 2 provides the background research used to create the MCTS model and manipulate the Scotland Yard gameboard to account for spaceflight dynamics necessary to transition AI to three-dimensional simulations. Chapter 3 describes the methodology to design the tests that examine how the MCTS model performs under specific conditions. Chapter 4 expands on the results of Chapter 3 to examine MCTS performance and residual factors. Finally, Chapter 5 describes how results support the hypothesis and identifies future work toward creating an autonomous defense, counter-offense model capable of protecting high value space systems from possible adversarial threats.

II. Literature Review

2.1 Overview

Artificial intelligence (AI) has a rich history of aiding research to solve complex problems. AI has had exponential industry and marketing growth to aid with using big data mining collections to push product to general commercialized marketing of AI agents and supercomputing for optimizing corporate operations and profits. Additionally, AI coupled with game theory has enabled researchers and engineers to develop innovative tactics and techniques used in communication, industrial, medical and military operations. This chapter begins by reviewing game theory history. Section 2.3 describes varying AI models and how reinforcement learning (RL) models are most useful in game theory applications. Section 2.4 gives an overview of search algorithms: αβ, Min-Max and Monte-Carlo with Upper Confidence Bounds Applied to Trees (UCT) are discussed. Section 2.5 introduces spaceflight dynamic applications.

2.2 Game Theory

Game theory has a long-coupled relationship with AI-focused research.[8] This section describes how game theory is combined with many machine learning models to inspire and aid researchers to solve complex problems. This section begins by describing game theory mechanics and focus. Section 2.2.2 outlines a brief but progressive history of games using AI and evolving AI models. Section 2.2.3 describes varying search techniques or algorithms AI incorporates to build search trees. Section 2.2.4 expands on the evolution of AI models and how the evolution of techniques has

produced more accurate and faster AIs. Finally, section 2.2.5 describes how AI coupled with game theory produces real-world applications in varying industrial fields.

2.2.1 Game Theory mechanics

Game theory, which has been around since the 1940s, enables new and refreshing means of learning by incorporating mathematics and coupling with outlying strategies and competitive environment to increase, improve or optimize an end objective.[8, 9, 10] There are varying game mechanic models to build around whether to target leadership or behavioral tactics, data analytical models, militaristic strategy, among others.[10] This research focuses on game theory mechanics using imperfect domain knowledge for pursuit-evasion differential games.

Perfect domain games deal with games where all moves are present from beginning until end of a game.[6] Examples of perfect domain games include *chess* and *checkers*. Unlike perfect domain games, imperfect domain games have a limited subset of known information to play at certain times in the game.[6, 11] Examples of imperfect domain games include *Poker*, *Go*, *Scotland Yard*, and *Battleship*. This background focuses on machine learning models effective in using imperfect domain knowledge to produce effective strategies in meeting desirable states. Furthermore, this research focuses on expanding the works of NijssenWinands and Nijssen's MCTS model employed in the game of Scotland Yard toward applying the model and game mechanics to operate with spaceflight dynamics.[6]

2.2.2 Games in Artificial Intelligence

As Turning asked “Can machines think”, he proposed a solution to this question using a game of an interrogator correctly identifying which of a test pair is male and which of the test pair is female through a series of questions and answers.[12] This foundational question of “Can machines think” has inspired researchers to build machines capable of challenging, to outperforming, human players. This question led to Arthur Samuel building a machine with a *Checkers* agent and Alex Bernstein’s *Chess* playing agent in 1958.[12, 13] While these agents were rudimentary, they provided the ground work to expand upon machine learning methods which led to *Kaissa*, *Chinook* and *Deep Blue* AI’s capable of besting world champions in *Checkers* and *Chess* in that time.[14] Other games which produced machine learning agents include traditional card games such as *Poker* and *Bridge* as well as exponential state case games such as *Go*, *Kriegspiel*, and *Scotland Yard*.[15] These varying games and the rules and mechanics required to play and win the games divide into separate problem areas which created a multitude of machine learning models for which to effectively solve. The underlying sections will expand upon the history of the algorithms to enhance the AI agents in creating winning solutions of a game and how branching models of machine learning converge into an umbrella of Artificial Intelligence, focusing on a MCTS model implementation using imperfect domain knowledge in pursuit-evasion games.

2.2.3 Algorithm Development

A popular AI algorithm built into games is Min-Max with Alpha-Beta ($\alpha\beta$) pruning.[16, 17] A reason for the popularity is the method to discretize the search space at depth levels, returning the best decision value from a certain depth. This heuristic approach returns the node with the best chance of success against the best move. A problem with this approach is that as games become more expansive, the likelihood of the best move becomes more unlikely due to the state having to be cut off at a much more shallow level than what's needed to evaluate.[18] This leads to the focus of this research, MCTS algorithm component.

The algorithm that drives the MCTS search space is the Upper Confidence Bound applied to Trees (UCT).[6, 11, 15, 18] The general UCT selection strategy is based on the virtual number of wins of a selected node divided by the number of times the node is visited. This strategy produces uneven trees, but usually produces stronger results as nodes are strengthened by the number of times it is visited. A tree is defined as a non-linear, data structure type to search and retrieve information in a hierarchical manner. Other heuristics can be scaled into the UCT to leverage known domain information to build stronger search trees.[6, 11]

2.2.4 Artificial Intelligence Evolution

In Samuel and Bernstein's *Minimax* AI implementation based on *Checkers*, they were able to create agents capable of playing at an amateur level.[12, 13, 14] A major contributing factor was the available memory to build and expand the

agent's tree of available states. Bernstein maximized his agent to available memory by linking a table for current state to a state of pieces that can attack, pieces that can defend, and informational states such as doubled pieces, self-checking, etc.[16] The tree was then limited to a width of seven moves, each having seven outcomes, played out to a maximum depth of four. In this fashion, 2800 states can be evaluated and scored for which the algorithm can decide to execute the 'best' move. While this method eliminates pieces left 'en prise', Bernstein recognized this evaluation method would summarily eliminate moves not having immediate attack or defend consequences leaving chance for better solutions throughout the game.

Kaissa expanded on the works of Shannon and Bernstein, by replacing the width and depth limitations of the depth-first tree traversal and applying the αβ heuristic algorithm to limit the state-space from overloading available memory. [16] Moving back to *Checkers*, work had ceased from Samuel until the early 90's when a team from Duke released *Chinook*.[17] This agent expanded the allowable depth of the Minimax tree to 19, having a much larger domain set to evaluate at a current state and provide an optimal solution. While recognized that this agent may not find the perfect solution at each state, as the depth required to evaluate a perfect solution is over 60 levels and that amount of computation was unavailable and unfeasible.

Deep Blue was an AI integrated by IBM that expanded on the Min-Max theorem to improve depth search to seven levels.[20] Using more computing power than its predecessor, Deep Blue was able to beat the chess world champion at the time. While this agent can continually be improved upon over time with the

concept of Moore's law adding computational power and memory, this method quickly becomes unfeasible for larger game data sets, as the case with *Go*, and games with imperfect domain knowledge, such as *Kriegspiel* and *Scotland Yard*. This led to the development of other machine learning models to build and evaluate optimal moves.

MCTS was a novel method originally devised for the game *Go*.[21] Winands and Ciancarini's work has been instrumental in expanding the UCT method for imperfect domain games such as *Hex*, *Lines of Action*, and *Kreigspiel*.[6, 22, 23] What makes a MCTS model effective in its UCT selection strategy is that the uneven pruning in building the search trees allows the AI to explore deeper paths and explore better decisions in games with a large memory space. Additionally, MCTS models have shown modularity and scalability in that they can be packaged into deep learning neural networks (DLNNs) as well as adding computational evaluation heuristics into UCT selection strategies to aid overall decision making.[3, 4, 6, 22] Implementation strategy impacts AI speed and performance, so model planning should take place to balance the most effective implementation strategy to environment.[4, 23]

2.2.5 Game Theory Application

Game theory has been instrumental in moving many industries forward. Cooperative games have helped drive economic and marketing strategies to levels unseen prior to Nash theory.[9, 24] Game theory has led to novel lifesaving medical procedures as well as training high quality next generation medical

professionals.[25, 26] Game theory and AI have enabled Amazon to dominate the supply chain.[27, 28] Finally, game theory and AI have been used to produce many new and improved military applications for ground, sea, air and space operations.[29]

2.3 Machine Learning

This section provides an overview of machine learning (ML) concepts, focusing on reinforcement learning applied to spaceflight dynamics. This section begins by providing details of different types of ML. Section 2.3.2 focuses on MCTS learning.

2.3.1 Learning Types

ML is the programming technique for computers to take statistical raw data models and form relationships in the data set to predict future behavior of a given problem.[11] Varying features or algorithms create a model family of machine learning methods for how the AI behaves and human in-the-loop interactions.[5] Machine learning concepts have been around since the early 1950's [11], although, the last two decades have brought abought a surge of ML-related research.[14] This surge can be attributed to the rise of computational power, combined with the use of deep learning Figure 2.1 illustrates the varying ML types.

This subsection details the differences between the machine learning models. Section 2.3.1.1 provides an overview of unsupervised learning model and techniques along with some applications. Section 2.3.1.2 gives an overview of supervised learning techniques and applications. Finally, section 2.3.1.3 describes

the reinforcement learning (RL) techniques and its use in game theory.

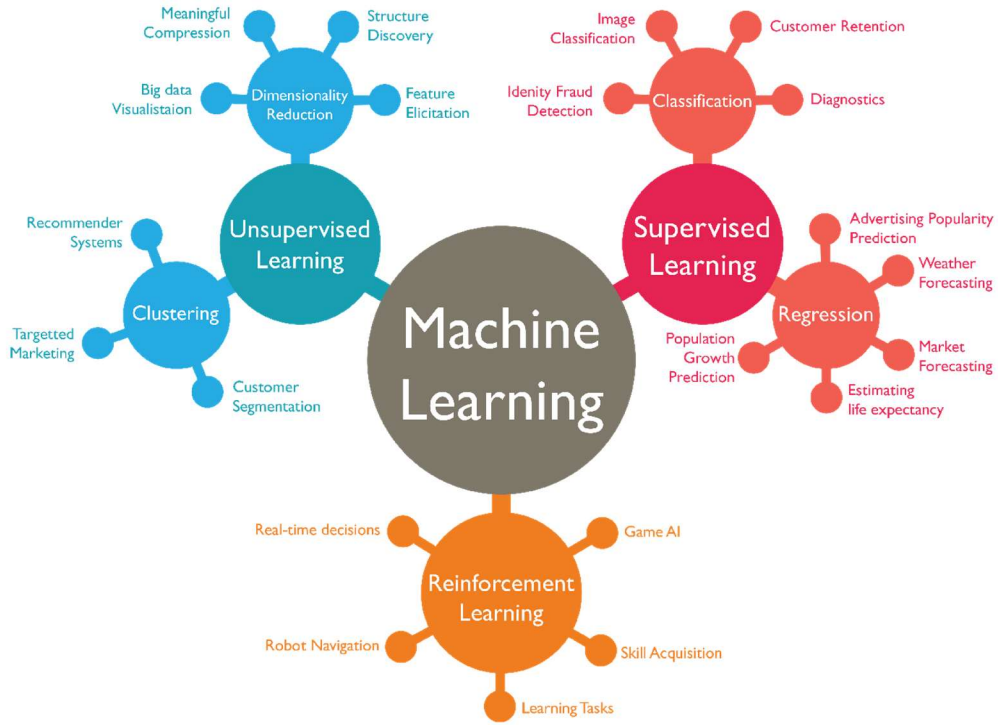

Figure 2.1: Machine Learning Models[30]

2.3.1.1 Unsupervised Learning

Unsupervised learning is the concept of gaining patterns from a series of sensory inputs.[31] Unsupervised learning models sort data into recognizable patterns. This model is used in a lot of big data operations and quantum computing as data can be clustered in groups designed for a specific purpose. Marketing is a leading benefactor from unsupervised learning AI models in personalized advertisements.

2.3.1.2 Supervised Learning

Like unsupervised learning, supervised learning also looks at a pairing/mapping relationship between large amounts of data.[32] Supervised learning then applies a set of rules and heuristics to produce specific output based upon its input. Linear regression is a common heuristic in this model. Supervised learning has numerous applications in the medical, mechanical, communication fields, among others.

2.3.1.3 Reinforcement Learning

Reinforcement learning models identify a collection of input which have a desired effect or output.[33] A reward is programmed as the model learns to achieve the desired state. Reinforcement learning is used in many game theory applications with many varying models as listed in Section 2.2.

2.3.2 Monte Carlo Tree Search Model

Winands and Nijssen have vast experience creating Monte Carlo (MC) agents for a multitude of perfect-domain and imperfect-domain knowledge games including agents built for *Go*, *Lines-of-Action*, *Scotland Yard* and *Ms. Pac-Man*.[34] The MC agent built for *Scotland Yard* has the four basic elements present for most MCTS schemes: *Selection*, *Expansion*, *Playout*, and *Backpropagation*; described in more details in Sections 2.3.2.1 through 2.3.2.4.[6] Additionally, the MCTS scheme employed by Winands and Nijssen incorporated *ε-greedy* playouts for domain knowledge. These playouts add knowledge of node locations for

cooperating *Detectives*, providing a heuristic, *Maximize Closest Distance* (MCD), to calculate the probability of *evader's* next moves. Another heuristic applied to the MC agent is *Determination*. This technique adds hidden information of possible *hider agent* locations using a progressive history of last known locations and transportation ticket cost used to build a list of possible next moves from where the *pursuer agent* has limited *hider agent* possible locations. Next, a bias is applied to approximate most probable node location of the *hider agent* based upon *Location Categorization* factors, which are *minimum-distance*, *average-distance* and *station* (number of available routes at each node). As the method of employment is a cooperative game of pursuers versus a hider, *Coalition Reduction* was employed to achieve a level of aggression and cooperation between the pursuers seeking the hider. This *Coalition Reduction* creates a score of 1 if the *pursuer* is the primary capturer of the *hider* and a value between 0 and 1 dependent if another *pursuer* captures the *hider*.

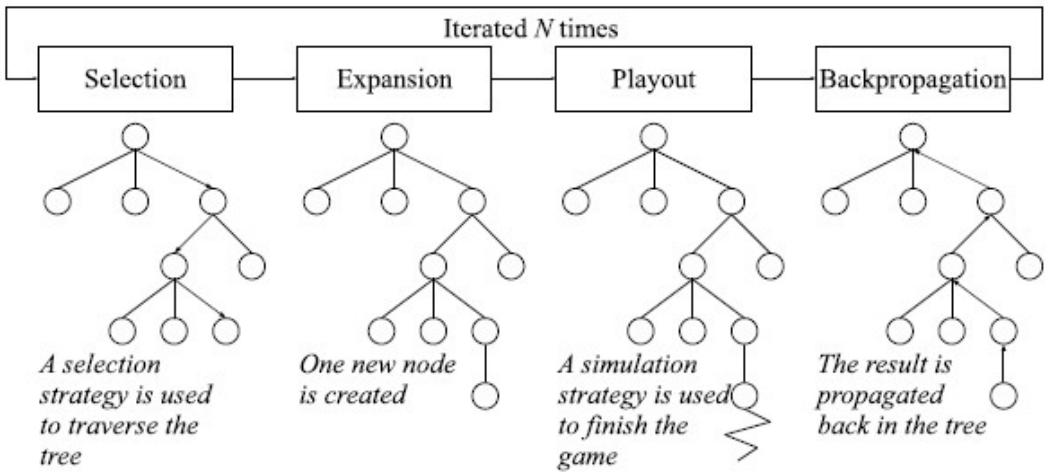

Figure 2.2: MCTS Design

2.3.2.1 Selection

In the selection phase, the search tree is traversed, starting from the root, using the Upper Confidence Bound applied to Trees (UCT) selection strategy. In Winands and Nijssen's Scotland Yard implementation [6], UCT is enhanced with Progressive History using Equation 2.1. This is a combination of Progressive Bias and the history heuristic. The child i with the highest score v_i in Equation 2.1 is selected.

$$v_i = \bar{x}_i + C\sqrt{\frac{\ln(n_p)}{n_i} + W\frac{\bar{x}_a}{n_i(1-\bar{x}_i)+1}} \tag{2.1}$$

Here, $\bar{x}_i$ denotes the average score of node i, n_i and n_p denote the total number of times child i and parent p have been visited, respectively. C is a constant, which balances exploration and exploitation. $\bar{x}_a$ represents the average score of move a, i.e. the average score over all playouts in which move a was played. W is a positive constant that determines the influence of Progressive History. The larger the value of W, the longer Progressive History affects the selection of a node. This selection strategy is applied until a node is reached that is not fully expanded, i.e. not all of its children have been added to the tree yet.

2.3.2.2 Expansion

Expansion is the execution of adding a child to the tree.[35] At the beginning of each turn, the root node begins building the tree by selecting itself

and expanding to first available child on the graph. Through each iteration, the unexplored subset of reachable child nodes is visited at random until all available children have been explored, or a cutoff point is reached.[36]

2.3.2.3 Playout

In playout, the MC agent plays through the newly created child node, recording wins and losses from that position as well as whether the node is terminal (no child states), and if the node yields a better reward state than the parent node. Here a simulation strategy can be incorporated to make playouts more realistic.[37, 38] Complexities of the simulation strategy impact the number of playouts per second the MC agent can execute. Such complexities include but are not limited to computational heuristics, statistical heuristics and domain dependent variables.

2.3.2.4 Backpropagation

Backpropagation feeds the results of the playouts back to the root node for the MC agent to determine best child node to select using the UCT strategy in the selection phase. Results are updated using the formula in Equation 2.1.

2.4 Relative Satellite Motion

When studying the motion of multiple nearby objects in space, typically satellites, in close proximity, or a single object's motion in its local region, the relative coordinate frame is commonly used, referred to herein as the Hill's frame, or more formally as the Hill-Clohessy-Wiltshire frame.[5] In this context, the definition of proximity depends

on the employed dynamics model as well as the altitude and time period of interest. Figure 2.3 shows the relative frame, where *x*, *y*, and *z* represent the relative Hill frame components in terms of the *i, j, k* Earth-centered inertial (ECI) frame.

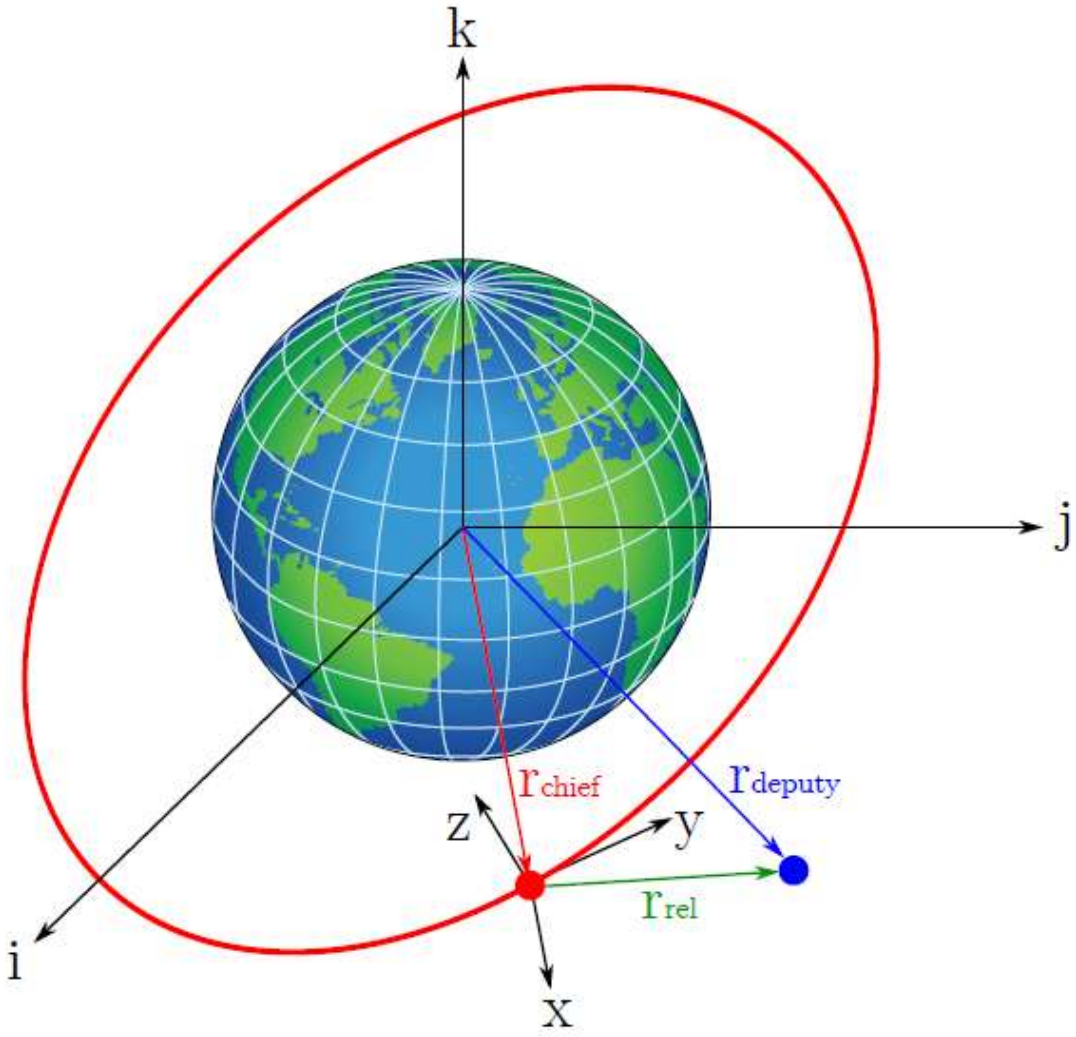

Figure 2.3: Relative Hill Frame [5]

This section gives an overview of some of the relative spaceflight dynamics concerning pursuit-evasion tactics, techniques and procedures. Section 2.4.1 gives an

overview of Hill-Clohessy-Wiltshire (HCW) model. Section 2.4.2 provides the mathematical functions for the various pursuit-evasion controls.

2.4.1 HCW Equations

The Hill-Clohessy-Wiltshire model is a linear model which describes the natural relative motion of objects in close proximity with respect to a circular reference orbit.[5, 7] For this research, the HCW model is introduced with the supporting mathematical matrices. Figure 2.4 displays a co-moving HCW frame.

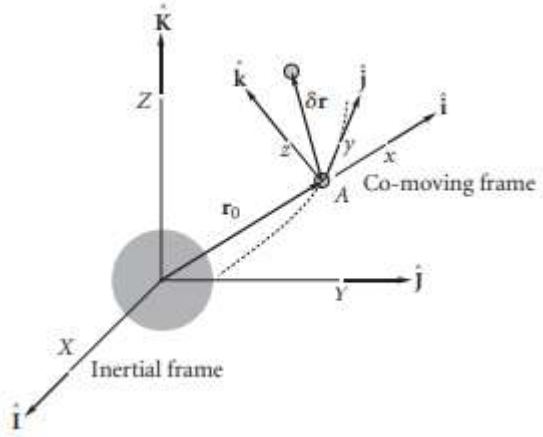

Figure 2.4: Co-moving Clohessy-Wiltshire frame.[7]

The differential equations describing the relative motion in the HCW frame are defined in Equation 2.2.[5, 7] These unforced equations of motion assume no acceleration due to thrust and that the origin is in a circular orbit where *x*, *y* and *z* represent the radial, in-track, and cross-track components with respect to the origin.[5]

$$\begin{aligned} \ddot{x} &= 3n^2x + 2ny \\ \ddot{y} &= -2nx \\ \ddot{z} &= -n^2z \end{aligned} \tag{2.2}$$

The mean motion n is defined by Equation 2.3 such that μ is the standard gravitational parameter and a is the semi-major axis of the specified origin's orbit, and for a circular orbit is directly related to orbit altitude.

$$n = \sqrt{\mu/a^3} \tag{2.3}$$

Equation 2.4 presents the closed-form solution to Equation 2.2 to present an HCW state transition matrix in Equation 2.5.[5, 7]

$$x(t) = \Phi(t)x(t_0), x = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T \tag{2.4}$$

$$\boldsymbol{\Phi}(t) = \begin{bmatrix} 4-3\cos nt & 0 & 0 & \frac{1}{n}\sin nt & \frac{2}{n}(1-\cos nt) & 0 \\ 6(\sin nt - nt) & 1 & 0 & \frac{2}{n}(\cos nt - 1) & \frac{1}{n}(4\sin nt - 3nt) & 0 \\ 0 & 0 & \cos nt & 0 & 0 & \frac{1}{n}\sin nt \\ 3n\sin nt & 0 & 0 & \cos nt & 2\sin nt & 0 \\ 6n(\cos nt - 1) & 0 & 0 & -2\sin nt & 4\cos nt - 3 & 0 \\ 0 & 0 & -n\sin nt & 0 & 0 & \cos nt \end{bmatrix} \tag{2.5}$$

This state transition matrix can be used to efficiently propagate the equations of unforced motion.

2.4.2 Pursuit-Evasion Controls

This section addresses initialization model for two spaceflight objects running pursuit-evasion using collocation method of functions.[39, 40] Equation 2.6 defines the objective function, J, through vectors of evader, E, and pursuer, P. Equation 2.7 defines the constraints for the pursuer and evader. Equation 2.8

defines optimal controls for pursuer and evader. Equation 2.9 defines the costate/adjoint functions for pursuer and evader. Equation 2.10 defines stationary functions for pursuer and evader. Finally, equation 2.11 defines the terminal function for pursuer and evader. (2.6)

$$J(u_E u_P) = \Phi\left(t_f, x_E(t_f), x_p(t_f)\right) + \int_{t_0}^{t_f} \mathcal{L}(t, x_E, u_E, x_P, u_P)dt \tag{2.7}$$

$$\dot{x}_E(t) = f_E(t, x_E, u_E)$$
$$\dot{x}_P(t) = f_P(t, x_P, u_P)$$

$$\dot{\boldsymbol{x}}_E^*(t) = \boldsymbol{f}_E(t, \boldsymbol{x}_E^*, \boldsymbol{u}_E^*)$$
$$\dot{\boldsymbol{x}}_P^*(t) = \boldsymbol{f}_P(t, \boldsymbol{x}_P^*, \boldsymbol{u}_P^*) \tag{2.8}$$

$$\dot{\boldsymbol{\lambda}}_E^*(t) = -\frac{\partial H_E}{\partial \boldsymbol{x}_E}(t, \boldsymbol{x}_E^*, \boldsymbol{\lambda}_E^*, \boldsymbol{u}_E^*)$$
$$\dot{\boldsymbol{\lambda}}_P^*(t) = -\frac{\partial H_P}{\partial \boldsymbol{x}_P}(t, \boldsymbol{x}_P^*, \boldsymbol{\lambda}_P^*, \boldsymbol{u}_P^*) \tag{2.9}$$

$$\boldsymbol{u}_E^*(t) = \arg\max_{u_E} H_E(t, \boldsymbol{x}_E^*, \boldsymbol{\lambda}_E^*, \boldsymbol{u}_E)$$
$$\boldsymbol{u}_P^*(t) = \arg\min_{u_P} H_P(t, \boldsymbol{x}_P^*, \boldsymbol{\lambda}_P^*, \boldsymbol{u}_P) \tag{2.10}$$

$$\boldsymbol{\lambda}_E^*(t_f) = \frac{\partial \phi_E}{\partial \boldsymbol{x}_E^*}(t_f, \boldsymbol{x}_E^*(t_f), \boldsymbol{x}_P^*(t_f))$$
$$\boldsymbol{\lambda}_P^*(t_f) = \frac{\partial \phi_P}{\partial \boldsymbol{x}_P^*}(t_f, \boldsymbol{x}_E^*(t_f), \boldsymbol{x}_P^*(t_f)) \tag{2.11}$$

These functions form the basis for a pursuit-evasion near-optimal solution.

2.5 Scotland Yard

The following section explains the rules to the boardgame Scotland Yard.[41] This section begins by describing the rules for playing. Section 2.5.2 describes the gameplay providing examples of winning strategies.

2.5.1 Rules

Scotland Yard is a boardgame consisting of five *detectives* (pursuers) attempting to capture *Mr. X* (evader) before he escapes from his most recent caper. The gameboard contains 199 possible locations *Mr. X* could be hiding. *Detectives* are given an initial ticket pool of ten taxi tickets, eight bus tickets and four underground tickets. *Mr. X* is given an initial queue of four taxi, three bus and three underground tickets as well as five black fare tickets and two double-move tickets. Gameplay begins with the five *detectives* and *Mr. X* randomly drawing starting locations on the gameboard. *Mr. X* has the first move and the only information revealed to the *detectives* is the mode of travel. As *detectives* spend their fare for the route they travel on their turn, the ticket is given to *Mr. X*. On rounds 3, 8, 13, 18, and 23, *Mr. X* has to reveal his location on the gameboard. *Detectives* have 24 rounds to attempt to capture *Mr. X* before he escapes, winning the game.

These rules and mechanics make this game a two-player imperfect domain knowledge game as *detectives* work as a team to capture *Mr. X*, but have limited knowledge for a period of time. Figure 2.5 shows a subgraph layout for the Scotland Yard gameboard.

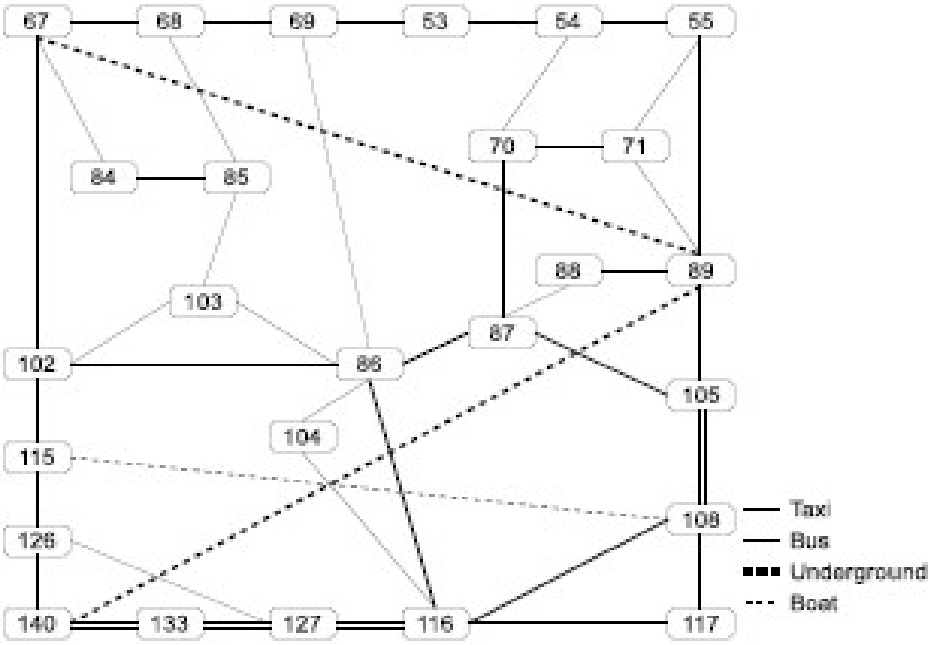

Figure 2.5: Subgraph of the *Scotland Yard* gameboard[6]

2.5.2 Gameplay

An effective strategy for *detectives* to employ as they gain knowledge of *Mr. X's* location is to surround possible escape routes so that, as a team, they can close in and capture *Mr. X*. For example, if *Mr. X's* location is 87 on round 3, as pictured in Figure 2.5, detectives within two nodes should take one of the following locations to limit escape routes before *Mr. X's* location goes dark again: 69, 102, 116, 105, 89, 54 or a closer node if possible. By taking these positions, *Mr. X's* escape routes have chokepoints and detectives can slowly close gap to capture him for the win.

Conversely, after *Mr. X* has had to reveal location and detectives employing routes close to choking escape, that would be the prime opportunity to use one of

the two double-move tickets and use the boat that is only available for *Mr. X's* use. As black fare and double-move tickets are the only tickets limited to *Mr. X*, limiting use to imminent capture and avoiding use on rounds where *Mr. X's* location has to be revealed are good strategies to maximize winning probability.

2.6 Chapter Summary

This concludes the literature review. The literature review began with research in the historical use of AI in game theory. Next, the types of AI were studied to determine that Reinforcement learning models are effective for game simulations. Then, MCTS research was discussed as the method has been popular with pursuit-evasion games with imperfect domain knowledge. Following that, Winands and Nijssen's work with MCTS in the game Scotland Yard was discussed for an effective model to use for application in a spaceflight pursuit-evasion game. Next, some spaceflight dynamic models and controls were discussed to approximate the effects with Winands and Nijssen's MCTS model. Finally, the background of Scotland Yard was discussed in order to recreate the work of Winands and Nijssen, modifying Scotland Yard to approximate some of the spaceflight dynamics effect to design the experiments presented in this research.

III. Methodology

3.1 Chapter Overview

This chapter details the approach in answering the research questions presented in Chapter 1. This chapter begins by restating the research goals and provides an overview on how questions will be answered by this methodology. Section 3.3 details how Scotland Yard was built and modified to approximate the effects of some spaceflight dynamics. Section 3.4 outlines the configuration of the MCTS to evaluate the current state between the pursuer and evader, how the MC agent builds its tree, and how the continuous space is discretized to provide a best move with known domain factors for each turn. Section 3.5 defines how Scotland Yard's rules are transformed to fit a two-dimensional view of rendezvous space objects in close proximity. Finally, Section 3.6 describes the performance metrics used to evaluate implemented MCTS algorithms.

3.2 Research Goals

Recall, given a problem of two spacecraft operating in close proximity with imperfect domain knowledge, this research will demonstrate that a MCTS algorithm can be an effective ML model. This goal therefore is to provide the MCTS foundation using Scotland Yard as a simplified two-dimensional platform to introduce scenarios of one-on-one and many-on-many simulations.

First a Monte-Carlo Tree Search (MCTS) model is used to train a system under a given set of conditions to pursue or defend. In this game, evader position is only given

at specific time-state durations making the mechanics MCTS operates in imperfect domain knowledge. Winands and Nijssen's MCTS implementation is used operating on the Scotland Yard gameboard they used to create their model.[6]

Experiment 1 tests the MCTS original model created by Winands and Nijssen in a one evader versus one hider simulation. Experiment 1 was run 2,500 times. Experiment 1 used the same gameboard transportation restrictions as traditional rules, only modifying the methods described in Section 3.5 as part of the spaceflight dynamic approximation. Experiment 1 results were calculated using the performance metrics detailed in Section 3.6.

Experiment 2 worked the same as Experiment 1, only removing the restrictions of gameboard routes. The entire gameboard is accessible to players between each turn from initialization to the end of the game. Due to the computational burden of this design, only 100 simulations were able to be collected in this design. Results were calculated as detailed in Section 3.6.

Experiment 3 used the classical player team of Scotland Yard of five pursers trying to capture one evader. Gameplay modifications were the same as described in Experiment 1. 2,500 simulations were collected in this design. Results were calculated as described in Section 3.6.

3.3 Scotland Yard Program Design

This section details the basic components to simulate Scotland Yard. Scotland Yard was created in Java as a text-based program of the boardgame. The program is built in

four main modules, the *Main* module which initializes each instantiation of the game, the *Gameboard*, *Players* and *Strategies*. Then there is the Winands and Nijssen MCTS model imported into the game. Section 3.3.1 will describe the game modules in more detail. Section 3.3.2 will describe the components for the MCTS integration.

3.3.1 Game Components

As outlined above, there are four modules to the game creation, the *Main* module, the *Gameboard* module, the *Player* module, and the *Strategies* module. Sections 3.3.1.1 through 3.3.1.4 will describe each module in detail, respectively.

3.3.1.1 Main Module

The main module manages the execution of each game. It houses the methods to call the other modules when needed to play the game. It is configured such that a human could interact as either *pursuer* or *evader* to test functionality. The main module is also where the score is kept for overall *pursuer* and *evader* wins. For each of the three experiments, the main module executes the simulation of *PlayOneGame* from one to *k*. *PlayOneGame* initializes gameboard with players and begins to control the game, having the *evader* move first followed by each *pursuer* player initialized as described in the experiments in Section 3.2. When the *PlayOneGame* concludes with a winner, the win is recorded for either evader or pursuer and the next instance of *PlayOneGame* is executed until the k^{th} game is played and recorded. Upon

conclusion of the k^{th} game and the score updated accordingly, the results are displayed, and program terminates.

3.3.1.2 Gameboard Module

The *Gameboard* module contains all the information to play each game of Scotland Yard. The gameboard module ties into the players module to put players on the gameboard, and the strategies module so that MC agent players can move about the gameboard on their turn with limited knowledge to make decisions as that respective player, *evader* or *pursuer*. The gameboard module contains three submodules to play the game, the *PlayersOnBoard*, *State*, and *Resources*.

The *PlayersOnBoard* submodule contains the information of each *Player* entity on the gameboard along with the information known for that player. This information includes the amount of fuel available for all players. Other player location is limited knowledge and provided as follows:

Pursuer players have the known location for other *pursuer* players. *Pursuer* players then have a distance list that has probable *evader* locations based upon last known location. *Evader* player always has the location of all players.

The *State* submodule contains the information of what the round is, the evaluation methods for determining if the game has been won, and the turn of the current active *Player* entity. The evaluation method examines a win under

two conditions. The first condition is a *pursuer* win if a pursuer moves to the same node as the evader. The second condition is an *evader* win if all players have made their 23rd move without capturing the evader.

The *Resources* submodule contains the map for *Player* entities to make a move on their turn. The corresponding map is available in the *Resources* submodule as listed in the experimental design listed in Section 3.2 and the modifications to the routes as described in Section 3.5. Additionally, the appropriate distance list for *pursuer* or *evader* entities are kept in the Resources submodule.

3.3.1.3 Players Module

The *Players* module houses the entity information to initialize *Player* entities within the *PlayersOnBoard* submodule of the *Gameboard* module. The *Player* entity includes the type of player the entity is: *evader* or *pursuer*. The information within the *Player* entity module include the amount of fuel available and the index of the player so that the player can move when the index matches the current player.

3.3.1.4 Strategies Module

The *Strategies* module contains the MCTS model strategies employed by the MC agents as discussed in Chapter 2. This module supports the MCTS UCT evaluation heuristics in the selection, playthrough and backpropagation phases of training.

3.3.2 Monte Carlo Tree Search Modules

This section describes how the MCTS is integrated into the Scotland Yard program. As discussed in Section 3.3.1, the Main module is built to run MC agents as either evader, pursuer or both player entity types. Section 3.3.2.1 describes the MCTS module and how the module is called. Section 3.3.2.2 describes the MC agent module. Section 3.3.2.3 describes the MCTS state module. Finally, Section 3.3.2.4 describes the MCTS tree module.

3.3.2.1 Monte Carlo Tree Search Module

The MCTS model module has the information necessary to make a deep copy or clone of the current state of the game to pass to the MC agent player on their turn. This module copies the entire gameboard and state information and passes the information into the MC agent's virtual state root node. The model module enables the agent module to train by playing itself in its four-phase iterative style as described in Chapter 2. In the four-phase training cycle, the MCTS tree module is used to build a tree hierarchy of virtual states from simulated play. This module is also linked to the Strategies module for the MC agent to execute its UCT heuristics.

3.3.2.2 Monte Carlo Agent Module

This module contains the Player entity information to act as the player when the player index matches the current state. The MC agent module begins by receiving a deep copy or clone of the current state as the root node of the

MCTS tree module. The MC agent module then uses the MCTS model module to execute its four-phase iterative training cycle to build child nodes virtual states using the MCTS state module and link back to the root node within the MCTS tree module. The MC agent module moves from training to decision by selecting the child node immediately following the root node with the highest UCT score.

3.3.2.3 Monte Carlo Tree Search State Module

The MCTS state module provides virtual state information to load into child nodes of the MCTS tree module. Virtual state information includes the results of that iteration of playthrough as described in Chapter 2. This method allows the MC agent module to train without impacting current state of the game.

3.3.2.4 Monte Carlo Tree Search Tree Module

The MCTS tree module contains the information for the MC agent to build its search tree. It has the parent node which for the root node is null, and any child nodes produced during the expansion phase of training. As discussed in Chapter 2, a tree in an abstract data type creating a hierarchical data structure. Each node within the MCTS tree module contains the MCTS state module information as the MC agent trains the best move from its four-phase iterative training method.

3.4 NijssenMonte Carlo Agent

This section details how the MCTS agent built by Winands and Nijssen [6] is modified to support a one-versus-one and many-versus-many playout of pursuers and evaders. This section begins describes how the MC agent is deployed. Section 3.3.2 describes how this MC agent accomplishes research goals.

3.4.1 Agent Deployment

MC agents are deployed as *evader* and *pursuer* agents in three testing conditions. The first experiment is designed to test performance between one pursuer versus one evader within Scotland Yard's location accessibility as depicted in the subgraph in Figure 2.5. The second experiment tests the one hider versus one pursuer, with an open accessibility between all locations between each turn. The third experiment tests the performance of one evader versus five pursuers. Winning parameters and other metrics analyzed are described in detail in *Section 3.6, Performance Metrics*.

3.3.2 How agent addresses research goals

The MC agent developed here creates a building block for yielding optimal controls for terminally constrained, proximal spacecraft maneuvers scalable to one-on-one to many-on-many pursuit-evasion framework. Scotland Yard was successful as a training tool in developing MC agent to work in a simplified two-dimensional environment. This MC agent is a first step toward an autonomous defense, counter-offense system capable of protecting high-value space systems.

The next step will be to expand the framework built here in an actual three-dimensional space simulation capable of testing agent performance with all dependent spaceflight dynamic principles at work.

3.5 Scotland Yard

This section describes how Scotland Yard was manipulated to support the development of a MCTS model capable of employment on space systems. Section 3.4.1 describes how the gameboard and gameplay mechanics were manipulated to account for some of the spaceflight dynamic principles within a two-dimensional environment. Section 3.4.2 describes how the changes to Scotland Yard accomplish research goals.

3.5.1 Game modifications

Implementing Winands and Nijssen's MCTS model for spaceflight dynamics, the Scotland Yard gameboard was heavily modified to account for some of these fundamental principles. In particular, the transportation between the nodes on the gameboard was altered to correlate fare consumption of Scotland Yard gameplay to fuel expenditure of space systems. This was accomplished by first streamlining fare consumption for all routes to be taxi routes. By making this conversion, the available taxi fare can directly correspond to available ΔV of space systems.

The next modification was a simplified method to approximate HCW dynamics. This was accomplished by dividing the 24 rounds of Scotland Yard's

gameplay into one of three scenarios: Rounds divisible by two, rounds divisible by three, and rounds not divisible by two or three. Table 3.1 details the transportation costs in each of these three cases.

Table 3.1: Varying Transportation Cost Scenarios

Case 1: Round Modulus 2	Node IDs Modulus 2 cost 1 taxi ticket Node IDs Modulus 3 cost 2 taxi tickets Node IDs not Modulus 2 or 3 cost 3 taxi tickets
Case 2: Round Modulus 3	Node IDs Modulus 3 cost 1 taxi ticket Node IDs not Modulus 2 or 3 cost 2 taxi tickets Node IDs Modulus 2 cost 3 taxi tickets
Case 3: Rounds not Modulus 2 || 3	Node IDs not Modulus 2 or 3 cost 1 taxi tickets Node IDs Modulus 2 cost 2 taxi tickets Node IDs Modulus 3 cost 3 taxi ticket

3.5.2 How modifications address research goals

The modifications detailed above provide the foundational testbed to directly compare how the MC agent deployed above compare against the well-designed model initially created by Winands and Nijssen. These modifications address some of the spaceflight dynamics in a simplified environment with a modular MC agent that can be then employed in a fully functional three-dimensional space simulation model to further test performance with full pursuit-evasion tactics.

3.6 Performance Metrics

An algorithm's win ratio provides the best metric for measuring agent's success. Other factors contributing to MC agent's effectiveness include average game length,

average distance, and average ticket consumption to make moves. This section details how results will be analyzed.

3.6.1 MCTS model performance

WinRate provides the primary means to measure MCTS performance. Equation 3.2 shows how the *WinRate* is calculated. A win is scored for each time the pursuer is able to capture the hider. For Block 3, the win is recorded for capturing 7 of the 10 hiders. Each block will have their own *WinRate* calculation.

$$WinRate = \frac{numWins}{numGames} \tag{3.2}$$

3.6.2 Average Win Time

Average win time is the number of rounds it takes the game to produce a winner. In runs that the hider wins, the win time is 24. Therefore, pursuer will have wins between 1 – 23. The average will be the sum of these wins divided by 2,500 runs. Equation 3.3 details the calculation.

$$WinTime = \frac{(t_1 + t_2 + \cdots + t_{2500})}{2500} \tag{3.3}$$

3.6.3 Average Distance

Location Categorization, as listed in Section 2.3.2, incorporates probable locations to look at distance as a measure of performance within the MCTS model. For the *evader* agent, further distances are awarded favorably as where the *pursuer*

agent is awarded for minimal distances. Going from initialization to terminal state, nodal position of evaders to closest pursuer will be calculated in part by the subgraph depicted in Figure 2.5. Using the information of the gameboard map, an adjacency matrix was built. The adjacency matrix if a 199 by 199 matrix showing the distance between the 199 gameboard nodes in a source node, destination node layout. Using this design, the adjacency matrix will have a diagonal line of zero's as the source and destination node is the same node. Figure 3.1 shows a sample of the adjacency matrix. Using the adjacency matrix, average distance will be taken between each round and each experiment and calculated to see how well *evader* and *pursuer* were able to maximize or minimize distance, respectively.

1	0	3	4	3	5	6	7	1	1	3	4	4	2	3	4	5	6	2	2	2	3	3	3	3	4	5	5	5	5	7	3	3	2	2	4	5	4	4	4	4	4	6	3	2	2	1	2	3
2	3	0	3	4	7	8	9	4	2	1	2	4	4	5	6	7	8	5	3	1	2	3	4	5	6	7	7	7	7	9	5	3	2	2	4	5	5	6	6	6	6	8	6	4	4	3	3	3
3	4	3	0	1	5	6	7	5	5	2	1	1	2	3	4	5	6	6	6	4	3	1	1	3	4	5	5	5	5	7	6	5	4	2	2	3	2	4	4	4	4	6	6	5	4	3	3	3
4	3	4	1	0	4	5	6	4	4	3	2	2	1	2	3	4	5	5	5	4	4	2	2	2	3	4	4	4	4	6	5	4	3	3	3	4	3	3	3	3	3	5	5	4	3	2	3	4
5	5	7	5	4	0	3	4	6	6	6	6	5	3	2	1	1	3	7	7	6	6	5	4	4	3	2	3	2	2	4	7	6	5	5	6	6	5	4	3	3	2	3	7	6	5	4	5	6
6	6	8	6	5	3	0	1	7	7	7	7	6	4	3	2	2	2	8	8	7	7	6	5	5	4	3	4	3	1	3	8	7	6	6	6	7	6	5	4	3	2	2	8	7	6	5	6	7
7	7	9	7	6	4	1	0	8	8	8	8	7	5	4	3	3	1	9	9	8	8	7	6	6	5	4	5	4	2	2	9	8	7	7	7	8	7	6	5	4	3	1	9	8	7	6	7	8
8	1	4	5	4	6	7	8	0	2	4	5	5	3	4	5	6	7	1	1	3	4	4	4	4	5	6	6	6	6	8	2	2	3	3	5	6	5	5	5	5	5	7	2	3	3	2	3	4
9	1	2	5	4	6	7	8	2	0	3	4	5	3	4	5	6	7	3	1	1	3	4	4	4	5	6	6	6	6	8	4	2	2	3	5	6	5	5	5	5	5	7	4	3	3	2	3	4
10	3	1	2	3	6	7	8	4	3	0	1	3	3	4	5	6	7	5	4	2	1	2	3	4	5	6	6	6	6	8	5	3	2	1	3	4	4	5	5	5	5	7	5	4	3	2	2	2
11	4	2	1	2	6	7	8	5	4	1	0	2	3	4	5	6	7	6	5	3	2	1	2	4	5	6	6	6	6	8	6	4	3	2	2	3	3	5	5	5	5	7	6	5	4	3	3	3
12	4	4	1	2	5	6	7	5	5	3	2	0	2	3	4	5	6	6	6	5	4	2	1	3	4	5	5	5	5	7	6	5	4	3	3	3	2	4	4	4	4	6	6	5	4	3	4	4
13	2	4	2	1	3	4	5	3	3	3	3	2	0	1	2	3	4	4	4	3	3	2	1	1	2	3	3	3	3	5	4	3	2	2	3	3	2	2	2	2	2	4	4	3	2	1	2	3
14	3	5	3	2	2	3	4	4	4	4	4	3	1	0	1	2	3	5	5	4	4	3	2	2	1	2	3	2	2	4	5	4	3	3	4	4	3	2	2	3	2	3	5	4	3	2	3	4
15	4	6	4	3	1	2	3	5	5	5	5	4	2	1	0	1	2	6	6	5	5	4	3	3	2	1	2	1	1	3	6	5	4	4	5	5	4	3	2	2	1	2	6	5	4	3	4	5

Figure 3.1: Adjacency matrix sample

3.6.4 Average Fuel Consumption

Average fuel consumption will look at the tickets used between each move for each *player* agent in each experiment. Fuel is an important factor for space systems requiring longer longevity and mission parameters could scale to become

a biased priority within the UCT selection strategy, although not for this research. For the purpose of this research, this variable is only to describe how the agent is consuming the resource during gameplay.

3.6 Summary

The methodology laid out in this chapter described the block design to test MC agent performance accounting for simplified spaceflight dynamics. This methodology outlines how research goals are accomplished within the experiments design for testing performance of one-on-one with gameboard travel restrictions, one-on-one with an open gameboard, and one-versus-five confining travel to gameboard routes. This methodology provided how the Scotland Yard environment was created to run the experiments. This methodology described how the Scotland Yard rules were manipulated to approximate some of the effects of spaceflight dynamics. Finally, this methodology describes how performance metrics were to be gathered for analysis.

IV. Analysis and Results

4.1 Overview

This chapter presents the results of the experiments and performance metrics described in Chapter 3. This chapter begins by breaking down the performance metrics described in Chapter 3 in Experiment 1. Section 4.3 details the performance metrics in Experiment 2. Section 4.4 completes the performance metric analysis for Experiment 3. Section 4.5 summarizes the results along with providing some general observations as the MCTS model presented in this research is migrated into 3D space simulations.

4.2 Experiment 1 Result Analysis

This section expands on the performance metrics described in Chapter 3 in Experiment 1. Section 4.2.1 analyzes the win ratio in terms of the pursuer agent along with general observations how the experimental design impacted this metric. Section 4.2.2 analyzes the gameplay in terms of win-time providing general observations. Section 4.2.3 analyzes the average distance between pursuer and hider agent as the game progresses giving general observations noticed in analysis. Finally, Section 4.2.4 analyzes ticket consumption noting general observations.

4.2.1 WinRate Analysis

This section describes how agents performed in Experiment 1 and gives general observations in how experimental design impacted performance. Analyzing pursuer WinRates among the three experiments provided the following results: Experiment 1

yielded a 1.2% ± .43% WinRate, Experiment 2 was 10% ± 5.9%, and Experiment 3 was 93.88% ± .94%. Figure 4.1 displays the WinRate among the three experiments. Results show experimental design was a major factor with Experiment 1 having a low win-rate among the 3 experiments.

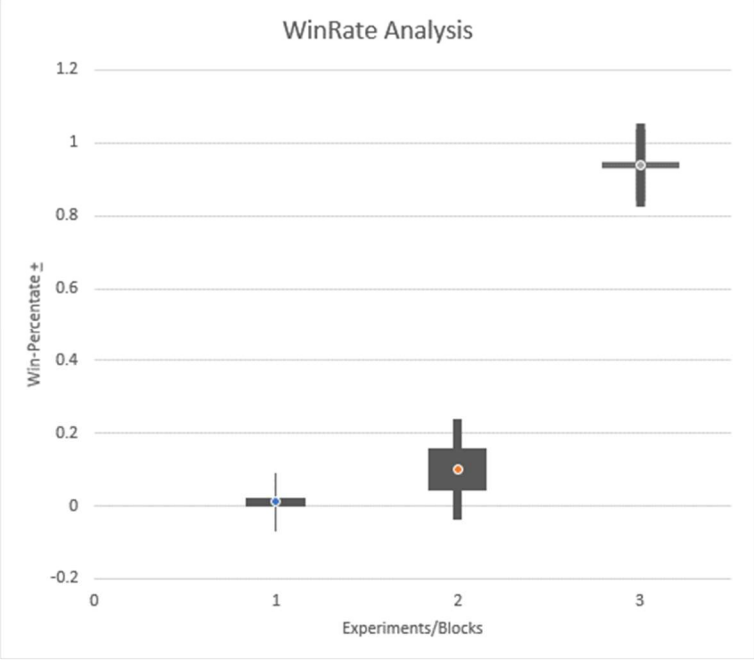

Figure 4.1: WinRate among three experiments showing confidence bounds

Experiment 1 was designed to be advantageous to the *evader* with limited time states of visibility to the *pursuer* and a contained movement gameboard to operate between each turn (e.g. pursuer on Node 53 can only move to nodes 69 or 54 as depicted in Figure 2.5). This advantage was evident as the pursuer was only able to win 30 of the 2,500 runs for a win percentage of 1.2% ± .43%. The analysis in Section 4.2.2 of average distance highlights how the experimental design impacted pursuer agent's ability to score wins.

Analyzing the statistics behind the *WinRate* produced the following results. The squared deviation (x^2) for the population produced by this simulation was 29.64 by taking the sum of the difference wins/losses from the sample mean squared. As we are calculating using sample population, the sample variance ($\hat{s}^2$) was derived by dividing x^2 by the population (n) minus 1 degree of freedom giving a result of .01186. Finally, the sample deviation is the square root of $\hat{s}^2$ which was .1089. Equation 4.1 shows the calculation for sample deviation.

$$\hat{s} = \sqrt{\frac{\sum_{i=1}^{n}(x^i - \bar{x})^2}{n-1}} \tag{4.1}$$

Using the information of the sample deviation, a t-test was calculated on the results using Equation 4.2. The t-value produced -338.36, using Winands and Nijssens' results of 74.9% for the null hypothesis (μ). Given the 2,500-sample size, the corresponding p-value shows less than a .00001, rejecting the null hypothesis and giving significance to the experimental design impacting win-rates.

$$t = \frac{\bar{x} - \mu}{\hat{s}\sqrt{n}} \tag{4.2}$$

With this information, the confidence interval was calculated for 95%. 95% confidence, produces a Z-score of 1.96. With this information and Equation 4.3, the negative confidence bound was .0077 and the positive confidence bound was .0163. Figure 4.2 provides a zoomed graphical view of this data.

$$CI = \bar{x} \pm Z(\hat{s}/\sqrt{n}) \tag{4.3}$$

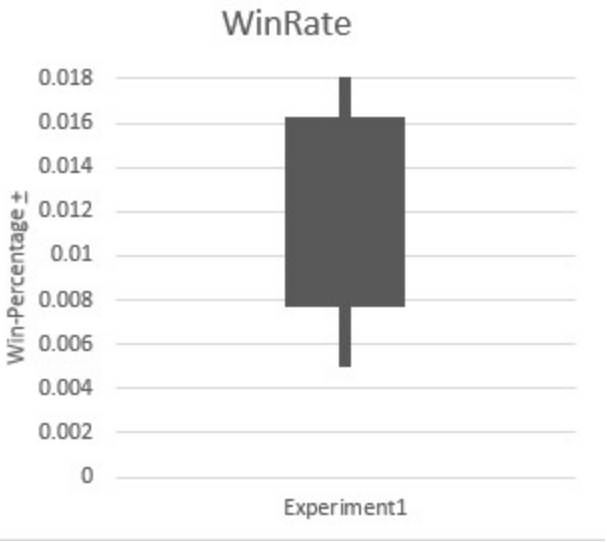

Figure 4.2: Experiment 1 *WinRate* showing confidence bounds

4.2.2 Average Win Time Analysis

Due to the nature of experimental design, *pursuers* were only able to score 2 wins on round 23 with the other 28 wins observed on round 24. As noted above, the gameboard travel restriction was the primary factor in this low result. With both wins on round 23, the *evader* ran out of fuel to move allowing *purser* to capture the hider. However, it was only seen in 7 of the 28 wins on round 24 where the *evader* ran out of fuel to move.

4.2.3 Average Distance Analysis

In this design, transportation on the gameboard is a significant contribution toward poor results for *pursuer* agent. An observation while having one human *pursuer* against an evader agent running a maximum distance bias for decision making, it was difficult for the human player to get any closer than two nodes away

in any turn. Analysis of the MC *purser* agent shows the same struggle to close the distance as the game progresses toward conclusion. Figure 4.2 shows the scatter point average by round in this round. In a more balanced design, the desirable effect would be for the adjacency between hider and pursuer converge toward 0 as the game progresses. Figure 4.3 shows that the pursuer agent plateaued at an adjacency of 2.5 during entirety of simulation.

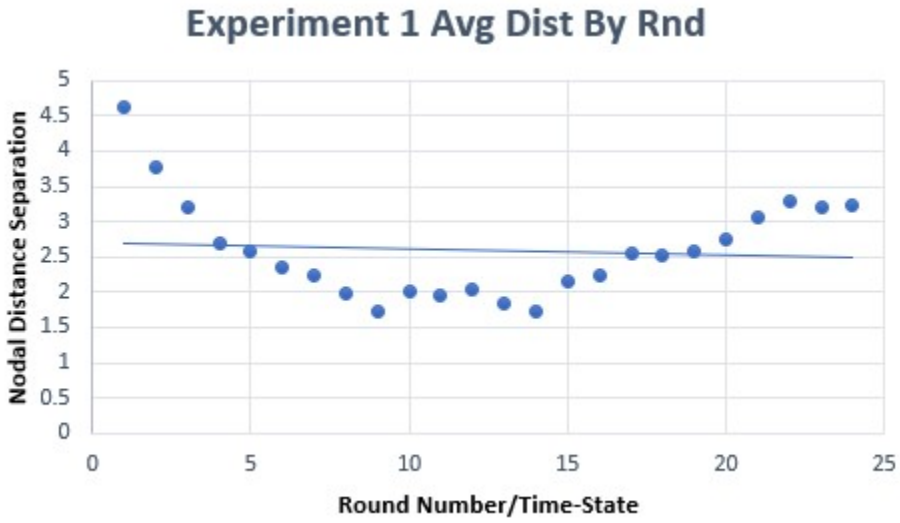

Figure 4.3: Experiment 1 average distance by round

4.2.4 Average Fuel Consumption

General observation of fuel consumption showed good balance between aggression and available fuel for the duration of gameplay for both *evader* and *pursuer* agents. Both agents observed a mean consumption of 1.67 fuel per round. The *evader* observed a slightly wider range of average fuel use per observed game with a low range of 1.21 tickets per round during an observed game to 1.8 tickets.

Likewise, the pursuer had an average ticket use ranging from 1.3 tickets per round to 1.8 tickets in an observed game.

4.3 Experiment 2 Result Analysis

This section expands on the performance metrics described in Chapter 3 in Experiment 2. Section 4.3.1 analyzes the win ratio in terms of the pursuer agent along with general observations how the experimental design impacted this metric. Section 4.3.2 analyzes the gameplay in terms of win time providing general observations. Section 4.3.3 analyzes the average distance between *pursuer* and *evader* agent as the game progresses giving general observations noticed in analysis. Finally, Section 4.3.4 analyzes fuel consumption noting general observations.

4.3.1 Average WinRate Analysis

Experiment 2's design showed to have better balance for the pursuer as the observed win-rate improved to 10 wins from 100 simulations. Experiment 2 needed a smaller sample due to the computational time of the agents between each move. The 100 runs took 2.5 times to complete as the 2,500 runs of Experiment 1 and 3. A big reason is there is a massive state space expansion of an open map for the MC agents to traverse. The available states in this design were 199^{199} while available states were limited in Experiments 1 and 3 to the traditional gameboard routes.

Using Equations 4.1 – 4.3, the following statistics were observed. The sample deviation was .302, t-test result was -21.47, yielding a p-value of less than .00001. Therefore, Experiment 2 is significant and null hypothesis is rejected. The

negative confidence bound was .041 and the positive confidence bound was .159. Figure 4.4 provides the graphical view for Experiment 2's win ratio.

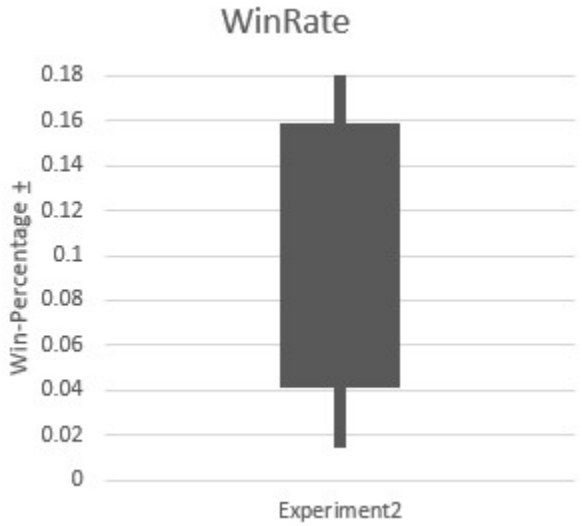

Figure 4.4: Experiment 2 *WinRate* showing confidence bounds

4.3.2 Average Win Time Analysis

Results show that in this design, pursuer agent was able to expand the breadth of its search tree and win some games in earlier rounds. These results also showed that diligence must be taken into consideration to better prune the state space to allow deeper searches and improve overall responsiveness. While the agent in this design was able score a win in rounds 2 and 4 in a simulation, most wins still came in the latter half in gameplay. The average win-time was 23.06, but the computation and response time made simulation run 2.5 times longer than Experiments 1 and 3 only having 1/25 of the samples.

4.3.3 Average Distance Analysis

Applying the traditional gameboard adjacency matrix to look at the average distance between each turn showed that the pursuer agent relied more on the progressive history to predict next move more than a progressive attempt to close the gap as the game progresses. The average distance in this experiment stayed consistently around 4.6 for duration of the game. Figure 4.5 shows the average distance seen in Experiment 2 by round.

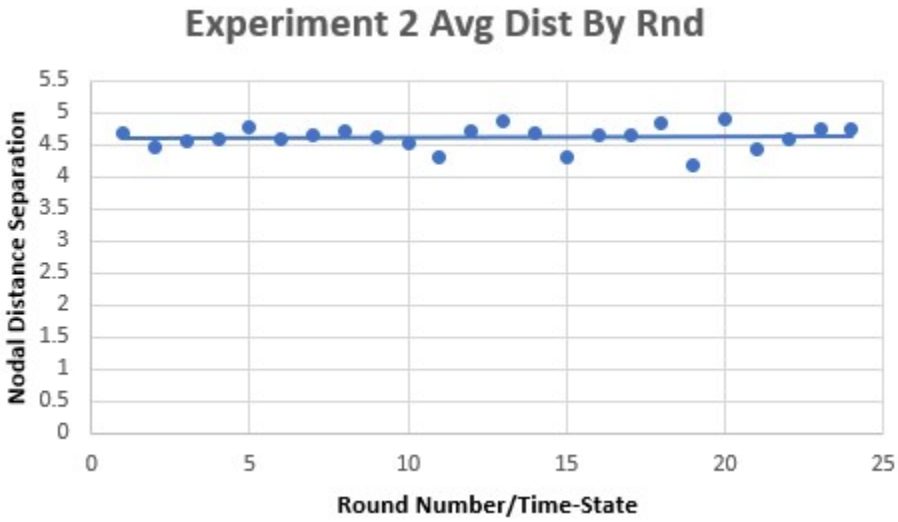

Figure 4.5: Experiment 2 average distance by round

4.3.4 Average Fuel Analysis

Average fuel use in this experiment showed more aggressiveness by both *evader* and *pursuer* with an open gameboard. *Evader* average fuel usage per round increased to 1.74 and *pursuer* increased to 1.76. Game ranges increased as well with *evaders* having low averages of 1.5 fuel per round games and high of 3. *Pursuer* also had peak fuel usage games of 3 but observed a lower floor of 1.31 fuel

per round games. The increased aggression in this experiment shows the need to configure favorability to conserve energy within the UCT algorithm when moving to 3D space simulations in future work.

4.4 Experiment 3 Results Analysis

This section expands on the performance metrics described in Chapter 3 in Experiment 3. Section 4.4.1 analyzes the win ratio in terms of the pursuer agent along with general observations how the experimental design impacted this metric. Section 4.4.2 analyzes the gameplay in terms of win time providing general observations. Section 4.4.3 analyzes the average distance between pursuer and evader agent as the game progresses giving general observations noticed in analysis. Finally, Section 4.4.4 analyzes fuel consumption noting general observations.

4.4.1 Average WinRate Analysis

This experimental design drew on the traditional implementation of Winands and Nijssen's MCTS implementation of Scotland Yard.[6] Results to win-rate were vastly improved over traditional game mechanics for *pursuer* agents. *Pursuers* observed 74.9% $\pm$ 2.7% under traditional rules implementation.[6] Experiment 3 observed 2,347 wins of 2,500 simulations for a 93.88% *WinRate*. It was expected to be closer to original observations with changes to mechanics being balanced on both sides.

Applying Equations 4.1 – 4.3 as with Experiments 1 and 2, observations show significance with experimental design rejecting the null hypothesis. Sample

deviation was .24 with a sample variance of .057. T-value was recorded at 39.576 resulting in a p-value less than .00001. Looking at the upper and lower 95% confidence interval, the negative confidence bound was .929 and the positive confidence bound was .948. Figure 4.6 shows a graphical view of the *WinRate* results.

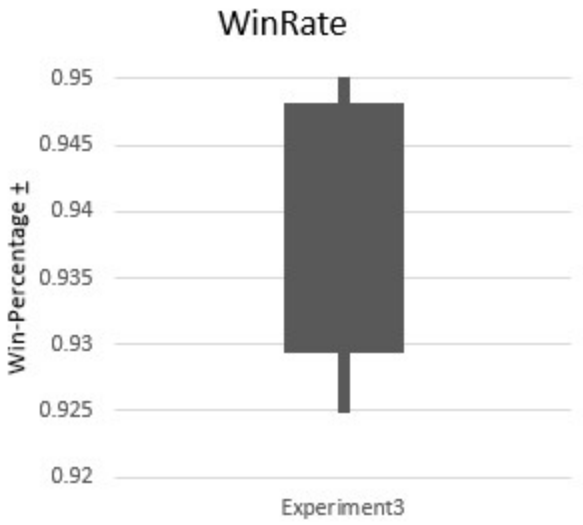

Figure 4.6: Experiment 3 WinRate showing confidence bounds

4.4.2 Average Win Time Analysis

As stated earlier, Experiment 3 performed really well with traditional Scotland Yard play modified for spaceflight dynamics. 5 wins were recorded from the first move and the most frequent round won was round 8 with 386 recorded wins. The distance with the 5 wins were all only 2 nodes away when initialized. The average win was 9.8 rounds of play. Initial distance did not appear to be a problem with wins or losses as *evader* wins were recorded within the same distances as recorded seeker wins at 8 rounds. Given these observations, it appears that the

loss of double *evader* moves may have been the contributing factor resulting in improved seeker performance in this design.

4.4.3 Average Distance Analysis

The manipulations for traditional rules to account for spaceflight dynamics was expected to be balanced for hider and pursuer agents. While all routes became taxi routes and location deduction would be unable to be made with method of travel, balance was applied with removal of black-fare, double-move and balanced queue of tickets to navigate on modified gameboard with varying ticket cost as detailed in Chapter 3. Among the available results, the removal of the double-move fare for hider is the most leading contributor for the observed win increase for pursuer agents.

Moving to the average distance, averages quickly converged toward 0 until the rounds progressed toward the average win time. Then, as many winning simulations had ended, the average distances began to rise. Figure 4.7 shows the average distances for Experiment 3 by round.

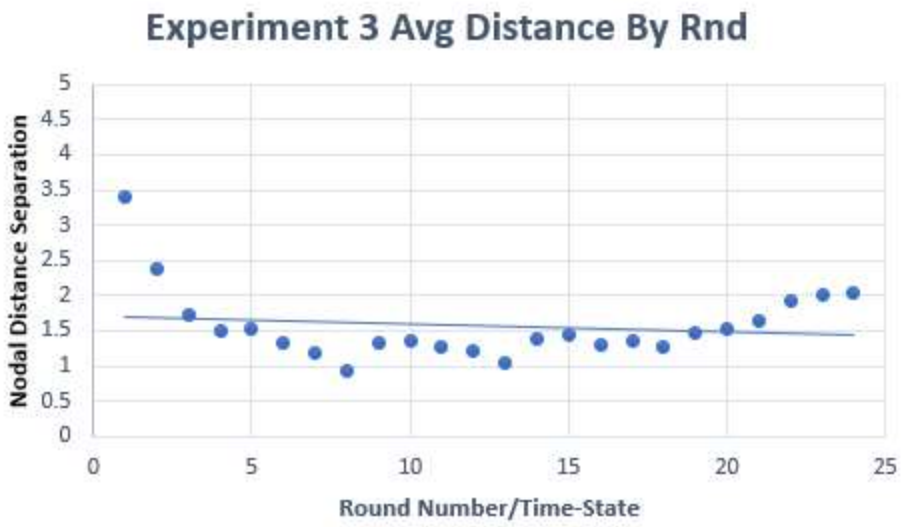

Figure 4.7: Experiment 3 average distance by round

4.4.4 Average Fuel Analysis

Experimental design did not appear to have an impact on average fuel use. Both *evader* and *pursuer* agents were able to balance their fuel between aggression and conservation without issue. This result is not an indication though that diligence can be spared when the agent is migrated to 3D space simulations. The average fuel use by *evaders* and *pursuers* appear to be centered between Experiment 1 which was the lowest among the 3, and Experiment 2, the highest.

4.5 Summary

This chapter analyzed data from each of the three experiments as outlined in Chapter 3. The results show MC agents are effective autonomous players given a limited set of information. The agents employed as described in Chapter 3 showed competent level of play between the three experiments. This chapter analyzed the

experimental design's impact on win ratio, game length, average distance between hider and pursuer, and average ticket use. This chapter then analyzed how the performance metrics effect gameplay at large. Finally, the results presented in this chapter show this MCTS model is capable of handling spaceflight dynamics, while presenting challenges which need to be planned and accounted for.

V. Conclusions

5.1 Chapter Overview

This thesis has created a new platform to test autonomous pursuer-evader algorithms using a simplified two-dimensional environment with some approximated spaceflight dynamics. A MC search algorithm was used as a proof of concept for the platform. The results show promise for further development while also highlighting challenges to be addressed in the future.

5.2 Research Conclusions

To address the hypothesis of a MCTS algorithm as an effective ML model problem for two spacecraft operating in close proximity with imperfect domain knowledge running a pursuit-evasion scenario, the following research questions were posed:

1. How can a MCTS model be used to provide a one-on-one to many-on-many pursuit-evasion framework of proximal spacecrafts?

   This research showed that the model employed by Nijssen/Winands can be expanded to account for spaceflight dynamics to achieve objective. The 3 experiments employed in this research highlighted challenges which must be further explored to ready an autonomous defensive, counter-offensive system, and this research is a foundational step toward achieving this state.

2. How can the MCTS algorithm be modularized to support the varying frameworks between one-on-one and many-on-many scenarios?

The MCTS algorithm can be deployed as an agent within the player or system to act autonomously. This research simulated this effect by giving the resource to the agent to act on their turn.

3. How does the model perform under the following circumstances: one pursuer versus one evader and five pursuers vs one evader?

   This research found that the MC agents were able to act based upon known information. This research also showed that as the state space expands, considerations to prune the iterative tree building process must be planned and accounted for decisions to be made effectively. Experiment 2 held a poor response time requiring sacrifice in the number of simulations that could be performed in this research. Modifying the UCT to prune the width to better approximate movement will help increase responsiveness within the agent to better act in real-time as research progresses to 3D space.

5.3 Significance of Research

This research provides a foundational baseline toward equipping an autonomous defense, counter-offense system for agents operating in space. MCTS is a proven reinforcement learning method for effectively making decisions based upon limited domain information. This research expanded upon an effective agent created for Scotland Yard to account for spaceflight dynamics.[6]

5.4 Recommendations for Action

As this research is a foundational product, it is recommended to expand upon lessons learned during implementation. First, it is recommended any implementation of a MC agent be done on the system itself over a master controller. The MC agent presented in this research simulates separate entities for each hider and pursuer when the program tracks their turn. Should there be a need for a system to defend itself in a hypothetical dogfight, the agent is best suited to function on the system implemented to perform actions real-time.

Next, state space must be truncated to best approximation over defined timeframes for the UCT algorithm to provide decisions in necessary real-time. Experiment 2 showed the need for this truncation as simulations had to be cut to 100 trials to gain results in necessary timeframe. As research expands into 3D space simulations, planning on state truncation is necessary to handle an infinite state traversal from any direction. It is recommended to truncate tree to a maximum width of 10 possibilities of one direction to allow deeper searches before reaching computational limits.

The third recommendation is a bias should be added to UCT algorithm to decide how much fuel is allowable in a set duration. Delta-velocity (ΔV) is a finite resource and care is necessary to sustain the system's mission while simultaneously managing incoming threat or threats. This bias should be applied toward maximum consumption for an immediate time state. The bias also needs a delimiter to manage available ΔV while defending against persistent threats.

Finally, the MCTS model is a modular RL toolset, that can be paired with a Deep Learning Neural Network (DLNN). It is recommended adding an expandable DLNN as more historical TTPs are presented toward hostile pursuit-evasion scenarios are presented in the space domain. This will aid the speed for MC agent's decision making in its UCT algorithm.

5.5 Recommendations for Future Research

As this iteration of MCTS model is a foundational, expansion based upon the MC agents created for Scotland Yard, it is recommended to take the actions presented in Section 5.4 and move toward a 3D space simulator. The MC agent presented with this research showed the capability to handle introductory spaceflight dynamics, however, a true space simulator will test the MCTS model's performance with more realistic scenarios. This research was limited to test full spaceflight dynamics keeping within the Scotland Yard gameboard.

As discussed in Section 5.4, it is recommended to research how the DLNN aids the MC agent's decision-making performance by pairing known recommended TTPs to observations outlined in a certain time state. DLNNs have been paired with MCTS models in parallel avenues of research and can be borrowed toward implementation as research progresses in pursuit-evasion tactics of spacecraft.

5.6 Summary

This concludes the thesis research for the development of a MCTS model designed for rendezvous spaceflight operations. This research began by introducing threats

happening in the space domain and the need toward creating an autonomous defense, counter-offense systems to protect vital space systems as threats increased. Chapter 2 provided background of relevant fields of research necessary to create an autonomous defense, counter-offense system, focusing on AI and ML models in use today combined with game theory to help produce and optimize TTPs for realistic scenarios. Chapter 2 also provided relevant spaceflight dynamics the MC agent would need to handle to successfully traverse between states. Chapter 3 created the framework and design to test the MC agent in three experiments and provided performance metrics to evaluate successfulness of the agents. Chapter 4 presented the results from the three experiments and analyzed the performance metrics under each experiment as well as the performance metric applied across all experiments to evaluate how the metric changed performance. Finally, Chapter 5 addressed the outcomes from this research as well as laid the framework for future work toward the creation of the autonomous defense, counter-offense system.

Bibliography

[1] Coates, G.M., Hopkinson, K.M., Graham, S.R. and Kurkowski, S.H. "A Trust System Architecture for SCADA Network Security." *IEEE Transactions on Power Delivery*, 25(1), pp.158-169, 2009.

[2] Duncan, M.C., Hopkinson, K.M., Trias, E.D. and Humphries, J.W. "Trust Management Approach to Satellite System Telecommanding Security." *Journal of Aerospace Information Systems,* 11(1), pp.19-34, 2014.

[3] Ferreira, P., Paffenroth, R., Wyglinski, A., Hackett, T.M., Bilén, S., Reinhart, R. and Mortensen, D. "Multi-Objective Reinforcement Learning for Cognitive Radio--Based Satellite Communications." *In 34th AIAA International Communications Satellite Systems Conference* (p. 5726), 2016.

[4] Ferreira, P.V.R., Paffenroth, R., Wyglinski, A.M., Hackett, T.M., Bilén, S.G., Reinhart, R.C. and Mortensen, D.J. "Multi-Objective Reinforcement Learning for Cognitive Satellite Communications Using Deep Neural Network Ensembles." *IEEE Journal on Selected Areas in Communications*, 36(5), pp.1030-1041, 2018.

[5] George, B.C. "Optimal and Robust Neural Network Controllers for Proximal Spacecraft Maneuvers." *Air Force Institute of Technology*, Wright-Patterson AFB, OH Graduate School of Engineering and Management, 2019.

[6] Nijssen, P. and Winands, M.H. "Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard." *IEEE Transactions on Computational Intelligence and AI in Games*, 4(4), pp.282-294, 2012.

[7] Curtis, H. “Orbital Mechanics for Engineering Students.” Amsterdam: *Butterworth-Heinemann*, 2005.

[8] Colman, A. “Game Theory and its Applications.” New York, NY, USA: *Routledge*, 2003.

[9] Watson, J. “Strategy, an Introduction to Game Theory.” New York, London. W. *W. Norton & Company*, 2013.

[10] Hamman, S., Hopkinson, K., Markham, R., Chaplik, A., Metzler, G. “Teaching Game Theory to Improve Adversarial Thinking in Cybersecurity Students.” *IEEE Transactions on Education*, Vol 60(3), pp 205-211. 2017.

[11] Turing, A. M. “Computing Machinery and Intelligence.” *Mind, New Series, Vol 59, No. 236*, pp. 433-460, Oxford University on behalf of Mind Assoc., Oct 1950.

[12] Samuel, A. L. “Some Studies in Machine Learning Using the Game of Checkers. II-Recent Progress” *IBM Journal of research and development*, 11(6), pp.601-617, Nov 1967.

[13] Bernstein, A., Arbuckle, T., De V Roberts, M. and Belsky, M.A. “A Chess Playing Program for the IBM 704.” *In Proceedings of the May 6-8, 1958, western joint computer conference: contrasts in computers* (pp. 157-159). ACM, 1958.

[14] Kurenkov, A. “A ‘Brief’ History of Game AI up to AlphaGo.” *Andrey Kurenkov*, 18 Apr 2016, URL: https://www.andreykurenkov.com/writing/ai/a-brief-history-of-game-ai/

[15] Ippolito, P. P. “Game Theory in Artificial Intelligence.” *Towards Data Science*, Sep 2019, URL: https://towardsdatascience.com/game-theory-in-artificial-intelligence-57a7937e1b88

[16] Shannon, C.E. “XXII. Programming a Computer for Playing Chess.” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314), pp.256-275, 1950.

[17] Whaland, N. “A Computer Chess Tutorial.” *In Computer chess compendium* (pp. 221-232). Springer, New York, NY, 1988.

[18] King Jr, D.W. “Complexity, Heuristic, and Search Analysis for the Games of Crossings and Epaminondas.” *Air Force Institute of Technology*, Wright-Patterson AFB, OH Graduate School of Engineering and Management, 2014.

[19] Goodfellow, I., Bengio, Y.and A. Courville, “Deep Learning.” *MIT Press*, 2016.

[20] DeCoste, D., “The Future of Chess-Playing Technologies and the Significance of Kasparov Versus deep Blue. In Deep Blue Versus Kasparov: The Significance for Artificial Intelligence.” *AIAA Technical Report* WS-97-04. (pp. 9-13). 1997

[21] Winands, Mark H.M. “6 x 6 LOA is Solved”. *ICGA Journal*, 234–238, December 2008.

[22] Ciancarini, P., Favini, G., “Monte Carlo Tree Search Techniques in the Game of Kriegspiel.” *In Proceedings of the 21st International Joint Conference on Artificial Intelligence* (IJCAI-09), C. Boutilier, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 474–479.

[23] J. A. M. Nijssen and M. H. M. Winands, “Enhancements for Multi-Player Monte-Carlo Tree Search.” *In Computers and Games* (CG 2010), ser. LNCS, H. J. van den Herik, H. Iida, and A. Plaat, Eds., vol. 6515. Berlin, Germany: Springer-Verlag, 2011, pp. 238–249.

[24] Friedman, D. “On Economic Applications of Evolutionary Game Theory.” *Journal of Evolutionary Economics*, 8(1), pp.15-43, 1998.

[25] Hanley, N. and Folmer, H. “Game Theory and the Environment.” *Edward Elgar*, 1998.

[26] Albino, D., Scaruffi, P., Moretti, S., Coco, S., Truini, M., Di Cristofano, C., Cavazzana, A., Stigliani, S., Bonassi, S. and Tonini, G.P. “Identification of Low Intra-Tumoral Gene Expression Heterogeneity in Neuro-Blastic Tumors by Genome-Wide Expression Analysis and Game Theory.” *Cancer: Interdisciplinary International Journal of the American Cancer Society*, 113(6), pp.1412-1422, 2008.

[27] Min, H. “Artificial Intelligence in Supply Chain Management: Theory and Applications.” *International Journal of Logistics: Research and Applications*, 13(1), pp.13-39, 2010.

[28] Nagurney, A. and Li, D. “A Supply Chain Network Game Theory Model with Product Differentiation, Outsourcing of Production and Distribution, and Quality and Price Competition.” *Annals of Operations Research*, 226(1), pp.479-503, 2015.

[29] Duan, H., Wei, X. and Dong, Z. “Multiple UCAVs Cooperative Air Combat Simulation Platform Based on PSO, ACO, and Game Theory.” *IEEE Aerospace and Electronic Systems Magazine*, 28(11), pp.12-19, 2013.

[30] Cognub, “Cognitive Computing and Machine Learning." http://www.cognub.com/.

[31] Ghahramani, Z. “Unsupervised Learning.” *In Summer School on Machine Learning* (pp. 72-112). Springer, Berlin, Heidelberg, 2003.

[32] Kotsiantis, S.B., Zaharakis, I. and Pintelas, P. “Supervised Machine Learning: A Review of Classification Techniques.” *Emerging Artificial Intelligence Applications in Computer Engineering*, 160, pp.3-24, 2007.

[33] D. Michie, “Game-Playing and Game-Learning Automata.” *Advances in Programming and Non-Numerical Computation*, pp. 183–200, 1966.

[34] Pepels, T., Winands, M.H. and Lanctot, M. “Real-Time Monte Carlo Tree Search in Ms Pac-Man.” *IEEE Transactions on Computational Intelligence and AI in games*, 6(3), pp.245-257. 2014.

[35] N. R. Sturtevant and R. E. Korf, “On Pruning Techniques for Multi-Player Games.” *In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence.* AAAI Press / The MIT Press, 2000, pp. 201–207.

[36] L. Kocsis and C. Szepesv´ari, “Bandit Based Monte-Carlo Planning.” *In Machine Learning*: ECML 2006, ser. LNCS, J. F¨urnkranz, T. Scheffer, and M. Spiliopoulou, Eds., vol. 4212. Berlin, Germany: Springer-Verlag, 2006, pp. 282–293.

[37] Gelly, Sylvain, Levente Kocsis, David Silver, and Csaba Szepesv´ari. “The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions”. *Communications of the ACM*, 55(3):106–113.

[38] Bouzy, B. and B. Helmstetter. “Monte Carlo Go Developments”. Heinz, Herik, and Iida (editors), *10th Advances in Computer Games*, 159–174. Kluwer Academic Publishers, 2003.

[39] Carr, R.W., Cobb, R.G., Pachter, M. and Pierce, S. “Solution of a Pursuit–Evasion Game Using a Near-Optimal Strategy.” *Journal of Guidance, Control, and Dynamics*, 41(4), pp.841-850, 2018.

[40] Li, Z.Y., Zhu, H., Yang, Z. and Luo, Y.Z. “A Dimension-Reduction Solution of Free-Time Differential Games for Spacecraft Pursuit-Evasion.” *Acta Astronautica*, 163, pp.201-210, 2019.

[41] “Scotland Yard | Board Game | BoardGameGeek,” http://www.boardgamegeek.com/boardgame/438/scotland-yard, December 2019.

[42] Winands, Mark H.M. “Analysis and Implementation of Lines of Action.” Master’s thesis, *University of Maastricht*, August 2000.

[43] Winands, Mark H.M. “Informed Search in Complex Games.” Ph.D. thesis, *University of Maastricht*, December 2004.

[44] Winands, Mark H.M., Yngvi Bj¨ornsson, and Jahn-Takeshi Saito. “Monte Carlo Tree Search in Lines of Action”. *IEEE Transactions on Computational Intelligence and AI Games*, 2(4):239–250, December 2010.

[45] R. Bjarnason, A. Fern, and P. Tadepalli, “Lower Bounding Klondike Solitaire with Monte-Carlo Planning.” *In International Conference on Automated Planning and Scheduling/Artificial Intelligence Planning Systems*, A. Gerevini, A. Howe, A. Cesta, and I. Refanidis, Eds., 2009, pp. 26–33.

[46] G. Van den Broeck, K. Driessens, and J. Ramon. “Monte-Carlo Tree Search in Poker using Expected Reward Distributions.” *In Advances in Machine Learning*, ser. LNAI, Z.-H. Zhou and T. Washio, Eds., vol. 5828. Berlin, Germany: Springer-Verlag, 2009, pp. 72–83.

[47] M. Ponsen, G. Gerritsen, and G. M. J.-B. Chaslot. “Integrating Opponent Models with Monte-Carlo Tree Search in Poker.” *In Interactive Decision Theory and Game Theory Workshop at AAAI*, vol. 10, 2010, pp. 37–42.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
15-06-2020	Master's Thesis	September 2018 – June 2020

TITLE AND SUBTITLE
Monte Carlo Tree Search Applied to a Modified Pursuit/Evasion Scotland Yard Game with Rendezvous Spaceflight Operation Applications

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Daugherty, Joshua A., Master Sergeant, USAF

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENG)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENG-MS-20-J-003

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Space has become a warfighting domain. To combat threats, an understanding of tactics, techniques, and procedures must be captured and studied. Games and simulations are effective tools to capture data lacking historical context. Artificial intelligence models use simulations to develop proper defensive and offensive TTPs capable of protecting systems against potential threats. Monte Carlo Tree Search is a bandit-based reinforcement learning model known for using limited domain knowledge to push favorable results. Monte Carlo agents have been used in a multitude of imperfect domain knowledge games. One such game was in which Monte Carlo agents were produced and studied in an imperfect domain game for pursuit-evasion tactics is Scotland Yard. This thesis continues the Monte Carlo agents previously produced by Winands and Nijssen, applied to Scotland Yard. In the research presented here, the rules for Scotland Yard are analyzed and modified to approximate spaceflight dynamics, providing a foundation for use within space environments. Results show promise for the use of Monte-Carlo agents in pursuit/evasion autonomous space scenarios while also illuminating some major challenges for future work in more realistic three-dimensional space environments.

15. SUBJECT TERMS
Artificial Intelligence, Monte Carlo Tree Search, Reinforcement Learning, Pursuit-Evasion

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kenneth M. Hopkinson, AFIT/ENG
U	U	U	UU	78	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4579 (kenneth.hopkinson@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18