A METHODOLOGY FOR TECHNOLOGY-TUNED DECISION BEHAVIOR ALGORITHMS FOR TACTICS EXPLORATION

A Dissertation
Presented to
The Academic Faculty

By

Andrew K. Hull

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

December 2021

A METHODOLOGY FOR TECHNOLOGY-TUNED DECISION BEHAVIOR ALGORITHMS FOR TACTICS EXPLORATION

Approved by:

Dr. Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Michael Steffens
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Scott McEntire
Sandia National Laboratories

Dr. Jennifer Jordan
School of International Affairs
Georgia Institute of Technology

Dr. Daniel Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: August 12, 2021

To my good friends, Tajr and Kristen.

To my running partner, Brigitte.

Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In 2016, the USAF determined that current development and acquisition methods may be inadequate to achieve air superiority in 2030. The airspace is expected to be highly contested by 2030 due to the Anti-Access/Area Denial (A2/AD) strategies being employed by adversaries. Capability gaps must be addressed in order to maintain air superiority. The USAF identified new development and acquisition paradigms as the number one non-material capability development area. The idea of a new development and acquisition paradigm is not new. Such a paradigm shift occurred during the transition from threat-based acquisition during the cold war to capability-based acquisition during the war on terror.

Investigation into current US development and acquisition methods found several notional methodologies. Effectiveness-Based Design (EBD) and Technology Identification, Evaluation, and Selection for Systems-of-Systems (TIES for SoS) have been proposed as notional solutions. Both methodologies seek to evaluate the *means* – the technologies used to perform a mission – and the *ways* – the tactics used to complete a mission – of the technology-tactic design space. Proper evaluation of the *ways* would provide critical information to the decision-maker during technology selection. These findings suggest that a new paradigm focused on effectiveness-based acquisition is needed to improve current development and acquisition methods. To evaluate the *ways* design space, current methods must move away from a fixed or constrained mission model to one that is minimally defined and capable of exploring tactics for each unique technology.

The Technology-tuned Decision Behavior Algorithms for Tactics Exploration (Tech-

DEBATE) methodology enables the exploration of the *ways*, or more formally, the mission action design space. However, the size of the mission action design space is immense and can sharply increase in complexity. To resolve this, the novel decision behavior approach is introduces which discretizes the mission action space into the decision-state space and the state-action space without limiting tactics exploration. Discrete decision behaviors and state behaviors provide a traceable way to record tactics that intelligent agents choose. The agents utilize decision and state behavior algorithms developed using deep reinforcement learning to explore the mission action design space for each technology and learn thier respective optimal tactics. These technology-tuned algorithms reduce reliance on Subject Matter Experts (SMEs) who inherently introduce bias in the development of tactics.

The decision-state space within the mission action space can still have an immense dimensionality. The novel behavior suitability matrix is introduced to reduce the dimensionality of the decision-state space. Proper management of the dimensionality will be required for investigating technologies in multi-agent systems. Extension of the decision behavior algorithms to cooperative multi-agent systems is addressed to ensure cooperative tactics can be captured during technology evaluation. The introduction of cooperative agent behavior increases the complexity of the mission action design space. The choice decentralized agents during the formulation of the problem can enable the identification of agent-independent decision and state behaviors that do not depend on cooperative decisions which reduces the number of algorithms required as the number of agents increases.

The methodology also addresses the synthesis of technology and tactical alternatives. The decision behavior approach defines the steps to develop an agent-based framework to enable the exploration of the *means* and *ways*. The use of a stochastic policy with reinforcement learning enables variation of tactical alternatives for each technology alternative. The novel quantification of the *means* and *ways* frontier provides critical information to the decision-maker during technology investment decisions.

This thesis is divided into five research areas to address the gap in tactics quantifica-

tion for technology evaluation methodologies. Identifying an appropriate agent modeling framework is first investigated. The development of an agent's behavior framework can then be addressed. Once the agent's modeling and behavior framework is formulated, methods to conduct mission action design space exploration will be discussed. The extensibility of the method to cooperative multi-agent systems will then be addressed. The final research area focuses on conditioning the results to enable the comparison of disparate technology-tactic combinations.

In conclusion, the Tech-DEBATE methodology provides a novel method that enables the variation of both technology and tactical alternatives to infrom technology investment during conceptual design. This methodology enables the use of an interactive trade-off environment that allows the decision-maker to explore optimal tactical alternatives for each technology alternative of interest. The novel formulation of technology-tactic algorithms provides insight into effective tactical alternatives to defeat the anti-access area denial strategy as well as the technology investments required to maintain air superiority. The significantly increase in effectiveness-based information available to the decision-maker helps to reduce the risks and costs associated with the development and acquisition of new technologies.

CHAPTER 1
INTRODUCTION & MOTIVATION

1.1 Achieving Air Superiority

> *"For good or ill, air mastery is today the supreme expression of military power, and fleets and armies, however vital and important, must accept subordinate rank." –Winston Churchill [1]*

Since the advent of military aircraft, the strategy of war has dramatically changed. Initially fielded by Italy in 1911, aircraft immediately provided significant advantages to the fielding army – first through reconnaissance and later as an offensive weapon. It was not until WWI that aircraft became a useful, but inaccurate, offensive weapon in battle [2]. As the technology of aircraft and munitions improved, the importance of achieving air superiority became paramount in achieving military objectives. At the operational level of warfare, air superiority enables supply, communication, and movement. Without it, a government is unable to accomplish its main purpose - protecting its citizens and property [3].

> *"From an offensive standpoint, winning strategic air superiority is the number one priority of the commander; once that is accomplished, everything else is just a matter of time." [3]*

Put simply, air superiority "provides freedom from attack, freedom to attack, freedom of action, freedom of access, and freedom of awareness" [4]; however, defining air superiority is not as simple. The United States Air Force (USAF) clarifies the definition of air superiority as a temporary condition over a specified area to enable joint operations. This condition of air superiority varies significantly depending on the environment, mission, or

phase of conflict [4]. Therefore in order to encompass all environments, missions, and phases of conflict, an air force must have a wide range of capabilities including offensive, defensive, and supportive roles. Over the 1990s and 2000s, the US has capitalized on their technological advantages in the air due to innovations in areas such as precision guided munitions [5]. Although the US has had the upper hand in air superiority for many years, it is worth noting that air superiority does not ensure success in a conflict, but merely reduces the risk of joint operation failures and cost of success [1, 4].

The US has decisively won conflicts, specifically the Persian Gulf War, in part due to it's ability to gain air superiority [1, 2, 3, 6]. The new strategies and lessons learned from the Gulf War and many other modern conflicts have had a significant role in future war planning and technology investments. Then Secretary of Defense Richard Cheney said "[The Gulf] war demonstrated dramatically the new possibilities of what has been called the 'military-technological revolution in warfare' " [6]. The insights gained from past conflicts have not only helped shape modern strategy and technology, but also shaped military acquisition. Threat-based acquisition during the cold war produced many technologically superior systems that worked well against a specific threat, but lacked the capability to be effective against others [7]. This was evident during the Gulf war when the CBU-89 scatterable mine - designed for a war against mobile Soviet tank armies - proved much less lethal against entrenched iraqi tanks [6].

In the early 2000s, the war on terror spurred a paradigm shift from threat-based to capabilities-based acquisition. This is seen in the "Transformation Planning Guidance" released by the Department of Defense (DoD) and later the "Transformation Flight Plan" released by the USAF [8, 9]. Capability-based acquisition focused on procuring capabilities instead of systems. These guides outlined the steps each respective department needed to take in order to proactively prepare for future threats. This paradigm shift brought on new methodologies, such as the Capability-Based Technology Evaluation Methodology, that addressed capability-based acquisition for the USAF with the goal of maintaining techno-

logical superiority against future threats [7]. Although this transformation was a step in the right direction, capability gaps still need attention in order to maintain technological superiority moving forward. In 2016, the USAF released their new flight plan aimed to identify and address future threats expected to be encountered in the highly contested air space of 2030 [4]. The flight plan found that current US development programs would be outpaced by adversaries, which would result in "late-to-need" superior warfighting capabilities and ultimately diminish air superiority by 2030. "After 25 years of being the only great power out there, we're returning to a world of great power competition" said then Lt. Gen. Mike Holmes, the USAF deputy chief of staff for strategic plans and requirements [10].

Current technology acquisition methodologies must quantify the expected highly contested air space to ensure current technology investments maintain air superiority and deliver superior warfighting capabilities before their needed. This thesis seeks to augment current technology acquisition methodologies by quantifying the effectiveness of a technology earlier in the design cycle to reduce the risks and costs associated with the development and acquisition of new technologies. Such a methodology will provide Research and Development (R&D) personnel with enhanced decision making data to inform early technology investment. The objective of this thesis is summarized by the Initial Research Objective below.

Initial Research Objective: *Augment current technology acquisition methodologies by quantifying the effectiveness of a technology earlier in the design cycle to inform technology investment.*

1.1.1 Future Threats

As advanced technologies proliferate, defeating an adversary's anti-access/area denial (A2/AD) capability will be critical in achieving air superiority. Anti-access capabilities directly threaten the theater of war – the area where military events and operations are occurring – denying the opponent access to the air, land, or sea. Area denial capabilities protect important military assets by denying the opponent access to the protected area. In the context of the 2030 air space, Brig. Gen. Alex Grynkewich wrote "[The attacking force] cannot attack an adversary's area denial threats because anti-access capabilities prevent them from projecting power into a theater. [The attacking force] cannot attack the anti-access threats because they are heavily protected by area denial capabilities" [11]. The USAF identified A2/AD as an increasingly common strategy that adversaries will use to create the highly contested environment expected in 2030. The Joint Concept for Access and Maneuver in the Global Commons (JAM-GC) doctrine specifically states that the US aims to defeat the adversary's A2/AD strategy used by threatening their command, control, communication, and intelligence (C3I) assets [12]. The results of the USAF's yearlong study into air superiority concluded that a multi-domain family of stand-in and stand-off solutions could succeed against the A2/AD strategy [4]. Stand-in solutions are those that operate in an adversaries A2/AD domain while stand-off solutions are those that are based outside an adversaries A2/AD area with the capability to penetrate that area. Penetrating bombers and standoff strike platforms, illustrated in Figure 1.1, are examples of stand-in and stand-off solutions respectively. The USAF plan identified critical areas to improve stand-in and stand-off capabilities as well as their own A2/AD capabilities by 2030, such as:

- Long-range strike against counterair targets
- Persistent Intelligence, Surveillance, and Reconnaissance (ISR) capabilities
- Penetrating Counterair (PCA) capabilities that maximize trade-offs between range, payload, survivability, lethality, affordability, and supportability [4]

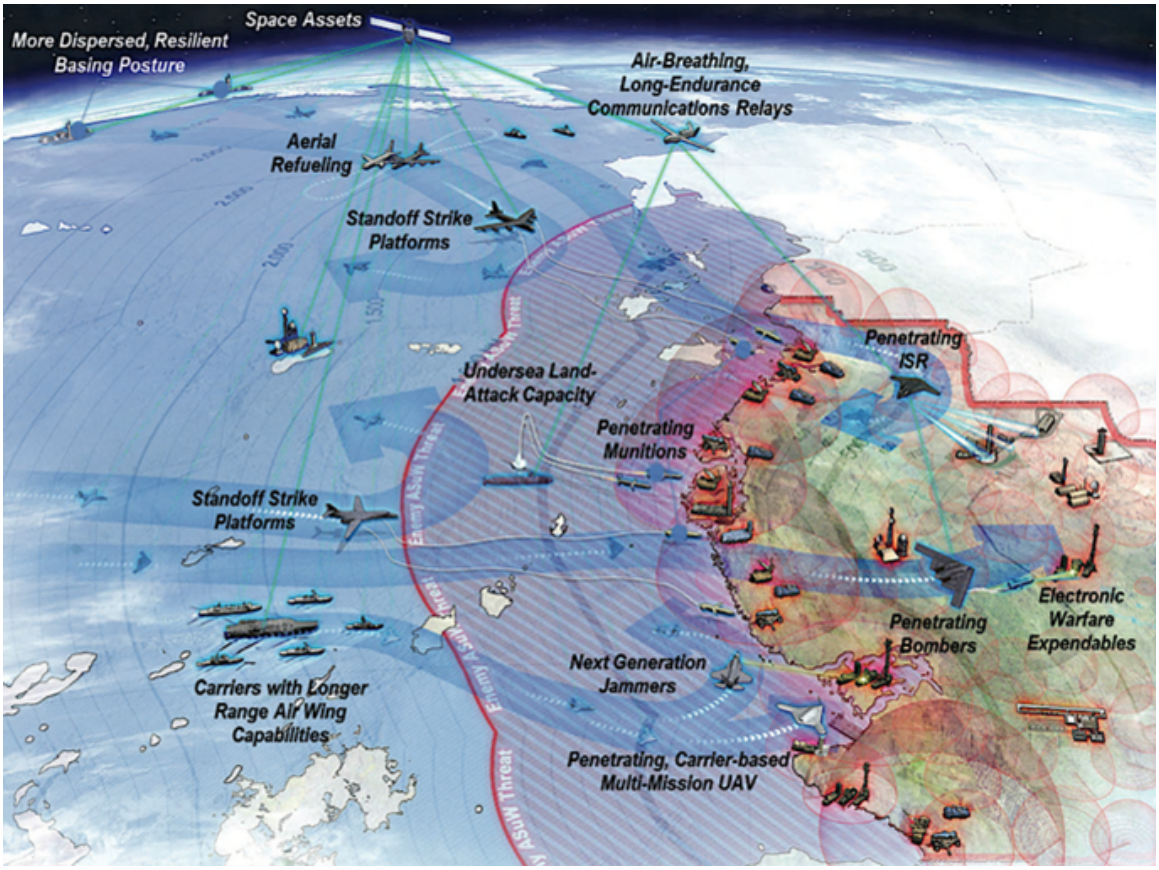

Figure 1.1: Combating the Anti-Access Area Denial Strategy [13].

The Long-Range Strike (LRS), persistent Intelligence, Surveillance, and Reconnaissance (ISR), and Penetrating Counterair (PCA) capability will be critical parts in gaining air superiority in 2030 and beyond. All three examples must have survivability to penetrate contested environments and provide sensor data to enable the employment of stand-in or stand-off weapons. The collection and persistence of ISR capabilities will focus on multi-domain alternatives while also utilizing PCA platforms for data.

Penetrating bombers, such as the B-21, are specifically a key piece of future stand-in capabilities by providing a platform that is survivable and can conduct repeatable offensive operations. Their ability to penetrate A2/AD domains not only contributes to defeating the A2/AD strategy, but also acts as a significant deterrent threat. The penetrating LRS capability is a more potent crisis management tool compared to the legacy bombers and short-range strike platform [14]. The B-2 bomber is a prime example of this potency.

Comparing the 1986 and 2017 Libyan conflicts, the B-2 bomber provided more potency with a much smaller force, significantly reducing the cost of the operation. In 1986, 77 aircraft with two supporting aircraft carrier groups were required for the raid, known as Operation Eldorado Canyon. Alternatively, the 2017 raid deployed only 2 bombers and 15 tankers to complete the mission [15]. The combination of stealth, range, precision, and payload of the B-2 enabled this leap in air superiority. Stealth capabilities of penetrating bombers provide the US with a formidable deterrent from surprise attack while mitigating – but not eliminating – the threat of a US surprise attack [14]. Although the penetrating LRS capability will be critical in achieving air superiority, it is important to note that no single system or capability is expected to provide a "silver bullet" solution [4].

Even though the next generation LRS platform is already in development, US development programs will still need to focus on the maturation and development technologies to enhance the LRS platform for contested environments. The USAF believes hypersonic weapons could enhance stand-in capabilities as an air-launched weapon or provide new stand-off capabilities with the development of boost-glide weapons. Although the emergence of hypersonic technology in combat is not yet known, the US will need to be prepared to defend against such a threat as well. Current US investment is focused on developing hypersonic boost-glide capabilities while also investing in air-breathing hypersonic technologies [16]. Air-breathing hypersonic technology would be especially effective when combined with the B-21 LRS platform. In particular, hypersonic cruise missiles could magnify the lethality of the B-21, further aiding in crisis management [14]. Though air-breathing technology may not emerge before 2030, the USAF recognizes the potential strategic advantage it can provide beyond 2030.

The maturation and development of PCA and ISR platforms is also needed to succeed in the expected highly contested environment. Many ISR platforms, such as those in Table 1.1, experienced unprecedented success in Iraq and Afghanistan in the 2000s. Such success is not likely to carry over into the highly contested airspace anticipated in 2030.

Table 1.1: Examples of US drone platforms adapted from [17].

Weight Class	**Drone**
Light	RQ-20 Puma
Light	RQ-11 Raven
Light	ScanEagle
Light	Switchblade
Medium	RQ-7 Shadow
Heavy	MQ-1C Grey Eagle
Heavy	MQ-1 Predator
Heavy	MQ-9 Reaper
Heavy	RQ-4 Global Hawk
Heavy	RQ-170 Sentinel

Larger platforms, such as the MQ-1 Predator, MQ-9 Reaper, RQ-4 Global Hawk, RQ-170 Sentinel, and the legacy U-2, may not be able to survive in contested environments [18]. The key to enable ISR in contested airspace is to develop survivable platforms that can carry over all the successes experienced from the previous platforms. New Unmanned Aerial Vehicles (UAVs) will be required to loiter, survive, and attack in an A2/AD environment. Some believe that the US may be losing its lead on some technologies [19], whether true or not, only contributes to the need to accelerate acquisition of such technologies.

In order to achieve air superiority by 2030 and beyond, the US must adopt or create new methodologies that reduce design and development investment risks earlier in the acquisition lifecycle to design and field stand-in and stand-off weapons. This focus could provide the US with the necessary tools to defeat A2/AD strategies as well as fortify their own. Enabling platforms and technologies identified that could achieve air superiority while contributing to crisis management include penetrating LRS platforms, PCA capabilities, and survivable ISR. While all these technologies are of interest to achieve air superiority, the focus of this thesis will investigate ISR in a contested reconnaissance environment.

1.1.2 The Acquisition Paradigm

Developing technologies to defeat the A2/AD strategy could enable air superiority; however, the development cycle time to field new capabilities and technologies is just as important. The non-material aspect of capability development is an area of opportunity in the acquisition lifecycle to reduce risks, investment costs, and time to operational capability. Reduced cycle times could prevent delivering "late-to-need" capabilities to the warfighter. The USAF identified five capability development areas to ensure air superiority in 2030. These areas were defined as Basing and Logistics; Find, Fix, Track and Assess; Target and Engage; Command and Control; and Non-Material (DOTMLPF-P). Specifically, the non-material capability development area identified six efforts to focus on, listed below.

1. New development and acquisition paradigms
2. Cyber-based capabilities
3. Increased contributions from space-based assets
4. Invest in foundational infrastructure
5. Continue to pursue "game-changing" technologies
6. Low Cost Systems [4]

The idea of a new acquisition paradigm is not new. Some examples include the push for threat-based acquisition during the cold war, the shift to capability-based acquisition during the war on terror, and now a new push to achieve air superiority in highly contested environments. The USAF states that a new acquisition paradigm must "enable the maturation, demonstration, and integration of advanced technologies into weapon systems on timelines that match the tempo of key underlying technology development cycles" [4]. Development and acquisition programs must also be capable of integrating incremental technology improvements on a regular basis after the initial capability is deployed. The

second and third efforts are critical capability development areas as well. The third effort is especially important in the development and acquisition process. Space-based assets can provide significant advantages to multi-domain forces that need to be captured in the development process. The fourth effort states that improved modeling, simulation, and analysis infrastructure is needed to enable "the rapid assessment of advanced technologies and concepts" [4]. The USAF specifically states that a unified operational environment across the government, industry, and academia as well as all classification levels would be critical for new development. The fourth and fifth efforts are straight forward, but important in enabling air superiority. Air superiority will not only require new technologies, but it will also require the continual pursuit of reducing development costs to field new capabilities.

The non-material capability development area highlights the need for a new paradigm shift in acquisition, similar to the paradigm shift from threat-based to capabilities-based acquisition [8, 9]. However, future acquisition paradigms must bring "agility to multi-domain acquisition" [10] as well as enable the necessary trade-offs to achieve air superiority in the future battlespace. Additional efforts that compliment the acquisition process listed in the other capability development areas include:

- "Mitigate Attack. Capability development for mitigating attacks will include development of active and passive defensive capabilities against ballistic missiles, cruise missiles, and hypersonic weapons...,"
- "Penetrating Counterair (PCA). Capability development efforts for PCA will focus on maximizing trade-offs between range, payload, survivability, lethality, affordability, and supportability," and
- "Weapons. Capability development in this area should focus on leveraging opportunities to create tradespace between platforms, sensors, and weapons... Both long-range and high capacity weapons will enhance the overall effectiveness of the AS 2030 family of capabilities" [4].

The efforts listed above demand the need for a new development and acquisition paradigm that can rapidly investigate concepts, trade-offs, and multi-domain solutions of both stand-in and stand-off capabilities. In the context of the USAF, a multi-domain perspective considers solutions across all domains including surface, air, space, cyber, and the electronic environment [20]. At the conceptual design level, a multi-domain approach could produce a non-aerial solution or provide an aerial solution that incorporates sensor information and/or capabilities from the other domains into the design environment. Modeling multi-domain capabilities in an advanced Modeling and Simulation (M&S) environment could provide the decision maker with more information to better understand which technologies and capabilities should be invested in. In turn, the overall development and acquisition of advanced technologies could be expedited by new conceptual design methodologies. This observation is stated below.

Observation: *Current development and acquisition methods need to be examined and improved upon with the goal to shorten the development and acquisition lifecycle of multi-domain advanced military technologies.*

1.2 Document Overview

Chapter 2 conducts an extensive literature review to identify development and acquisition methods. Specifically, this chapter investigates technology evaluation methodologies and their interplay with operation analysis. The effective combination of technology and tactic evaluation may provide improvements to current development and acquisition methods.

Chapter 3 addresses the gaps in current technology evaluation methods by presenting the research questions that concern this thesis. The discussion builds on research from

Chapter 2 by conducting targeted literature reviews to formulate a set of hypotheses to answer their respective research question.

Chapter 4 will outline the proposed methodology for conducting tactics exploration to aid in addressing the overarching research objective. Each step of the proposed methodology is presented and discussed. An overarching research hypothesis is presented based on the proposed methodology.

Chapter 5 will outline the experiments required to justify the methods chosen for the methodology. Research conclusions are presented after each experiment.

Chapter 6 will present the results of the technology evaluation study. The *means* and *ways* frontier is discussed along with an in-depth analysis of results.

Chapter 7 concludes the research proposed in the thesis. This chapter will summarize the proposed methodology and required experimental efforts to draw research conclusions for the defense of the proposed hypotheses.

CHAPTER 2
BACKGROUND

The initial observations in chapter 1 identified new adversarial strategies and increasingly competitive technology development that could threaten air superiority. Those observations identified a practical gap in current development and acquisition methodologies that could produce "late-to-need" capabilities. New solutions to close that gap must use an agile acquisition process to accelerate the development and acquisition of practical and affordable technologies that will provide real capabilities to the warfighter.

Previous development and acquisition methods have historically been chosen by weighing threat-based or capability-based criteria. Section 1.1 briefly discussed the transition from threat-based to capability-based acquisition methods after the cold war. This transition was made formal through the Joint Capabilities Integration and Development System (JCIDS) in 2003. Quantifying the capability of a design required some level of OA paired with a vehicle design environment. The push for capability-based acquisition aimed to increase the design knowledge and design freedom early in the development process to produce more capable designs. In most cases, increasing the design knowledge and design freedom available early on in the design process can substantially affect the cost of the overall life-cycle of a design. This interaction between design knowledge, design freedom, and cost committed is illustrated in Figure 2.1.

An opportunity may exist to accelerate the design and development cycle time if technology evaluation methodologies are further investigated. The following section reviews current methodologies for technology evaluation and a review of modeling and simulation methods for quantifying mission capabilities.

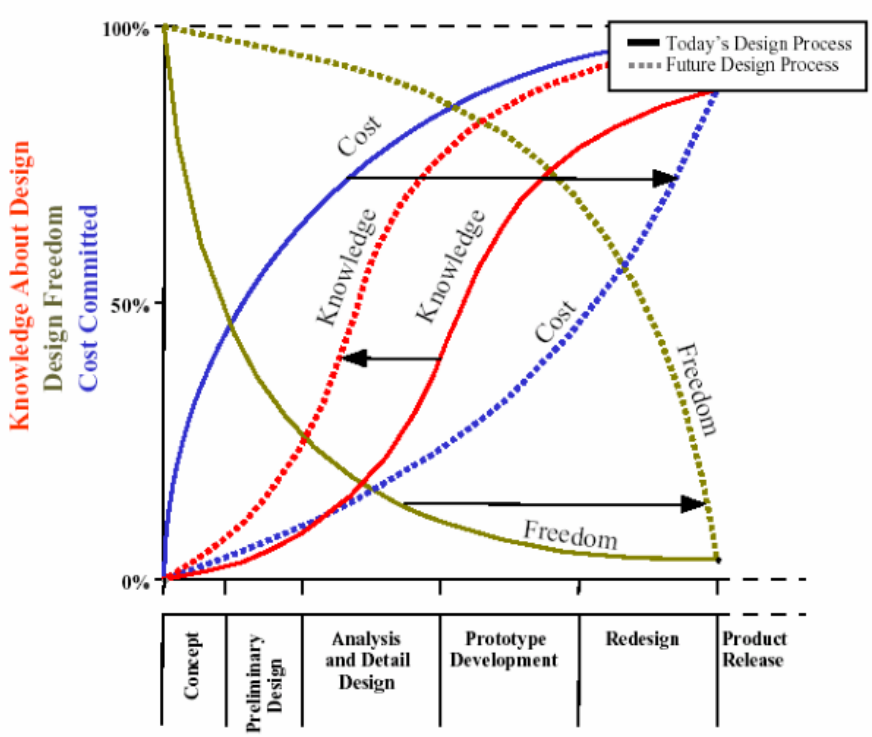

Figure 2.1: Life-Cycle Design Stages [21].

2.1 Design Methodologies for Technology Evaluation

The past few decades of design have focused on enabling the design of both evolutionary and revolutionary technologies. Evolutionary technologies are those that provide incremental improvements to existing technologies while revolutionary ones introduce new variables and solutions that are well outside the limits of current solutions [22]. Revolutionary technologies are not science fiction, but merely concepts that do not have historical data to rely on. These technologies can require complex, physics-based analyses to predict their capability, reliability, affordability, and performance [23]. Achieving air superiority in 2030 and beyond will require some degree of revolutionary technologies that need to be evaluated as a system and its place in a family of systems. New design methodologies will require the use of physics-based analyses to model the revolutionary technologies at the system level while also extending those analyses to the Operations Analysis (OA) environment.

New design methodologies are not developed out of thin air. Just like with any re-

search objective, previous work or methodologies help guide the research and hypothesis to be investigated. Several design methodologies exist that aim to conduct technology evaluation. Some key aspects of design methodologies that were sought included advanced design methods, system-of-systems analysis, and integration of an OA simulation. Biltgen's Dissertation conducted a review of technology evaluation methods which included experimental approach, seminar war games, scientific advisory board, technology development approach, technology performance risk index, technology identification, evaluation, and selection (TIES), and quantitative technology assessment (QTA) [7]. His research concluded that there are nine criteria that are key attributes of technology evaluation methods. These criteria are defined as:

- Quantitative: Measurable process for enumerating ways and means and comparing solutions
- Traceable: Enables identification of the effectiveness drivers of a proposed technology solution
- Flexible: Generalizable to multiple problems in the same class with minimal modification
- Reusable: Method and environment can be used to study multiple attributes of the same problem
- Rapid: Can be applied in a reasonable time frame without unrealistic resource requirements
- Parametric: Avoids point solutions and provides visibility into behaviors previously obscured by the complexity of the problem
- Scalable (to Systems-of-Systems): Avoids simplistic representations of interactions between systems
- Affordable: Produces valid results without extensive manpower commitments and uses commercial off-the-shelf tools when possible
- Simple: The steps in the methodology are reasonable, logical and teachable [7]

Biltgen identified TIES and QTA as best-in-class methods to review further. Those two methodologies along with Biltgen's capability-based technology evaluation method and a more recent value-based design process are the current state-of-the-art technology evaluation methodologies. All four methodologies have unique strengths that must be considered when choosing a methodology for technology evaluation. More recently, two new methodologies – EBD and TIES for SoS – have emerged as potential contenders as state-of-the-art

technology evaluation methodologies and should also be considered as possible methods. Overall, six methodologies stood out from literature due to their proven track record for technology evaluation or incorporation of mission effectiveness. Each methodology will be investigated further to better understand their impact on the development and acquisition process. These methodologies are listed below.

- Technology Identification, Evaluation, and Selection (TIES)
- Quantitative Technology Assessment (QTA)
- Capability-Based Design for Systems-of-Systems (CBD for SoS)
- Value-Based Design (VBD)
- Effectiveness-Based Design (EBD)
- Technology Identification, Evaluation, and Selection for Systems-of-Systems (TIES for SoS)

2.1.1 Technology Identification, Evaluation, and Selection

The TIES methodology was developed as a "comprehensive, structured, and robust methodology for decision making in the early phases of aircraft design" [24]. Kirby's methodology also emphasized the importance of meeting aggressive customer requirements for complex systems. The eight step methodology performed well in its ability to identify technologies to meet the customer's performance and cost requirements by identifying technically feasible and economically viable design space alternatives. Figure 2.2 outlines the methodology's eight steps, which include:

This methodology was considered the best-in-class for a flexible, reusable, and parametric methodology according to Biltgen. He also ranked TIES as 'very good' in the quantitative and traceable category and 'good' as a rapid, scalable to system-of-systems, affordable, and simple methodology [7]. Kirby succeeded by utilizing a physics-based modeling

1. Define the Problem
2. Define Concept Space
3. Modeling and Simulation
4. Investigate Design Space
5. Feasible or Viable
6. Identify Technologies
7. Evaluate Technologies
8. Select Technologies [25]

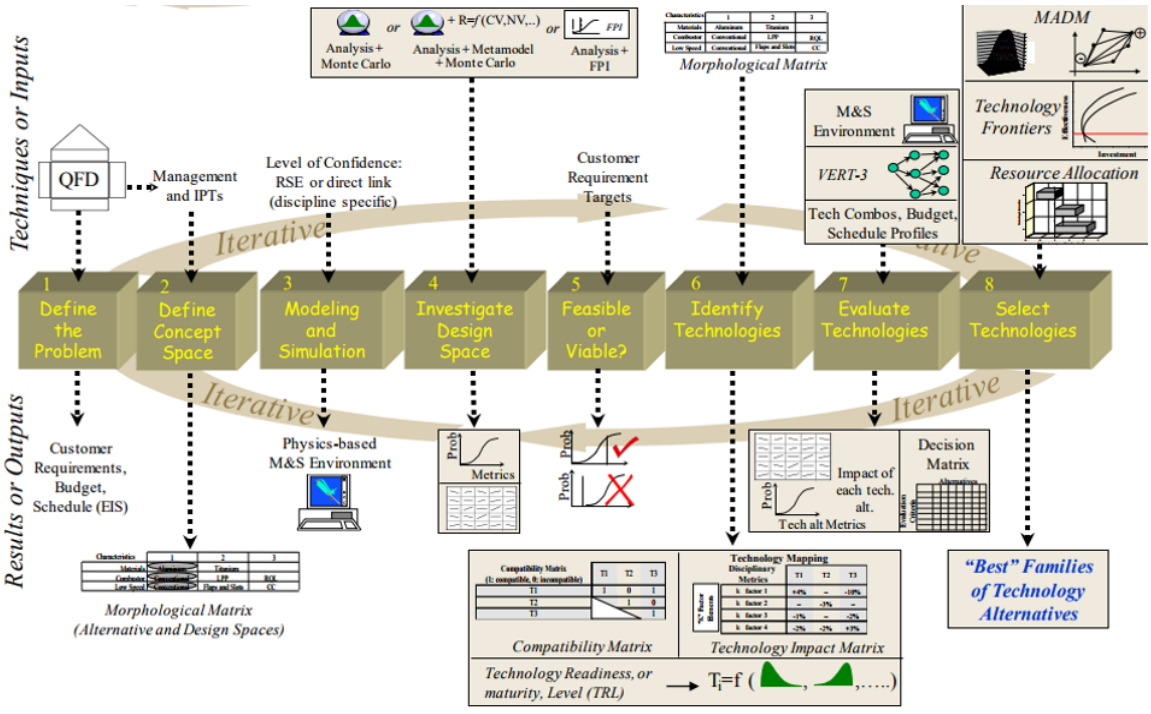

Figure 2.2: Technology Identification, Evaluation, and Selection Methodology [25].

and simulation environment to quantify the impact of revolutionary technologies of interest while also using advanced design methods to identify their impacts on the design space and quantify the uncertainty of their performance. Technology evaluation was conducted by the use of "k-factors" which help assess the impact of a technology on performance metrics. Physics-based disciplinary analysis and accounting for uncertainty in design are two methods that can increase the design knowledge and design freedom while reducing cost committed according to Mavris, who also contributed in the development of TIES [24, 21]. Many aspects of TIES are also effective in enabling the evaluation of technology in a system; however, the methodology lacks a way to evaluate mission effectiveness of the system or its part in a family of systems.

Overall, the Technology Identification, Evaluation, and Selection methodology is useful to consider in developing an effectiveness-based design methodology. Although no mission effectiveness metrics are quantified, the structured methodology has many useful aspects that could be utilized or altered in a new methodology.

2.1.2 Quantitative Technology Assessment

During the push for capability-based acquisition in the early 2000s by the DoD and USAF, AFRL implemented a program called Quantitative Technology Assessment (QTA). This program sought to support the Capability-Focused Technology Investment paradigm within AFRL at the time. QTA supported that paradigm by implementing a methodology which focused on increasing available information on a new technology before the decision maker continued funding the development process. The new capability-focused approach required that "the Air Force be able to quantify the impacts of any proposed technology program on each key capability" [26].

The QTA methodology focused on integrating new Modeling and Simulation (M&S) tools and existing "industry-standard" tools to create a multidisciplinary design environment that could enable capability-focused technology evaluation. The resulting framework,

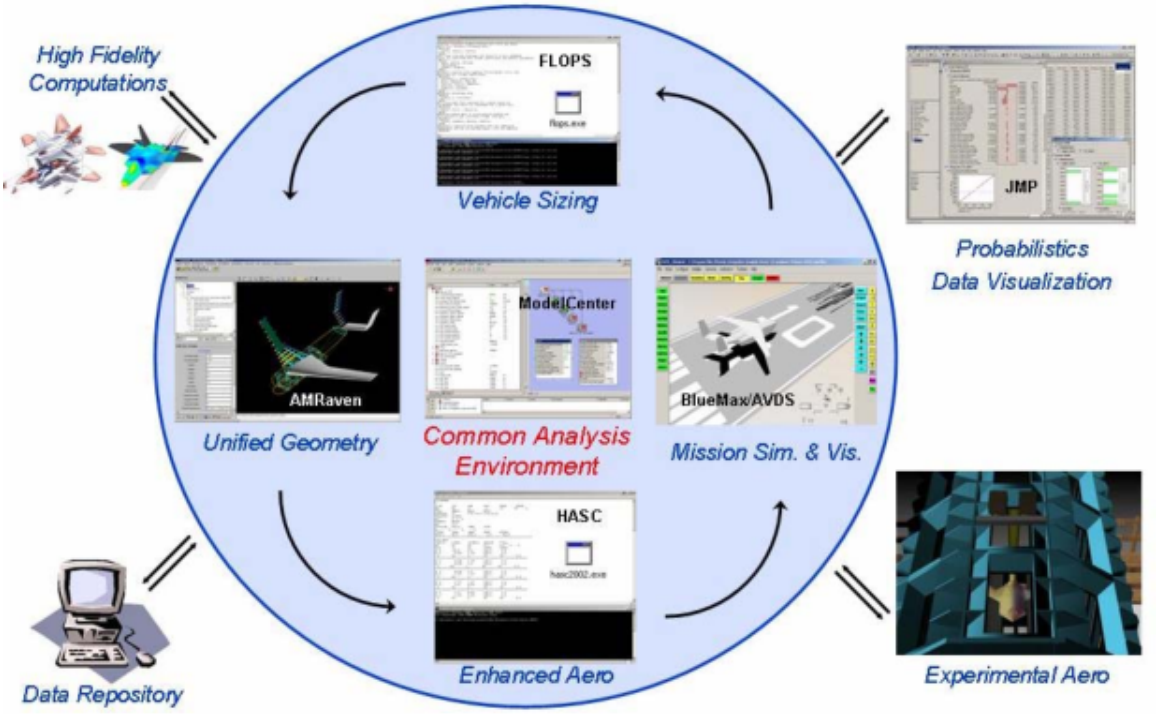

Figure 2.3: Quantitative Technology Assessment Environment [26].

illustrated in Figure 2.3, utilized advanced design methods including probabilistic analysis, parametric modeling, and mission simulations all wrapped in a common analysis environment [26]. The mission simulations specifically sought to "perform sensitivity-type studies of the operational effectiveness of concepts and technologies" [27].

Similar to the TIES methodology, QTA was also considered best-in-class according to Biltgen. QTA was rated to be excellent as a quantitative, traceable methodology, which is scalable as a systems of systems methodology [7]; however, the methodology did not fair well as a rapid and affordable technology evaluation method. Many aspects of the Quantitative Technology Assessment methodology will certainly be useful to study in the development of an effectiveness-based design environment.

2.1.3 Capability-Based Design

Biltgen developed "A Methodology for Capability-Based Technology Evaluation for Systems-of-Systems" [7] which succeeded in modeling both vehicle design and OA. His ten step

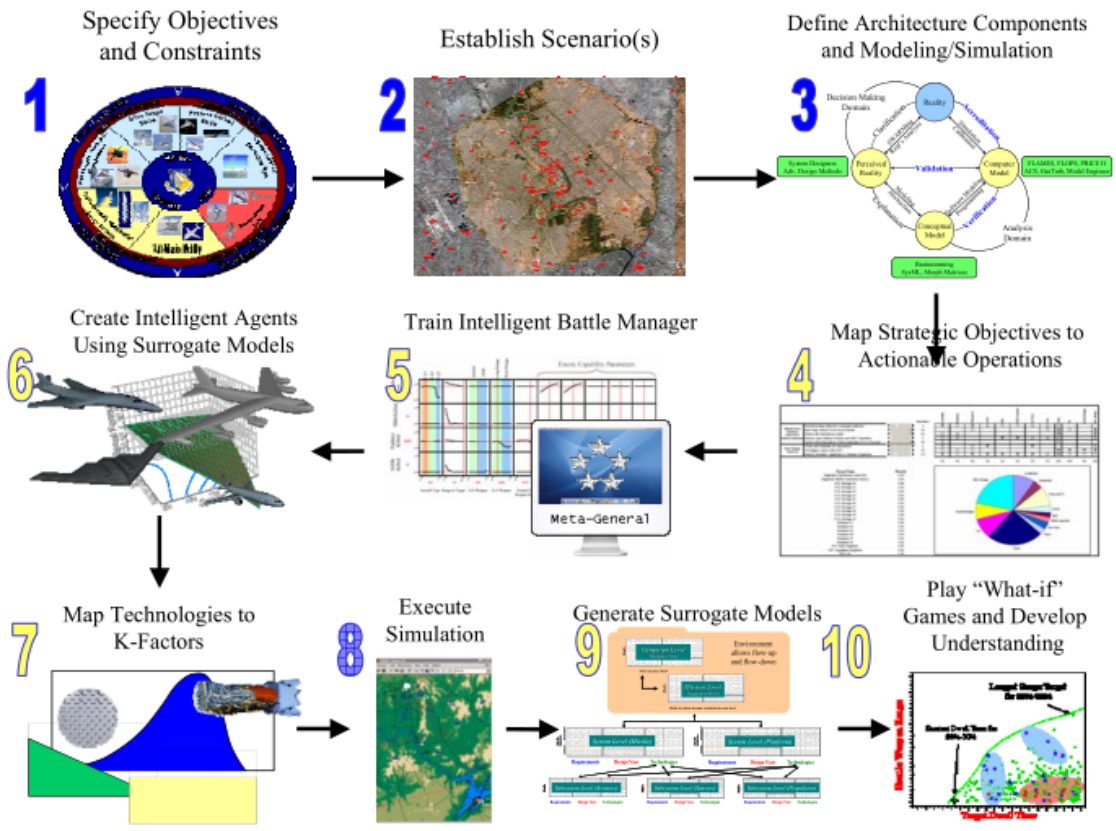

Figure 2.4: Capability-based design methodology for technology evaluation [7].

methodology, illustrated in Figure 2.4, used physics and cognition models to capture the behavior of the real system [7]. During his formulation, both TIES and QTA were studied as partial solutions to the systems-of-systems focused methodology he developed. Biltgen aimed to close the gap in technology evaluation for large-scale heterogeneous systems. His ten step process used familiar techniques such as a multidisciplinary M&S environment, surrogate models, technology "k-factors", and quantifying the capability of concepts and technologies though mission simulations.

The use of a framework that can take advantage of probabilistic methods for uncertainty quantification (UQ) was also emphasized. Biltgen suggests that UQ can be used in step 9 of his methodology due to the computational speed of surrogate models. He later implements probabilistic methods for UQ using his proposed methodology in [28]. This capability-based design methodology utilized both physics-based models and UQ to im-

prove the amount of design knowledge and design freedom available to the decision maker.

Although this methodology contributed to capabilities-based acquisition, it had several limitations due to complexities in modeling systems-of-systems. Biltgen wrote that "the challenge of balancing fidelity and scope is one of the fundamental problems of systems-of-systems design" [7]. As systems become highly integrated, the need for increased fidelity is a requirement due to the interdependencies of functions and disciplines. This ten step methodology is certainly a great starting point for future designers, but may not be able to completely close the gaps in current US development programs.

2.1.4 Value-Based Design

Value-driven design (VDD) is a framework for developing a design environment focused on three main objectives: 1) enabling design optimization in conceptual design for the entire system as a whole; 2) avoiding 'dead-loss' trades by implicitly capturing component conflicts; 3) by not assigning requirements at the component level to prevent increased costs and reduced performance from overly restrictive requirements [29]. The second objective speaks of 'dead-loss' design trades which occurs when design teams each work to optimize their component while unintentionally degrading the performance of the overall system. Gorissen implements this framework as a value-based design method which focuses on "systematic and simultaneous evaluation of cost and benefits of design alternatives" [30]. The value-based design methodology sought to satisfy the objectives of a VDD framework in several ways by developing the Decision Environment for Complex Designs (DECODE). The design environment utilized multidisciplinary M&S modules to optimize the system as a whole, which are illustrated in Figure 2.5. The DECODE modules were linked to allow for trade-offs throughout the design process. Gorissen noted that "despite all the advances that have been made in computational analysis, the fundamental hallmark of design is not analysis but synthesis" [30]. The overall methodology also allowed for system constraints to be flexible in order to find more valuable solutions in the design space.

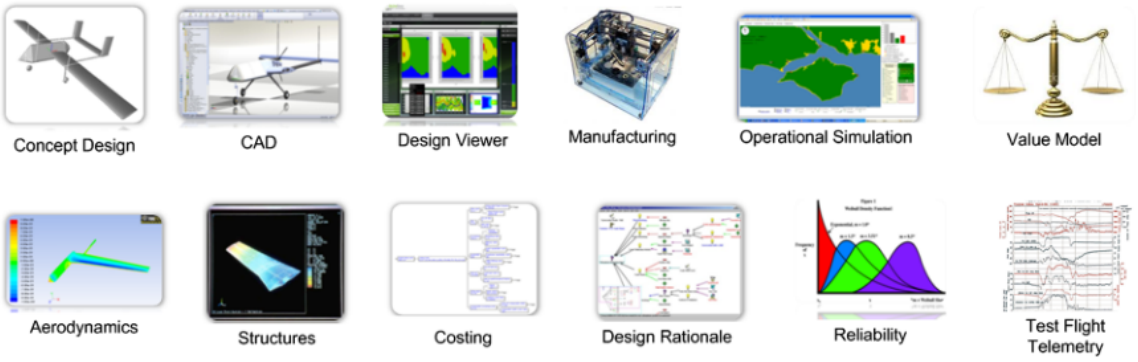

Figure 2.5: Decision Environment for Complex Design [30].

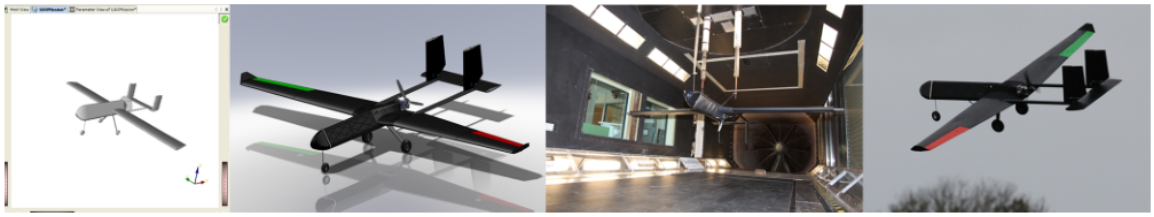

Figure 2.6: Year-long project using DECODE for rapid prototyping [30].

One element of note in the DECODE project are the operational simulation module. Gorissen's implementation of value-based design utilized an agent-based discrete-event simulation for the operational simulation module. He also utilized probabilistic analysis and intends to implement surrogate models to quantify uncertainty and reduce computation time in optimization scenarios. This specific implementation of value-based design succeeded in supporting a rapid prototyping program to design an Unmanned Aerial Vehicle (UAV) for a search and rescue mission. However, the year long project could not thoroughly investigate the design space due to the computational expense of iterating the high-fidelity, complex modules. Gorissen's project limited design space exploration in order to develop this concept in a timely manner.

2.1.5 Effectiveness-Based Design

The Air Force Research Lab (AFRL) has recently been seeking to close several gaps in the development and acquisition process by focusing on advanced conceptual design methods. After much research by AFRL and industry into the Efficient Supersonic Air Vehicle

Exploration (ESAVE) program [31, 32], the focus of research began to shift towards an emphasis on OA methodologies. The OPTimized Integrated MUltidisciplinary Systems program (OPTIMUS) conceived in 2016 by AFRL was developed to address design challenges in early phases of design [33]. This program emphasized mission effectiveness and defined Effectiveness-Based Design (EBD). The program formally defines EBD as "the process of finding an optimal integrated system solution that accomplishes the purpose for the existence of the system" [34]. The OPTIMUS program outlined four objectives to solve current challenges which included:

1. Expansion of a multidisciplinary optimization design process
2. Integration of operations analysis to allow mission effectiveness measures to drive the mission/vehicle design
3. Demonstration of real time geographically distributed architecture to virtually connect different design groups
4. Application of the multidisciplinary design optimization process developed in the first three objectives to an air superiority aircraft suitable for application in a 2030 A2/AD operational environment [33]

AFRL continued development of conceptual design methodologies by leveraging work from the OPTIMUS program through the EXPanded Multidisciplinary Design Optimization for Effectiveness based DesIgn TEchnologies (EXPEDITE) program [35]. This program further emphasizes the coupling of the vehicle and mission design environments through formal development of EBD as a methodology. This *proposed* methodology seeks to increase the design knowledge and design freedom while reducing cost committed by better defining the mission effectiveness of a technology in the conceptual design process.

Mission performance is considered one of seven common disciplines used in Multidisciplinary Design, Analysis, and Optimization (MDAO) of aerial vehicles. However, almost half of the studies surveyed in literature did not include mission performance [36]. Mission performance is analogous to mission effectiveness, but encompasses both non-military and military missions. The other common disciplines in MDAO included aerodynamics, weight estimation, structural analysis, propulsion, geometry, and stability and trim, with mission

performance ranking as the least common out of the top seven disciplines. Although this review was conducted for all aerial vehicles, the report included several studies related to military aerial vehicles. Taking a closer look at the studies that included mission performance, 62% utilized low-fidelity empirical equations to estimate mission effectiveness [36].

The objectives of the EXPEDITE program are similar to those stated for the OPTIMUS program; however, EBD specifically seeks to augment both MDAO and OA methodologies to improve technology evaluation by [34]:

- Numerically coupling physics-based models from the MDAO M&S environment and the OA M&S environment
- Developing path dependent models that can simulate an entire mission in succession using physics-based models and constraints
- Creating a SoS solution space that will remain completely open without any predetermined elements

In order to conduct EBD, the medium- to high-fidelity mission performance modeling used during preliminary and detailed design phases are required for conceptual design [36]. Such modeling methods include discretized numerical solutions or specialized software for mission performance. Higher fidelity mission performance modeling allows for proper coupling of data from physics-based models used in MDAO.

Clark and Allison most recently implemented EBD for a generic fighter aircraft that modeled the mission under uncertain conditions [37, 38]. The study succeeded in not only coupling the physics-based models used in the MDAO process with the OA, but also by modeling a path dependent mission; however, the computational expense of the M&S environment used to model uncertainty for a single mission was high and the use of an open SoS solution space was not investigated. In order to mature the EBD methodology, some research into OA simulations may still be required.

2.1.6 Technology Identification, Evaluation, and Selection for Systems-of-Systems

Kirby's original TIES methodology had many successes as a robust technology evaluation method for commercial aircraft [25]. However, the TIES methodology looses its advantages for the design of military systems due to the lack of an OA framework. The *proposed* TIES for SoS methodology builds off of TIES by incorporating a SoS OA environment [39]. This methodology seeks to quantify the *means* and *ways* of the system to open the design space further when compared to just a *means* perspective.

The *means* and *ways* are typically used to discuss strategic and operational concepts that lead to the *ends*. From this perspective, the *means*, *ways*, and *ends* are concerned with the military resources, strategic concepts, and objectives of a mission, respectively [40]. These concepts can be extended to technology evaluation methodologies to isolate the impact a technology has on the vehicles performance and effectiveness. From the technology evaluation perspective, the *means* is defined as "the impact of individual vehicle technologies on Measures of Performance (MoPs)" while the *ways* is defined as "the impact of complex interactions among multiple elements on mission-level Measures of Effectiveness (MoEs)" [39]. TIES for SoS proposes an expansion of the sixth and seventh steps by adding in steps for tactics identification and evaluation, illustrated in Figure 2.7.

The TIES methodology utilized MoPs to evaluate technologies and identify a Pareto Frontier of optimal solutions. Comparatively, the TIES for SoS methodology seeks to identify a new set of optimal solutions through the exploration of both the means and ways design space, utilizing both MoPs and MoEs. The combined improvement from a means and ways design space exploration could produce a new Pareto Frontier, illustrated in Figure 2.8.

This *notional* methodology could open the design space without additional technologies and reveal feasible solutions where none existed previously. However, the improvement of both MoPs and MoEs requires the ability to identify and evaluate tactics. Much research is still needed to characterize the tactics trade space, quantify the tactics, and develop a

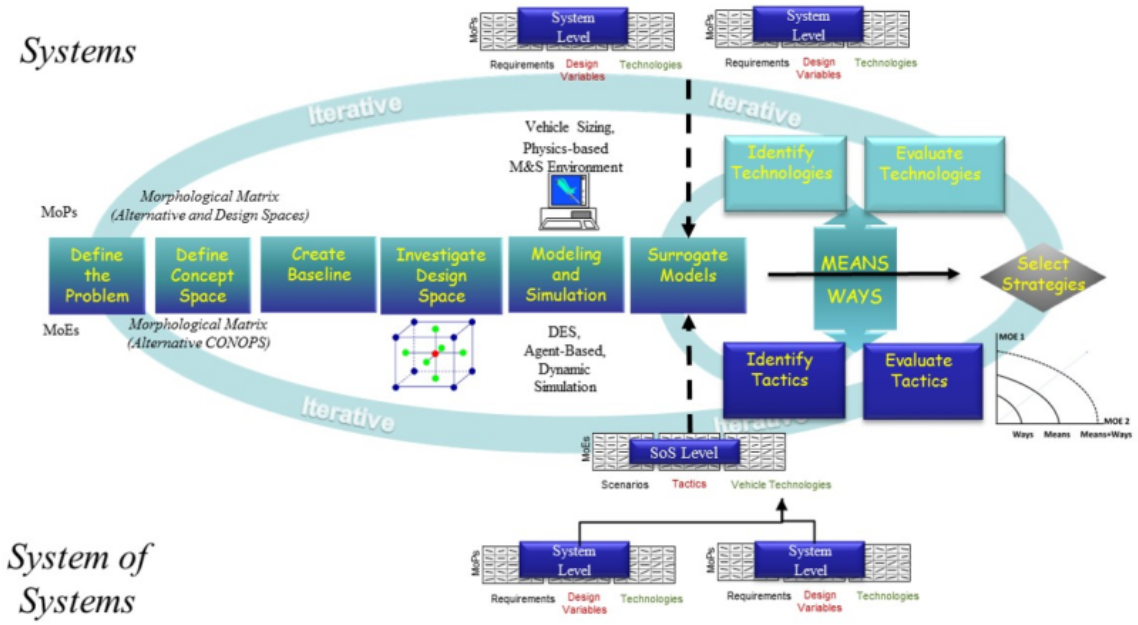

Figure 2.7: Formulation of TIES for SoS [39].

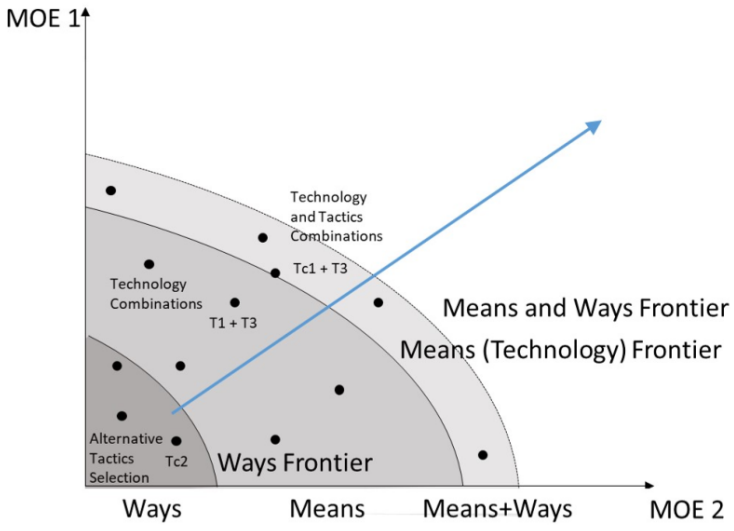

Figure 2.8: Means and Ways trade space [39].

dynamic OA environment that can explore the SoS solution space.

2.1.7 Summary of Methodologies

The research and development of new technology evaluation methods are all motivated by the increase of design knowledge during the early phases of design. The increase of knowledge presented to the decision maker – whether its performance, capability, effectiveness, or focused on a different metric – would enable more informed decisions. Either way, a more informed decision on which technologies to invest in or how to better utilize existing technologies could reduce the cost and time of the development and acquisition process.

The six technology evaluation methods that were reviewed all have advantages that may depend on the type of system under consideration, such as a civil versus a military aircraft. The eight criteria identified by Biltgen, referenced in Section 2.1, are used to compare the previously summarized technology evaluation methodologies for the design of advanced military systems. Biltgen ranked TIES, QTA, and his own methodology, CBD for SoS, in 2007; however, the values of merit for those methodologies from his rankings may not be the same today due to the inherit improvement in the state-of-the-art. The six methodologies – TIES, QTA, CBD for SoS, VBD, EBD, and TIES for SoS – are comparatively ranked by the author in Figure 2.9.

All six methodologies ranked as best-in-class for at least one of the eight criteria, which is denoted by a green box. EBD and TIES for SoS received best-in-class marks as overall methodologies; however, it is worth noting that both are *notional* methodologies and require further research to properly implement. TIES for SoS would have ranked as the state-of-the-art technology evaluation methodology, but lost marks as an affordable and simple method due to the significant amount of research still required. On the other hand, EBD will also require research and development to fully implement, but the degree of research required is less. Both EBD and TIES for SoS merit enough value to pursue as new technology evaluation methodologies, each providing unique contributions. However,

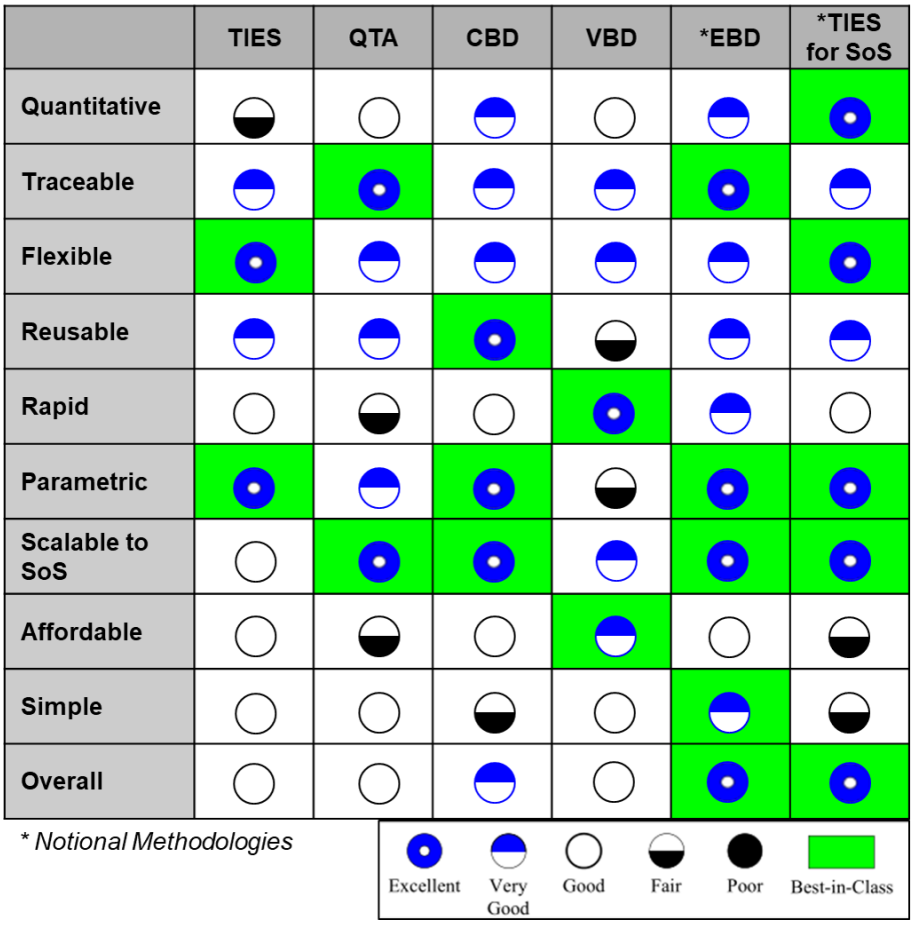

Figure 2.9: Ranking of Technology Evaluation Methodologies modified from [7].

these findings reveal that further development in technology evaluation methods are needed in order to improve and shorten the development and acquisition process.

The common thread among emerging technology evaluation methods is the need to quantify mission effectiveness. Biltgen, the developer of the CBD for SoS methodology, wrote in 2007 that current "military decision-makers lack techniques to rapidly assess the mission effectiveness of technology-rich systems-of-systems across a broad trade space encompassing multiple concepts, technologies, tactics, missions, and domains" [7]. QTA, CBD for SoS, and VBD all succeeded in modeling mission effectiveness for military systems; however, their Achilles heel is the use of a fixed mission design space that did not fully quantify the effectiveness of a technology. Quantifying the entire mission design space is traditionally conducted late in the design process with a fixed vehicle; however, if performance characteristics are not satisfied at this point, there is little to no room left to iterate on the design [33]. Properly exploring the mission design space of new technologies could enable better technology development decisions by providing the decision maker with more practical design metrics grounded in realistic mission simulations.

Current design methods can be improved upon by utilizing methodologies focused on quantifying mission effectiveness during the conceptual design phase of a development program to enable more design knowledge and design freedom. The characterization and modeling of the ways trade space is a gap in the TIES for SoS methodology. The exploration of both the means and ways in a design space could be a critical tool for technology evaluation; however, many steps must occur before utilizing that methodology. Both EBD and TIES for SoS call for better quantification of the mission effectiveness by exploring a larger portion of the mission design space. This observation is outlined below.

***Observation**: Further research and development into the characterization of the mission design space and the quantification of mission effectiveness is necessary to enable the proposed technology evaluation methods.*

2.2 Methods for the Design of Operations Analysis

> *"A concept based in tactics or technology is interesting, but only when paired with a concept of operations can it become compelling" [20]*

The discipline of OA can significantly vary in scope, complexity, and fidelity. The key is to find the balance in the scope and complexity of the mission while maintaining the fidelity of the vehicle model. Design methodologies must be able to design and compare both evolutionary and revolutionary technologies to understand their impact on a system or a family of systems. From this perspective, it is necessary to understand the hierarchy of system architectures in conceptual design. Technologies impact the subsystem they define, the system they are a part of, and their place in a system-of-systems environment [7]. Technology evaluation requires that each level of a system architecture be considered to some degree in design. Systems-of-systems research seeks to understand the interaction and behavior of highly complex scenarios, which usually requires a reduction in the fidelity of the vehicle models for practical reasons. Meanwhile, 6 degree-of-freedom (DOF) analyses require an immense amount of vehicle knowledge to model the flight characteristics, making that level of fidelity prohibitive in capturing the interaction or trade-offs of joint capabilities. The transition from foundational/physics level to campaign level modeling is illustrated in Figure 2.10.

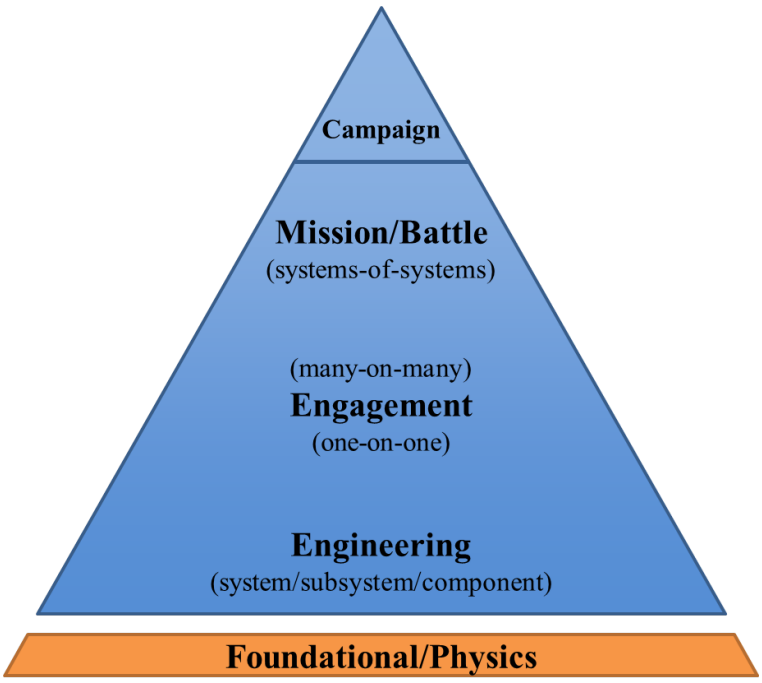

Figure 2.10: The progression of modeling from the foundational/physics level to the campaign level [41].

As the demand for highly integrated systems increase, foundational/physics level models and the engagement or mission/battle models must also become more integrated to investigate a variety of combat levels with varying degrees of fidelity while also reducing risk, costs, and developmental and acquisition time. Some modeling methods focus on modeling one level of the pyramid very well, while other methods provide a more flexible framework to modeling a variety of combat levels with various fidelity.

2.2.1 Mission Modeling Frameworks

Modeling OA can be conducted in a spectrum ways, all related by their level of abstraction – "the process of mapping the problem from the real world to its model in the world of models" [42]. Before further discussing OA modeling techniques, it is important to define what is meant by a model.

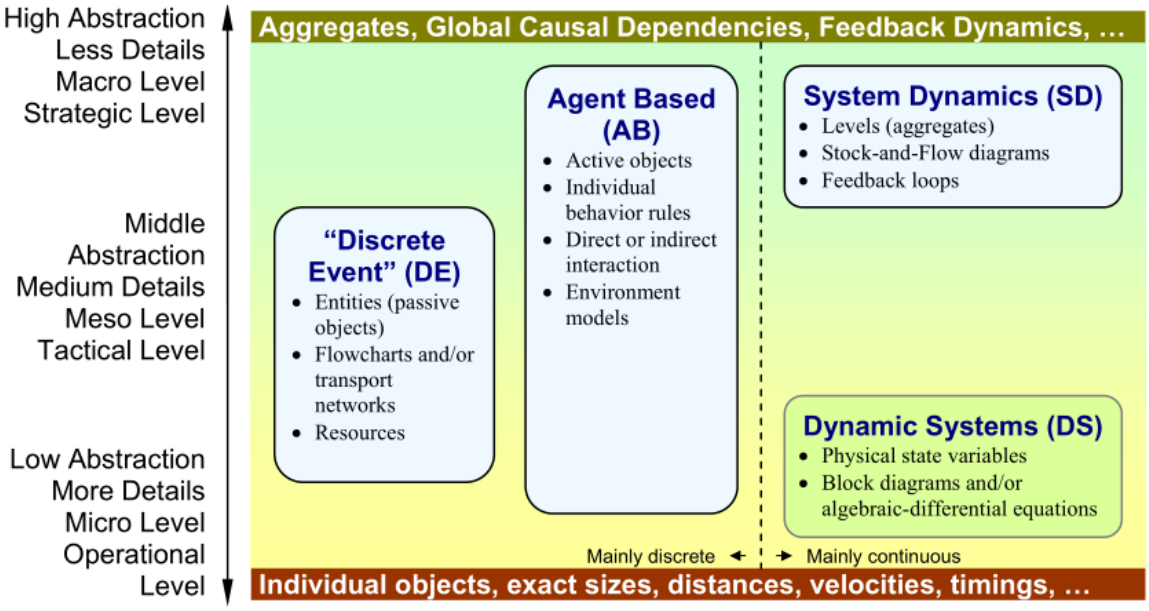

Figure 2.11: Comparison of modeling and simulation methods [42]

Definition: "A model is an abstraction of reality or one's concept that is used as an aid in answering a set of questions or to aid in communication" [43]

The main motivation to formulate a model that represents some aspect of the real world lies in the inability to conduct an experiment due to factors such as cost, complexity, or forecasting. Abstraction is necessary in modeling OA because not all real world complexities can or need to be represented in a model. The degree of abstraction for modeling is illustrated in Figure 2.11.

Borshchev identifies several modeling methods and their respective abstraction levels. He further splits the modeling methods into discrete methods, such as Discrete Event Simulation (DES) and Agent-Based Modeling and Simulation (ABMS), and continuous methods, such as System Dynamics (SD) and Dynamic Systems (DS). Each technique has their place in modeling military systems. [39] and [43] use DES for a Humanitarian Aid/Disaster Relief model; [7] uses ABMS for technology evaluation for SoS; [44] uses SD for modeling a maintenance supply chain for military weapons; and [45] uses DS for aircraft missile avoidance. This thesis will not investigate SD methods due to the high abstraction level

used in the models. DES, ABMS, and DS methods may be better suited for modeling highly integrated systems due to their low to medium abstraction levels. Before further discussion, a simulation – as opposed to a model – should also be defined.

Definition: A simulation "is the process of model 'execution' that takes the model through (discrete or continuous) state changes over time" [42]

Dynamic Systems (DS)

Dynamic Systems (DS) are typically used to model a continuous variable space. Such models are common in modeling "physical" systems for many engineering disciplines [42]. In the context of military tactics, optimal control is commonly used to determine the control laws of a dynamic system. Optimal control can be used to model military engagement tactics for scenarios such as aircraft missile evasion, ballistic missile intercept, or cooperative missile evasion [45, 46, 47]. These types of scenarios are categorized as variants of the pursuit-evasion problem. Such studies of pursuit-evasion can provide novel insight into engagement tactics using unique objective functions [48].

The low level of abstraction in dynamic systems could enable the employment of physics-based models providing a higher fidelity solution. Optimal control typically utilizes kinematic equations to model the system which can provide meaningful results to understand optimal engagements. However, those results typically require many assumptions and simplifications of the problem [45]. Assumptions are typically required because it can be very challenging to determine an analytic solution for nonlinear dynamics in optimal control problems, such as those found in the pursuer-evader problem [49]. As the complexity of the optimal control problem increases, the difficulty in finding a solution can be cumbersome. Such a modeling method may not be suitable for modeling the mission action design space for problems which desire the integration of physics-based models.

Discrete Event Simulation (DES)

The field of Discrete Event Simulation (DES) has been the main technique used for Operations Research (OR) in academia and industry for almost 50 years [50]. DES explores sample paths in a dynamical system that aims to characterize the models behavior [51]. As evident by its name, DES models change at discrete-time intervals, rather than continuously. Although ABMS is also predominantly modeled using discrete-events, DES differs in its representation of entities. Entities in DES have no capacity to learn or develop cognitive reasoning [52]. In the context of DES and ABMS, an entity is defined as "an object in a system that requires explicit representation within a model of the system" [51]. These entities typically rely on an "event scheduling function" that changes its state [52]. Siebers et al. defines the attributes of a DES model as:

- Focused on modeling the detailed system, not the entities
- Top-down modeling approach
- Centralized control
- Passive entities; intelligence lies in the system
- Queues are a key element
- Macro behavior
- Input distributions are typically based on objective data [50]

Applications of DES include manufacturing, distribution, communication, transportation, health-care, and many more [51]. In the context of military applications, DES can be used from training to wargaming for all aspects of the military including acquisition to force deployments [53]. Additional examples that may be helpful in understanding the capabilities of DES relate to humanitarian aid and disaster relief missions [54, 55, 43, 39].

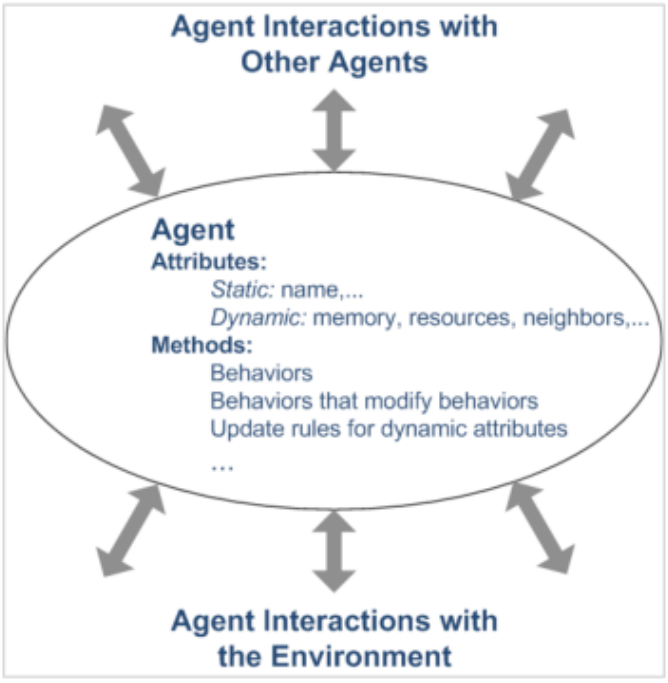

Figure 2.12: Elements of an agent within an Agent-Based Model and Simulation environment [56]

The overall capability of DES is vast with many possible applications of the modeling framework. DES may be most suitable for campaign or mission level simulations due to its ability to model macro behavior with an abstraction level in the middle of the spectrum.

Agent-Based Modeling and Simulation (ABMS)

Agent-Based Modeling and Simulation (ABMS) overlaps with the field of DES in many ways, such as their use of discrete-event models, but some argue its difference has to do with the use of active entities as opposed to passive entities. Active entities are those that have some degree of autonomy and proactive behavior [50]. ABMS can often be defined using three main elements: the active entities or *agents*, the relationships between agents, and the agent's interaction with their environment [56]. Figure 2.12 illustrates these three elements of ABMS.

Applications of ABMS have been used by many in academia [57, 7, 53, 58]; however, their implementation in industry may not be as prevalent due the lack of agreed upon

methodologies [50]. Siebers et. al also defines the attributes of ABMS as:

- Focused on modeling interactions of the entities
- Bottom-up modeling approach
- Decentralized control
- Active entities; intelligence lies within each individual entity
- No concept of queues
- Macro behavior emerges from the micro decisions of the entities
- Input distributions are typically based on subjective data [50]

The idea of a bottom-up modeling approach is another key difference between ABMS and DES. Development of an agent-based model usually starts with defining the agents and their environment without defining their specific actions within the simulation. This modeling framework may be suitable to model simulations from the campaign to engagement level of combat due to its ability to model simple or complex micro decisions, which depends on the fidelity of the agent defined.

2.2.2 Quantifying Mission Effectiveness

The terminology used in literature to define the effectiveness of the system and its mission are similar, but have subtle differences. Tillman defines system effectiveness as "the over-all capability of a system to accomplish its mission" [59]. This definition varies only slightly to how Tillman defines mission effectiveness – "the probability of successfully accomplishing the mission." Both system and mission effectiveness seek to quantify how successful a system performs its mission; however, system effectiveness is more concerned with evaluating the *capability* of the overall system while mission effectiveness is more

focused on the mission outcome. Biltgen identifies several definitions from literature of capability in the context of military acquisition and design. One such definition of capability is "the ability to achieve an effect ... to perform a set of tasks" [7]. From this perspective, system effectiveness should be used to define the capability of a system across multiple tasks or missions. Alternatively, mission effectiveness should be used to evaluate a single mission. System and mission effectiveness can be formally defined as:

Definition: System Effectiveness is the over-all capability of a system to accomplish its mission(s) [59]

Definition: Mission Effectiveness is the degree a system is able to successfully complete a mission

Since the emergence of mission effectiveness during the space age, its definition in literature has changed very little [60, 59, 61]. In contrast, the quantification of mission effectiveness has varied in literature. [60] quantifies mission effectiveness through some combination of availability, reliability, and performance of the system. [62] proposes that mission effectiveness should be comprised of survivability and some quantification of the work performed and objectives achieved. More recently, mission effectiveness has been quantified by Measures of Effectiveness (MoEs) and subsequent Measures of Performance (MoPs) [7, 43, 54, 55, 63]. MoEs and MoPs are formally defined as:

Definition: Measures of Effectiveness "provide a standard by which it can be established how well some thing achieves the purpose for which it is intended" [64]

Definition: Measures of Performance "are units established to quantify those properties of a system necessary to determine the capabilities of a system" [65]

As evident from the above definitions, MoEs and MoPs seem to vary only slightly. Sproles uses an analogy of effectiveness versus efficiency to better distinguish between MoEs and MoPs. MoEs or effectiveness relates to "how well something does its job" while MoPs or efficiency relates to "how well something does what it is doing" [66]. Alternatively, MoEs ask "are we doing the right things?" and MoPs ask "are we doing things right?" [67]. A system can have great MoPs without completing its intended mission successfully. From this perspective, Sproles goes on to explain that MoEs relate to the external viewpoint – the mission – and not any particular system, while MoPs quantify the internal viewpoint [66].

The overarching technology evaluation methodology should understand the effect MoEs can play in solution development. Improper MoEs could impact the direction a solution takes [64]. Such bias could occur in mission modeling. In order to maximize a desired MoE, the modeler may focus on modeling certain tactics that may or may not be optimal for the overall mission effectiveness. This bias from the MoEs can be reduced through formal processes such as the MoE formulation process illustrated in Figure 2.13.

The formulation of MoEs should consider the Critical Operational Issues (COIs) which evaluate a system's ability to succeed in a mission. The focus on COIs is in contrast to a focus on mission parameters, objectives, or thresholds [64]. This perspective lines up with the EBD methodology which seeks to model an open mission design space with minimal definition by focusing on operational outcomes [34]. Such a perspective requires further characterization of the mission design space to be able to extract operational outcomes as opposed to varying mission design parameters.

2.2.3 Characterization of the Mission Design Space

Both the EBD and TIES for SoS methodologies seek to explore the SoS mission design space without pre-defining a mission path or tactic. This approach – keeping the SoS design space open – could allow for the exploration of novel combinations of technologies

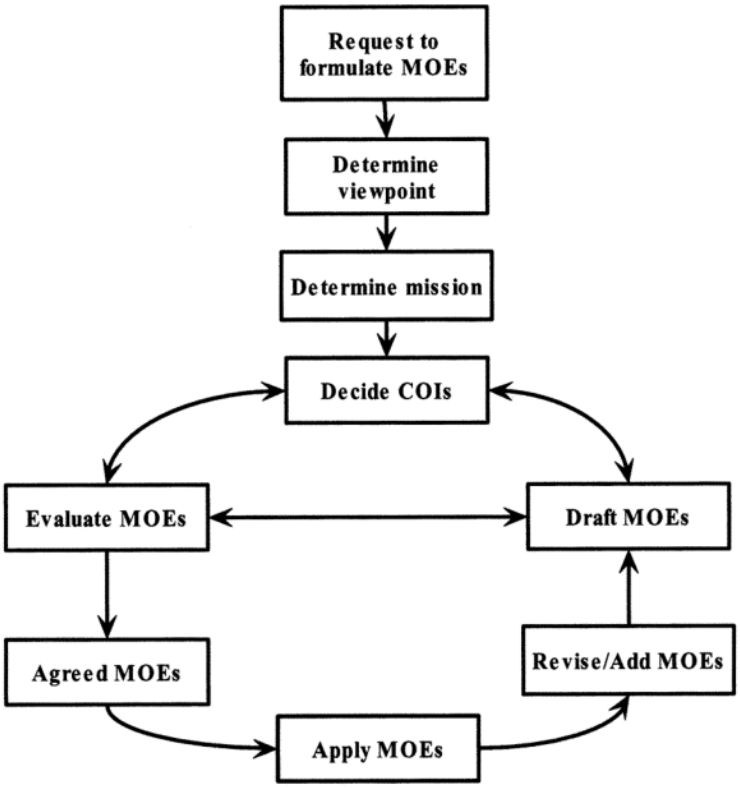

Figure 2.13: A process to formulate MoEs [64]

and tactics. EBD specifically seeks to develop an OA that has "no pre-determined mission path, number of platforms, or mission envelope" defined [34]. The benefit of identifying novel tactics seems straight forward. A novel tactic could allow for the use of an evolutionary technology that may be cheaper and provide lower risks compared to a revolutionary technology. Alternatively, the best tactic for a revolutionary technology may be unknown, in which case, mission design space exploration could identify the effective use of the technology. In either situation, the characterization of a mission design space needs to be defined before any tactics can be identified.

In the field of operations research, very few articles have been published on characterizing the mission design space. Most research in this area focuses on characterizing the players within a simulation. Characterization of the player – or agent in the context of agent-based modeling – has been studied much more extensively due to its utility in many fields such as business process modeling, film-making, manufacturing, robotics, and

video games [68]. A recent attempt to characterize the mission design space defined two categories, which were mission architecture variables and mission action variables [57]. Mission architecture variables were defined as the overall properties of the scenario in a more aggregate manner. These properties include things such as the type of assets used or the level of coordination permitted.

The second category within the mission design space is the mission action variable space. This category contains the detailed breakdown of the actions taken during a scenario such as entity movements, current targets, firing order, etc. [57]. Defining the architecture of the mission and the actions within a mission provides insight into the mission design space that could aid in the field of operations research. Although further characterization within the mission architecture space and mission action space could be conducted, this initial characterization may provide the beginning of efforts to defining the *ways* design space.

2.2.4 Exploration of Tactics

> *"Perfection of means and confusion of ends seem to characterize our age"*
> *–Einstein [69]*

Understanding the ends – or the *ways* – a system or technology is used seems to still be an afterthought in conceptual design. Limiting factors include the lack of research and modeling capabilities required to effectively model the *ways*. This thesis seeks to define a formal methodology to aid in the understanding of the *ways* design space by improving the modeling capabilities of mission effectiveness.

The *means* and *ways* design space can be more simply defined as assets used to perform a mission and the tactics used to complete a mission. These definitions line up fairly well with the mission architecture and mission action design spaces. Therefore, efforts to quantify the mission action design space, or the *ways*, involve the exploration of tactics. Tactics are defined as the employment of available means needed to attain the objectives [70, 71].

SAA:	The Stochastic Agent Approach
Step 1:	Develop an agent-based model. Define a balanced agent decision-making structure. Specify mission architecture variables and MoEs of interest.
Step 2:	Identify agent-based variables in the decision-making algorithm. Classify agent-based variables.
Step 3:	Identify the agent action design point(s) with non-trivial variability. Run the model and use Bootstrapping to determine the appropriate number of replicates for convergence.
Step 4:	Use a space filling Design of Experiments to inform mission space exploration. Produce statistical distribution metrics from each design point.
Step 5:	Use the distribution metrics from the output to develop a Neural Network. This 'black box' equation is then used for higher-level campaign simulations.

In the world of operations research, a methodology exists that formulates a 'black box' model to represent the tactical capabilities of a technology. The purpose of such a model was motivated by its use as a surrogate model for higher-level campaign simulations. The Stochastic Agent Approach (SAA) provides a five step methodology to develop a surrogate model of the tactical capabilities of a technology [57]. This methodology is summarized below.

The SAA succeeded in characterizing the mission design space and developing a way to model the mission action design space. Application of the SAA to aid in technology evaluation must be examined. This examination should identify any gaps that may exist in the methodology when applied to exploration of the mission action design space for technology evaluation.

The first step in the SAA reduces the reliance on subject matter experts (SMEs) during tactics formulation. Tactics are typically formulated a priori or identified posteriori of the mission simulation by SMEs. The use of SMEs in the modeling and simulation of tactics along side system design has been shown to produce a more effective design [72]; however, both methods have drawbacks due to their reliance on SMEs. The formulation of tactics a priori the mission simulation may immediately limit the tactics considered [7, 72]. This can quickly become evident when revolutionary technologies – those without historical data –

are included in the simulation. In those simulations, SMEs may have no knowledge on how the revolutionary technology should be used at the tactical level of combat. The SAA approach utilizes state machines with pseudo decision trees to define the transition behavior between states for their agent formulation. Such a formulation may result in overly defined state transition criteria that could be constraining the agent's tactical options. A method to formulate an agent's framework that can reduce a modeler's reliance on SMEs is desired to enable tactics formulation.

The second step in the SAA, which involves the formulation of a decision-making algorithm for tactics, is loosely defined in literature. A decision-making algorithm for tactics must be formulated in such a way as to not limit the solution outcome. The SAA defines Agent-Based Variables (ABVs) that can be tuned through stochastic simulations to provide a probabilistic transition values which represents an agent's decision-making likelihood. However, once the state machine is tuned, the ABVs do not change. The fixed nature of such a variable does not capture the temporal aspect of decision-making. The agent framework may capture optimal stochastic behavior for a single event in its mission, but as the environment changes, the agent is unable to change its behavior. For example, an agent's preference to engage or evade a defender may change over time due to a degradation in capability. The temporal aspect of decision-making must be included in the exploration of the mission action design space. This presents the second gap in the current methodology. Further definition of the agent's decision-making approach is required to enable tactics exploration.

The third and fourth step in the SAA conduct the exploration of tactics. These steps involve the execution of the simulation. When SMEs are utilized during step 1, the exploration of the mission action design space has already been conducted during tactics formulation due to the limited number of tactics chosen to model. The use of SMEs is typically required due to the dimensionality of the mission action design space. However, the SAA managed dimensionality by investigating a decision-making variable space as opposed to

action variable space. This perspective helps to manage dimensionality, but may not address the complexity encountered for multi-agent systems. The decision-making state space for cooperative and/or competitive multi-agent systems increases exponentially due to the inherent complexity of modeling interacting agents. Tackling the challenge of dimensionality must be addressed for a single agent system before exploration of cooperative and/or competitive tactics can be explored. Additional methods to manage the dimensionality of the mission action design space are required to enable exploration of tactics for technology evaluation purposes.

The SAA does not address its applicability to interacting multi-agent systems. Such systems make up a large part of the mission action design space due to the larger system-of-systems problem that is motivating technology evaluation. The ability to measure a technologies effectiveness in integrating with existing platforms is one of many reasons why multi-agent systems are required to capture the mission action design space. The extension of the SAA for multi-agent systems must be addressed to close the gap in modeling capabilities of the mission action design space.

The fifth step of the SAA does not address the trade-off of technologies and tactics. This is purposeful due to the motivation behind the development of the SAA. However, extending the SAA for technology evaluation requires that optimal tactics can be identified for each technology of interest. Tactics identification typically utilizes SMEs posteriori the mission simulation. The use of SMEs for this step may have limitations which are quickly evident for large sets of simulation data. The complexity of identifying tactics stems from the difficulty in defining what an 'interesting' or 'significant' pattern is within the data that warrants the label of a tactic. Devaney used a variety of experts and non-experts to identify tactics within several large battles [73]. These large battles consisted of a combined 1277 friendly and enemy units. Although the research succeeded in identifying tactics from the data, it also found disagreement among subjects on the level of granularity of a pattern or tactic.

Another limitation of tactics identification by SMEs is their inability to quickly review hundreds of battles. Devaney limited the study to 15 battles due to the time required for the subjects to identify tactics from the large data sets. Tactics evaluation will require a new method to sift through the data of single- and multi-agent simulations to determine the optimal tactics. Such a methodology must also be traceable to evaluate the combination of optimal tactics with their respective technologies. Once determined, the modeler must be able to effectively compare each technology and tactic combination to aid in the over arching technology evaluation methodology.

Several gaps were identified in the application of the SAA to evaluate the *ways* tradespace. The agent's formulation, the agent's decision-making framework, the dimensionality of the mission action space, the application of such an approach on multi-agent systems, and the proper evaluation and comparison techniques for technology-tactic combinations must be addressed in order to use such an approach for the exploration of the *means* and *ways*.

A new framework is needed to solve the identified gaps in current mission modeling and simulation methodologies. Reducing a model's reliance on SMEs to define tactics a priori could allow for an increase in the exploration of the tactics design space. A framework that places minimal limitations on the decision-making tactics a model can simulate would provide an improvement in modeling capability for the mission action design space. A well designed methodology would maintain traceability to enable tactics evaluation. This new methodology for tactics exploration could aid in the quantification of mission effectiveness. This observation is stated below.

Observation: *A new methodology for the formulation and exploration of tactics is required to enable the quantification of mission effectiveness.*

2.3 Research Focus: Development of a New Methodology

This thesis seeks to shorten the development and acquisition process by developing a framework that better quantifies mission effectiveness for use in the design and evaluation of aircraft technologies in adversarial environments. The focus of research will be in exploring the gap in modeling a minimally defined mission to explore the mission action design space. This work specifically aims to identify the necessary steps needed to formulate and explore tactics within the mission action design space. The majority of the research will explore tactics modeling with the goal of quantifying mission effectiveness for new technologies. The revised research objective is stated below.

Revised Research Objective: *Enable the formulation and exploration of tactics to augment the quantification of mission effectiveness in adversarial environments to reduce risk and costs, and shorten the development and acquisition life cycle.*

Many aspects of the SAA and technology evaluation methodologies can be carried over to inform the steps of a new methodology. The structure of a new methodology should be comprised of four general steps defined as problem definition, tactics formulation, tactics exploration, and technology-tactics evaluation.

The problem definition should seek to substantiate the mission and MoEs, the technologies of interest and the agent-based model. The formulation of tactics seeks to develop an agent for use with a minimally defined mission. Tactics Exploration conducts mission action design space exploration through perturbation of an agent's tactics. The final step involves the evaluation of tactics which seeks to quantify the system effectiveness of each technology-tactic combination. This final step involves quantifying the technology's capability by comparing performance metrics and mission effectiveness. Some aspects of the

methodology are already well understood; however many aspects of the proposed methodology, such as the definition of an agent-based model for a minimally defined mission, need further research in order to enable the evaluation of the mission action design space. This proposed methodology is summarized below.

Proposed Methodology	**literature review required*
1. Problem Definition:	Define the mission and MoEs
	Identify technologies of interest*
	Identify an agent framework*
2. Tactics Formulation:	Develop agent-based model for minimally defined mission*
	Define algorithm for an agent's behavior*
3. Tactics Exploration:	Train the agent(s)*
	Perturb the ways*
4. Technology-Tactics Evaluation:	Infuse technology-tactic combinations
	Evaluate system effectiveness

CHAPTER 3
FORMULATION OF METHODOLOGY

The overall research problem seeks to model tactics alongside technologies in a more open mission design space. In the context of this research, an open mission design space is one that uses a minimally defined mission by reducing the constraints and preset criteria within the defined modeling and simulation environment. Before technology and tactic trades can be conducted, a methodology needs to be defined for tactics modeling. The scope of this research will focus on the mission action design space while holding the mission architecture variables as fixed values. This chapter will address the gaps in the proposed methodology to enable exploration of the mission action design space for technology evaluation methodologies.

The development of a new methodology will be pursued in the context of a relevant scenario. The A2/AD strategy has the potential to deny reconnaissance aircraft from accessing contested air space. The investigation of a contested reconnaissance mission could provide a standard problem to conduct technology evaluation. This mission provides similarity and applicability to the field of military tactics while minimizing the complexity of potential tactics. Specific M&S settings will be defined and further discussed in Section 5.

3.1 Problem Definition

The first step in the proposed methodology seeks to define the the mission and relevant metrics, identify the technologies of interest and formulate the agent-based model. The mission definition and MoEs should provide clear objectives without limiting the tactics used to achieve the objectives. The mission alternatives and technologies of interest can be determined using a morphological analysis similar to the ones implemented in TIES and its predecessor, TIES for SoS [24, 39]. This analysis, introduced by Fritz Zwicky [74],

Vehicles	Relative Quantity		Many	Average	Few	None
	Size/Capacity		Large	Moderate	Small	
	Range		Long	Moderate	Short	
	Transit Frequency		Weekly	Biweekly	Daily	12 Hourly
	Features	Component Robustness	Majority Consumable	Balanced Consumable/Repairable	Majority Repairable	
		Security Architecture	Complete Threat Avoidance	Most Prevalent Threat Avoidance	Minimal Threat Avoidance	
		Comm/Nav Redundancy	All Systems Redundant	Critical Systems Redundant	No Redundancy	
	Weather/Environmental		Extreme	Moderate	Limited	
Tactics	Package Delivery		Manned	Unmanned		
	Refueling Capability		Ferry Tank	Aerial Refuel		
	Refueling Operability		Gas-Go	Remain Over Night		
	Level of Autonomy		Fully Autonomous	Semi-Autonomous	Teleoperated	

Figure 3.1: Example of a morphological matrix with vehicle and mission attributes [39].

provides a structured aid during the generation of technology alternatives. The morphological matrix is initially used to identify the vehicle architecture's concept design space as well as a baseline vehicle concept. This matrix should consist of both evolutionary and revolutionary technologies.

Implementation of the morphological matrix can also extend to the mission concept space. Figure 3.1 illustrates the vehicle and tactical alternatives for a humanitarian aid and disaster relief mission [39]. This concept space analysis enables the definition of the technology and mission concept space without placing any constraints on the mission.

The morphological matrix provides a satisfactory amount of mission and technology alternative definition without limiting the mission action design space. Extension of the traditional concept alternative matrix to the mission and tactical options has been proven. The formulation of an agent-based model is suspected to have a much larger influence on the mission action design space.

3.1.1 Identifying the Modeling Framework

Defining a problem that can explore the mission action design space requires a more targeted framework to enable the formulation and exploration of tactics. Background research has already revealed a down selection in modeling methods that can be made to formulate a more targeted problem definition framework. Both ABMS and DES were identified as common mission modeling frameworks. Research findings suggest that DES may be more suitable for problems that are focused on higher abstraction levels. DES is a useful framework for OA, but its applications are more aligned with the mission architecture design space [57]. The exploration of the mission action design space will require a lower level of abstraction to capture higher fidelity mission effects. The need for a higher fidelity mission modeling framework also aligns with the objectives of the EBD methodology. One of the three main objectives, highlighted in Section 2.1.5, was the need to numerically couple physics-based models from the MDAO M&S environment and the OA M&S environment [34].

The level of fidelity desired for the mission action design space aligns more closely with ABMS. An agent-based modeling framework provides a larger range of model fidelity available to the modeler. The choice of Agent-Based Model (ABM) would not limit the possible missions or mission paths that could be modeled due to its ability to incorporate physics-based models. For this reason, the first step in this methodology will seek to identify the mission and define an agent-based model; however, the mission path should not be defined a priori to prevent unintentional limitations on the mission action design space.

An agent-based model can be formulated many ways depending on the purpose of the model. For the purpose of mission action design space exploration, the agents within the model need to be defined in a specific way as to not limit the mission paths a priori. Further research into ABMS methods find that there are many agent framework formulation methods available. These methods provide a range of definitions of behavioral rules for the agent to follow. Explicit agent behavior frameworks such as decision trees and state machines

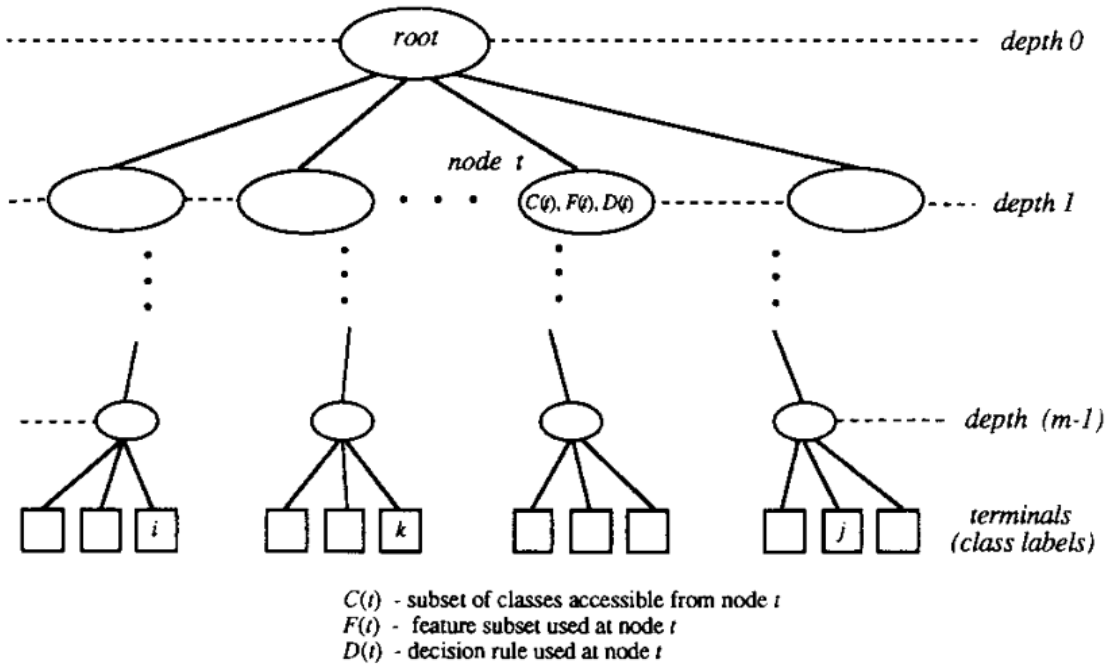

Figure 3.2: General framework of a decision tree for classification [75].

are reviewed and compared to more abstract agent frameworks with implicit behaviors.

Decision Trees

Decision trees are one of the most common decision making tools seen in ABM [57]. They have been successfully implemented for applications such as radar signal classification, medical diagnosis, remote sensing, and speech recognition [75]. The diverse set of applications that decision trees can be used for is due to the simplicity of the decision-making tool. The structure of a decision tree is typically a hierarchical set of branches stemming from a single root. Figure 3.2 illustrated a general framework of a decision tree. Each branch in the decision tree can be converted to an if-then-else statement based on a user defined condition [75].

One example of a decision tree implementation in an ABM is for a predator-prey scenario. The complex interaction of the two agent types – predators or prey – can be represented as a simple set of rules that define their behavior. Each agent type has a unique

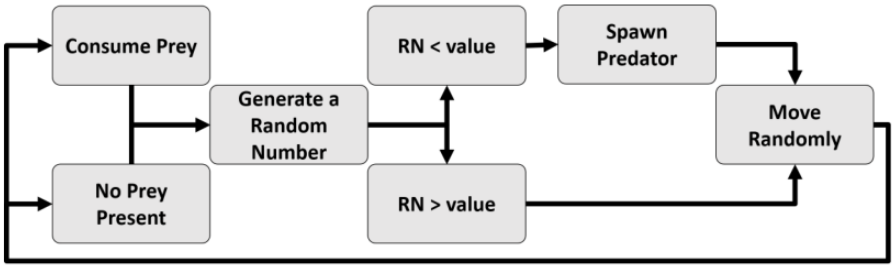

Figure 3.3: Decision making structure of a predator in a predator-prey ABM [57].

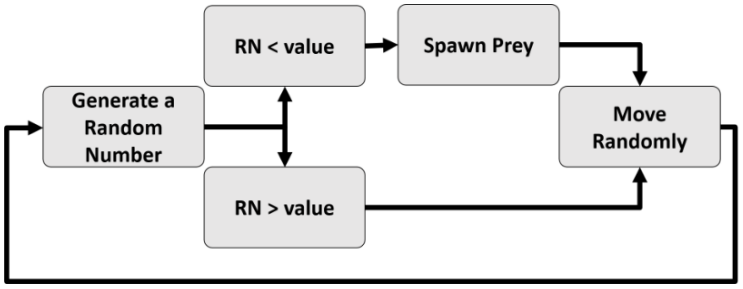

Figure 3.4: Decision making structure of prey in a predator-prey ABM [57].

decision tree, illustrated in Figures 3.3 and 3.4. Each branch in the tree is chosen based on a conditional statement that involves a random number (RN) compared to some value. Although simplistic, this ABM framework can provide heuristics to a complex interaction with the potential to identify macroscopic behavior between the populations of both predators and prey. The heuristic nature of decision trees is an advantage when compared to black box methods due to the traceability of an agent's decision-making process [76].

More complex decision trees can be developed and implemented in ABMS; however, more complex decision trees may not produce more realistic agent behavior. Several drawbacks of a decision tree framework occur as the depth and number of nodes increase. Large decision trees can have nodes that overlap or produce the same outcome from different paths [75]. This overlap can be inefficient for large decision trees due to the "memorylessness" of a decision tree framework. Every time the root decision tree criteria is called, the entire decision tree is traversed without prior knowledge of past decisions, significantly increasing computations [57]. Another drawback present in increasingly complex decision trees is the propagation of error. As a decision-making algorithm traverses the levels in a

decision tree, errors from each level can accumulate [75].

The implementation of decision trees can be successful for some hard coded behaviors to avoid overlap in branches and the propagation of errors; however, the inherent drawback of hard coding an agent's behavior is the potential to reduce the abstraction of the behavior. Research has been conducted to overcome this by producing abstract behaviors affected by many possible conditions through the use of decision trees generated from a genetic algorithm [77]. This research modeled the co-evolution of agents in a predator-prey model to produce more intelligent decision trees through artificial co-evolution. The use of an intelligent decision tree may be a possible framework to use for agents to explore the mission action design space.

State Machines

State machines provide an alternative way to model agents in an ABM. This framework focuses on the possible states an agent can be in rather than traversing a set of discrete decision branches in a decision tree. The decision-making aspect of state machines comes into play when formulating the criteria required to transition states. State transitions can occur given a user defined decision or a reaction to stimuli [57]. Each state can have multiple decision criteria that initiate a transition to a new state or back to a previous state. For the simplistic predator-prey scenario, the decision tree and state machine are fairly identical in definition and implementation. The state machines of a predator and prey, illustrated in Figures 3.5 and 3.6 respectively, can simplify the agent's framework by condensing several decision tree branches into each state. This advantage becomes much more significant as the complexity of the agent increases.

Another distinguishing characteristic of a state machine is its ability to be aware of its current state [57]. This characteristic is significant when compared to large decision trees. Decision trees are typically evaluated in their entirety for each time step to determine the correct decision to make. Meanwhile, state machines do not need knowledge of the entire

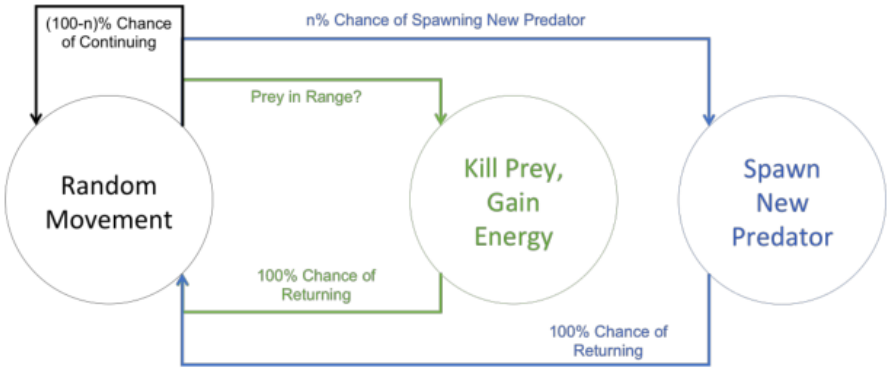

Figure 3.5: State machine structure of a predator in a predator-prey ABM [57].

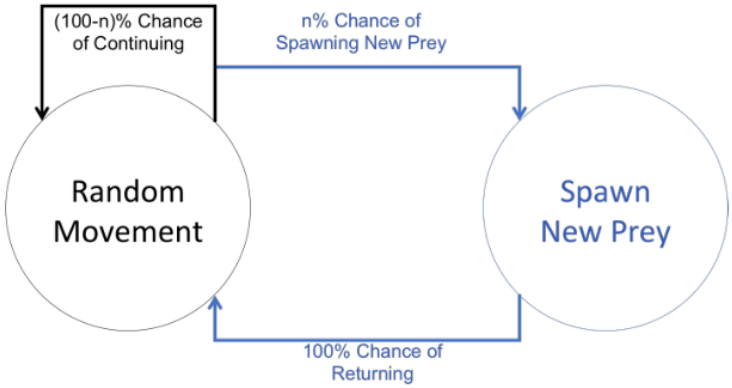

Figure 3.6: State machine structure of prey in a predator-prey ABM [57].

set of defined states of a system and only need to review the set of state transition criteria relevant for that state. The ability to predefine each state with a template of predetermined state transitions significantly reduces the complexity of each behavior by isolating the transitions in and out of a state [78]. These characteristics of state machines can also be found in a sub-category of state machines known as Markov Decision Processes (MDPs).

MDPs are state machines that have states and transition criteria similar to those shown in the predator and prey state machine models. This framework explicitly defines the states of the system and aim to better define the relationship between microscopic and macroscopic properties [79, 80]. The best action is determined by inputting the state of the system. MDPs assume that the entire state of the system is known, but the impact that the chosen action will have on the system is uncertain [81]. More formally, an MDP can

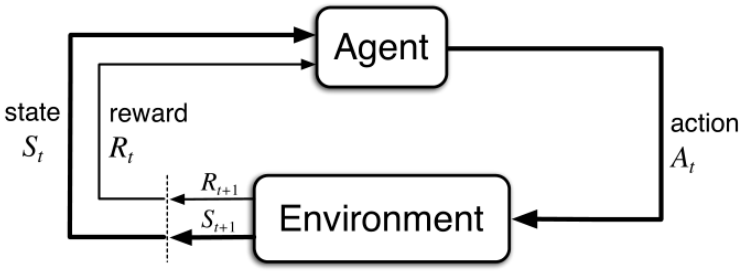

Figure 3.7: Markov Decision Process of an agent interacting with its environment [82].

be defined by $\langle S, A, T, R \rangle$. S defines the set of finite states of the system or environment, A defines the finite set of possible actions to choose from, T defines the state-transition function, and R defines the reward function. The state-transition and reward functions determine the policy, $\pi(s)$, to choose state s' and the immediate reward given state s and action a, respectively. This type of framework, illustrated in Figure 3.7, is similar to those used in traditional Artificial Intelligence (AI) systems which plan a set of sequential actions. However, a plan may rarely execute as expected in an uncertain system or environment. In order to create a more open mission action design space, an AI planning model that can operate in a stochastic environment is desired.

MDPs with imperfect information differ from traditional MDPs and state machines due to the introduction of stochasticity into the transition criteria between each state [80, 83]. This allows the agent to react to an uncertain state. MDPs with imperfect information could be useful in defining a system or ABM where not all state variables are known [84]. This class of MDP is formally known as a Partially Observable MDP (POMDP) [81]. POMDPs is generally defined by $\langle S, A, T, R, \Omega, O \rangle$. S, A, T, R are defined above by the MDP, Ω is the set of finite observations the agent may experience, and O is the observation function which defines the probability of an observation given action a and chosen state s'. The ability of an agent to use observations to estimate its state would enable further exploration of the mission action design space through the ability to reevaluate its actions based on changing observations of the environment. POMDPs provide a possible framework for the agents of an ABM.

- Beliefs: Information about the environment; *informative*.
- Desires: Objectives to be accomplished, possibly with each objective's associated priority/payoff; *motivational*.
- Intentions: The currently chosen course of action; *deliberative*.
- Plans: Means of achieving certain future world states. [89]

Abstract Agent frameworks

Several agent frameworks that are based on more abstract ideas aim to define the internal behavior of intelligent agents. Intelligent agents are distinguishable from agents due to their requirement to be reactive, proactive, and social [68, 85]. Decision trees and state machines could be used as the framework of intelligent agents, but the use of either of those agent frameworks does not inherently mean they produce intelligent agents. Two agent frameworks that focus on intelligent agents are the Belief-Desire-Intention (BDI) framework and the Observe-Orient-Decide-Act loop.

The BDI framework is a widely known agent framework that is based in philosophy [86, 87]. This framework is frequently used to develop intelligent agents that seek to model human behavior [88]. The BDI architecture excels at modeling reactive and deliberative behavior that mimics human reasoning. The central concepts of the BDI framework are described below.

This philosophical framework has moved into theory and implementation as a result of research into formalizing the relationship between the central concepts of BDI for use in developing an agent's behavior [90]. One notable formalism of the BDI framework is the notion of an agent's commitment to their goals. The ability of agents to revise their goals or abandon goals based on explicit criteria aided in the implementation of a BDI framework for developing intelligent agents [91]. An agent that utilizes a BDI framework is proactive as a result of its goals, reactive as a result of its internal events, and social due to its actions and precepts with the environment [89]. An intelligent agent based on the BDI framework

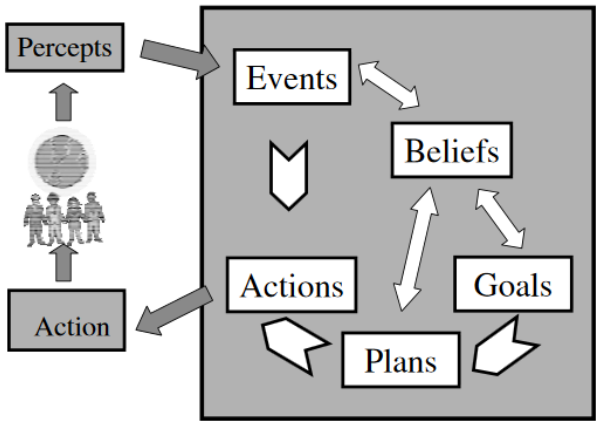

Figure 3.8: A generic agent architecture based on BDI concepts [89].

is illustrated in Figure 3.8.

Since the BDI framework was introduced, many methodologies have been developed for modeler's to use in developing intelligent agent architectures based on the central concepts of BDI [68]. However, these architectures have been seldom used due to their complexity and computational cost for large multi-agent systems [92].

The Observe-Orient-Decide-Act (OODA) loop, developed by USAF Col. John Boyd, could provide a simple yet intelligent structure for an agent's behavior. The OODA loop, illustrated in Figure 3.9, was originally developed for fighter pilots to use as a tool for air-to-air combat, but has since been utilized in corporations, governments, and militaries from the tactical to strategic level of an organization [93].

The four steps in an OODA loop are verbatim: observe, orient, decide and act. Although simplistic, the method can be applied to a diverse set of applications of varying complexity. The key to successful implementation of the OODA loop lies in the speed at which the user – or agent – can cycle through the steps. In the context of air-to-air combat, "the pilot who goes through the OODA cycle in the shortest time prevails because his opponent responds to actions that have already changed" [95]. The four steps of the OODA loop are defined below in a general context.

1. Observe: collect information from the environment

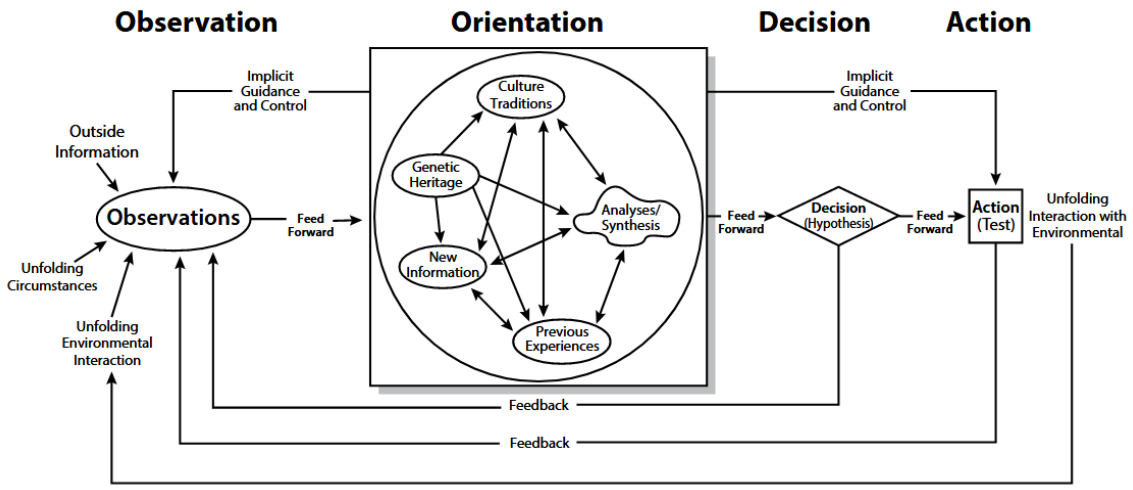

Figure 3.9: The Observe-Orient-Decide-Act (OODA) loop [94].

2. Orient: process the information for relevance and importance to gain a mental perspective

3. Decide: determine the course of action for the perceived situation

4. Act: execute the actions that correspond to the previously determined decision [93, 96]

The OODA loop has been utilized for agent decision-making to aid in an agents reasoning structure [96]. This framework may provide a more simplistic way to develop an intelligent agent that can sense, think, and act based on observations in its environment. It has also been used in literature as a classical state machine to define cognitive systems in an autonomous cognitive network [97].

Both the BDI framework and OODA loop are becoming popular tools in the decision-making process for intelligent systems due to their ability to conduct practical reasoning [98]. The application of these frameworks are typically those which seek to capture human decision-making in the ABM. Concepts from both frameworks may be useful to incorporate into the agent's framework, but caution should be taken when implementing either framework due to the intentionally abstract concepts.

3.1.2 Selecting an Agent Framework

The investigation of agent frameworks for an ABM identified several frameworks used in academia and various industries. Each framework seemed to have a best-use-case along with specific drawbacks. A proper agent framework for mission action design space exploration is one that defines no preset mission path or mission envelope to create an open solution space [34]. However, minimizing the number of preset mission options increases the complexity of the agent due to the desire to have agents that can explore undefined mission paths and envelopes. This type of agent also needs to have traceability in its decision-making process for the modeler to extract believable information from the actions taken by the agent. An agent framework for mission action design space exploration must balance a general agent framework with traceability in its decision-making process.

A review of agent frameworks from literature has identified several frameworks that are used in ABM. The choice of agent framework must satisfy the requirements for mission action design space exploration. Decision trees provide the user with traceability and heuristics for decision-making, but have many drawbacks. Some drawbacks of decision trees include the lack of intelligent behavior, inefficient computational requirements, and prohibitive size for complex agents. Meanwhile, state machines provide a more simplistic way to model complex agent behavior, but require SMEs to handcraft states and transition criteria. Further definition of a state machine using MDPs or POMDPs may provide a general agent framework with traceability due to its set of finite possible actions. The investigation of decision trees and state machines suggest that state machines would better satisfy the characteristics of an agent framework desired for mission action design space exploration.

In order to determine and justify the selection of agent framework, one must quantify the exploration of the mission action design space. This quantification can be conducted by reviewing the potential tactical options agents may have during a mission simulation. In the context of a contested reconnaissance mission, a decision tree and state machine can

be formulated for an agent. Figures 3.10 and 3.11 illustrate a decision tree and state machine agent framework for a contested reconnaissance mission, respectively. These figures visually show the contrast in complexity and efficiency. Decision trees not only look more complex, but also require many more decision branches for the agent to choose between. Meanwhile, state machines can produce the same agent framework in a much more simple manner. The simplicity in agent framework allows for the modeler to reduce time required for model development and improve the traceability of the decision-making within an agent by following the state an agent chooses, as opposed to reviewing consecutive evaluations of a decision tree.

The comparison of the decision tree and state machine agent frameworks already favors the state machine due to its simplicity and extensibility. However, the degree of design space exploration must also be addressed. The decision tree framework requires specific paths to be chosen in order to 'unlock' certain decisions. For example, if an agent using the decision tree framework from Figure 3.10 wanted to evade a defender in the middle of an engagement, it simply could not because it does not have that choice. A counter argument would say that the agent framework could just be expanded to include more branches, but at some point the decision tree will become too complex and include too many repeated branches to provide an efficient agent framework. In comparison, the state machine has the freedom to transition to any other defined state from its current state. This agent framework can also define additional states without a significant increase in complexity. In conclusion, the state machine and decision tree may have similar potential to explore the mission action design space, but the state machine provides a more simple framework to formulate while providing a more concise way to trace agent decisions.

The BDI agent framework makes up for its lack of structure with its ability to formulated agents with intelligent behavior. This agent framework seems to excel in models that require simulation of human behavior. Some of the state machine frameworks sought to capture human decision-making through stochasticity, but may still be orders of magni-

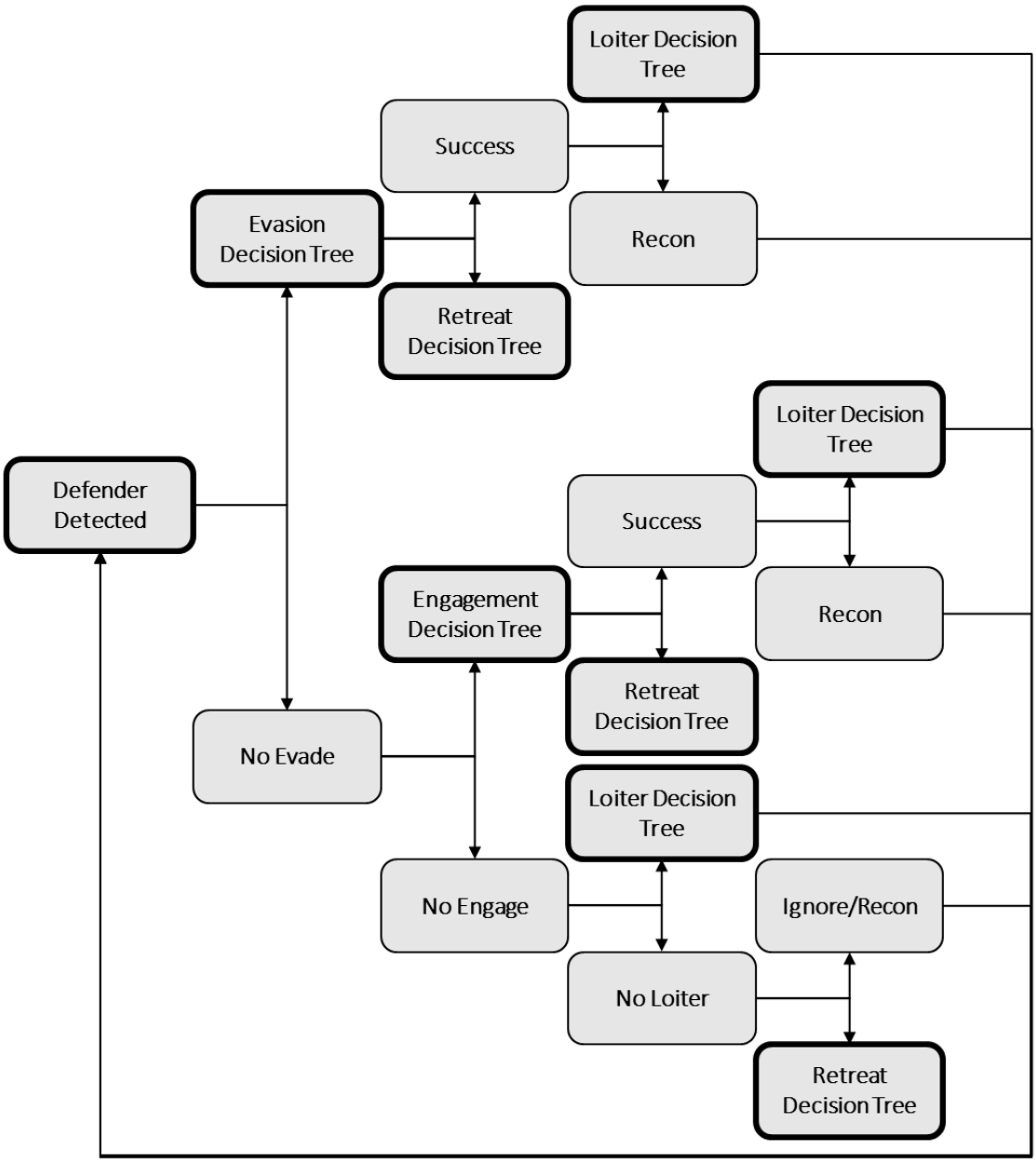

Figure 3.10: Decision tree framework of an agent for a contested reconnaissance mission.

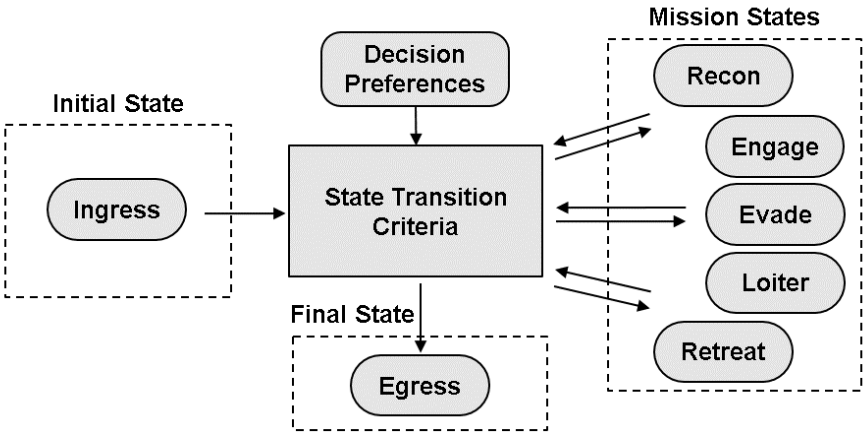

Figure 3.11: State machine framework of an agent for a contested reconnaissance mission.

tude behind the BDI framework. However, modeling human decision-making or behavior can be complex due to issues modeling soft factors such as subjective choices or irrational behavior, often resulting in only qualitative results [99]. When investigating autonomous vehicles, human decision-making may not be necessary to model. For unmanned aerial technology, agents could be viewed as fully autonomous vehicles that have to conduct their own decision-making in the field. Although this viewpoint may not capture how a human would react in a given scenario, it could inform a human operator on the optimal tactics that should be chosen in a given scenario.

The comparison of state machines and the BDI framework can also be made in the context of mission action design space exploration. Several distinctions should be made first in order to properly compare the two agent frameworks. State machines can be decomposed into two types of behaviors: decision behaviors and state behaviors. Decision behaviors are comprised of the decision-making preferences of an agent to transition from one state to another. The individual state behaviors are states in which the possible actions are unique to that state. The combination of decision and state behaviors over an entire mission will produce the collective mission behavior of a single agent. The collective mission behavior is what the BDI and OODA loop frameworks seek to model. These three classifications of agent behavior are summarized below and illustrated in Figure 3.12.

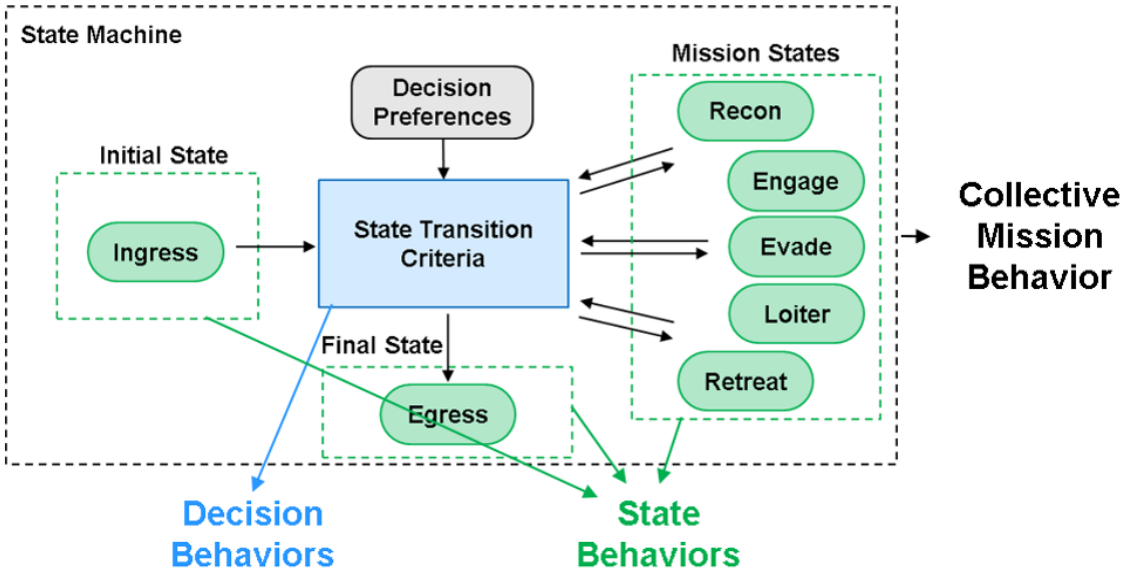

Figure 3.12: Three classifications of agent behaviors with a state machine agent framework.

- Decision Behavior: discrete decisions each agent makes to transition states throughout a mission
- State Behavior: behavior of a given state (i.e. ignore, engage, evade, or reconnaissance)
- Collective Mission Behavior: combined set of decision and state behaviors that produce an collective mission tactic for a system

The collective mission behavior of an agent is the desired outcome for mission action design space exploration; however, the BDI agent framework lacks the ability to produce believable results due to the absence of traceable behavior. The introduction of decision and state behaviors provides a way to trace the behavior of an agent throughout a mission simulation. Although complex and difficult to trace, intelligent agent frameworks could enable more exploration of the design space due to the lack of decision and state behavior definitions. The improvement the BDI framework has in design space exploration sacrifices traceability and simplicity. Technology evaluation methodologies may seek to model many technologies that each have a unique collective mission behavior. The complexity of for-

mulating a BDI agent for each technology would be prohibitive on the overall technology evaluation methodology. Alternatively, a single state machine framework could include a wide variety of technology specific states without a significant increase in agent development. Even with a simple BDI agent framework, the traceability of an agent's decision and state behaviors are confounded in the collective mission behavior.

The OODA loop has several contributions that could be made to enhance an agent's framework, but it may not be a complete stand-alone framework. The OODA loop lacks structure for its decision and action steps, but could be a useful tool when combined with a structured decision making tool. Several aspects of the OODA loop seem to overlap with the more well defined POMDP. The concepts behind the OODA loop meet the requirements of a general agent framework with traceability, but lacks structure for ABM. POMDP provides a similar concept as the OODA loop by using observations to orient and decide what action to take; however, it provides structure for the agent's framework where the OODA loop does not. Concepts from the OODA loop could be incorporated into the general state machine agent framework to provide a more appropriate framework for the agent's formulation. Such a framework, illustrated in Figure 3.13, would incorporate observation and orientation steps during the agent's execution of its decision behavior. The state machine framework also provides structure to the decision and action steps of the OODA loop when combined into a single agent framework.

Mission action design space exploration needs an agent framework that can provide traceability, such as the state machine framework, but also not limit the mission action space. Based on the conclusions drawn from literature, a combination of the frameworks may lead to the optimal agent framework. State machines are particularly notable due to the simplicity of their framework combined with their extensibility. Conversely, observations made by the agent provide critical information for uncertain mission environments. Both the OODA Loop and POMDPs provided insight into the development of the appropriate agent framework for mission action design space exploration. A state machine that

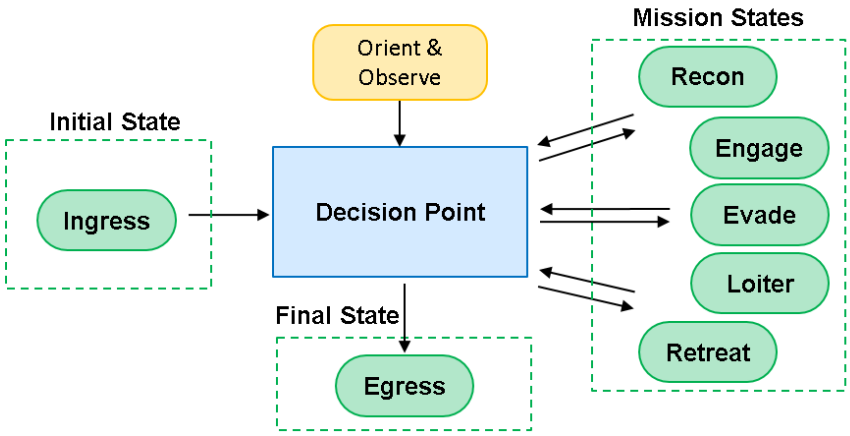

Figure 3.13: A state machine agent framework incorporating OODA loop concepts.

incorporates observations like the OODA loop or POMDPs would enable an open mission action design space while maintaining traceability of the agent's decision and state behaviors. This observation is stated below.

***Observation**: The use of an observation-based state machine to formulate an agent provides a traceable framework for the exploration of the mission action design space.*

3.2 The Decision Behavior Approach

The observation-based state machine framework enables the discretization of the mission action design space into decision and state behaviors. This unique discretization of an agent's behavior provides a framework to conduct mission action design space exploration of a minimally defined mission. This agent framework – *the decision behavior approach* – enables traceability of mission tactics within a minimally defined mission. Implementation of the decision behavior approach requires further definition of the decision and state behaviors of the agent.

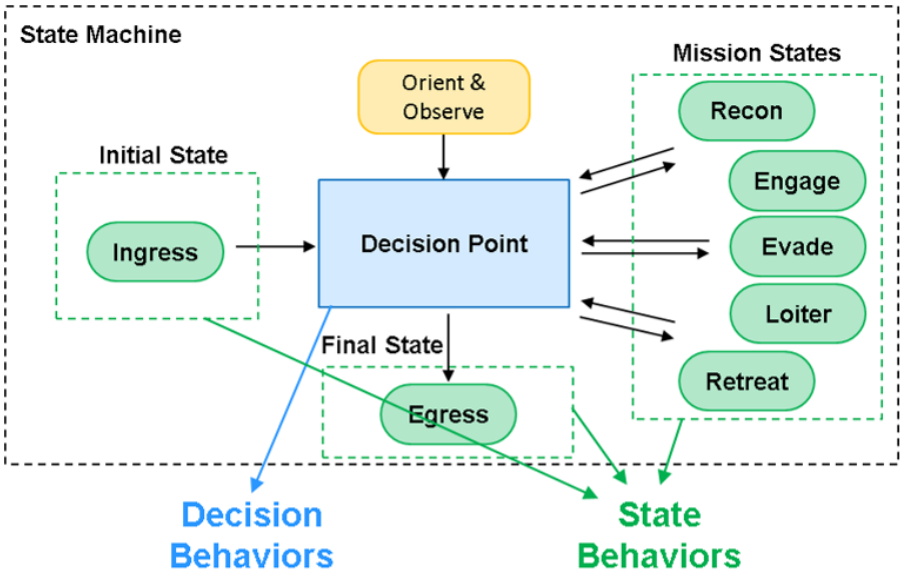

Figure 3.14: An agent's state machine framework made up of decision and state behaviors.

3.2.1 Determining Agent Behavior

Traditional tactics formulation for an agent typically involves SMEs defining routes, maneuvers, actions, etc. for the model to simulate [7, 72]. The use of SMEs to develop an agent framework is necessary to identify the correct capabilities of each technology under investigation; however, SMEs should not dictate the actual tactics an agent should take within a given simulation for a given technology. A line should be drawn between the use of SMEs developing an agent framework and formulating agent tactics to prevent bias in the later.

The formulation of tactics seeks to identify and define the agent's behaviors necessary to conduct mission action design space exploration. Introduction of the decision behavior approach discretizes the mission into critical decision points and corresponding state behaviors to choose at each point. Further definition of tactics formulation involves defining an agent's decision behavior and state behaviors. These two types of behavior within an agent's state machine framework, highlighted for clarity in Figure 3.14, need further definition to enable tactics exploration.

State behaviors, such as the reconnaissance or evasion states, have been studied exten-

sively in literature. Optimal control algorithms exist that can find the optimal reconnaissance flight path or the optimal evasion maneuver [100, 101]. These algorithms could be one solution used for the defined state behaviors. Alternatively, the state behaviors could be defined by their own learning algorithm, such as reinforcement learning which has been successfully implemented for wildfire surveillance [102]. The state behaviors included in the contested reconnaissance mission could be defined using existing algorithms in literature. Based on these findings, the state behaviors of an agent's state machine framework seem to be understood and may not require further research.

The decision behaviors of an agent are not as well understood or easily implemented in ABM. The SAA, discussed in Section 2.2.4, introduced ABVs to enable decision-making in the design space. These variables seek to turn decision-making spaces that are discrete or categorical into continuous ones. For example, fixed binary decision-making parameters such as those seen in decision trees, can be converted to continuous ones by introducing probability to the decision-making process [57]. Although ABVs succeeded in bridging the gap between discrete and continuous design variables, further research is required to implement optimal decision making for a minimally defined mission.

The SAA produces either a surrogate model of the mission action design space or tuned agents that have preferential decisions defined by ABVs for further use in the M&S environment [57]. The surrogate model may have use in higher-level mission simulation, but loses some traceability of tactics. In order to maintain traceability, the tuned agents produced by the SAA could be used to define an agent's decision behavior; however, these agents lack the ability to change their ABVs – or decision preferences – during a mission simulation which may result in suboptimal behavior. The fixed preferences that an agent formulated with ABVs for its decision-making behavior would limit the decision space as the mission environment changes. Further formulation of an agent's decision behavior needs to be conducted to develop a more general decision-making behavior. The desired formulation would be able to change its decision-making behavior based on information

about the agent's current state.

The first step in formulating a decision behavior is to define a general framework that can allow the agent to alter its decision preferences to changes in its environment. This agent formulation should be of a functional form to allow for observed stimuli to impact the agent's decisions. In order to determine the functional form of the decision behavior function, an algorithm must be formulated that can solve for the optimal behavior. From this perspective, a decision behavior algorithm can be defined as an agent's internal decision-making preferences that are a function of stimuli from its environment. In the contested reconnaissance mission, an agent's preference to evade or engage a defender would change as additional defenders are encountered. This one example justifies the need for a decision behavior algorithm rather than ABVs. However, in order to formulate a decision behavior, research into optimal decision-making algorithms may be necessary. Defining an agent's decision behavior as an algorithm for a minimally defined mission could provide further exploration of the mission action design space. The formulation of a decision behavior algorithm is necessary to enable the exploration of the mission action design space. This research finding is stated below.

Research Question 1*: How can the formulation of a decision behavior algorithm efficiently balance optimality of a minimally defined mission?*

3.2.2 Defining a Decision Behavior Algorithm

The notion of a decision behavior algorithm has been indirectly studied in literature through the concepts of decision theory and game theory. Decision theory is a class of formal frameworks used to study decision-making problems. These frameworks provide a way to

study optimal and sub-optimal decision-making behavior [103]. Decision theory is particularly useful for decision-making in ABMs that have an unpredictable environment [104]. Many decision theories have been studied and implemented as decision-making frameworks [105]. Similar to decision theory is game theory, which is focused on studying the interaction of agents that pursue contradicting goals [104].

Both decision theory and game theory are useful concepts to understand when formulating the proper framework of a decision behavior algorithm for an agent. The combination of these concepts with early frameworks from Artificial Intelligence (AI) have been successful in enhancing the decision theory framework [106]. Several frameworks found in ABMS have also been influenced by decision theory concepts [82, 107]. Specifically, Reinforcement Learning (RL) and Genetic Algorithms (GA). Each algorithm type must be researched further to determine the impact each one may have on the solution space.

Reinforcement Learning

RL focuses on algorithms that map stimuli to actions [82]. This type of learning algorithm is neither supervised nor unsupervised learning. At first glance, RL is similar to unsupervised learning in that it learns as it interacts with the environment, but its purpose is quite different. RL is utilized when the agent must discover the optimal behavior through rewards in an environment where the rules are unknown. This discovery is conducted through trial-and-error for model-free RL algorithms [103]. A standard agent using an RL reward scheme is illustrated in Figure 3.15.

There are four key components to RL which are the policy, reward signal, value function, and model of the environment. The policy defines the behavior of the learning agent. The reward signal defines the immediate reward of a given action. The value function defines the overall sense of success which defines a delayed reward feature unique to RL. The final component of RL in the model of the environment. These four components of RL along with trial-and-error and delayed reward features, define the core elements of RL al-

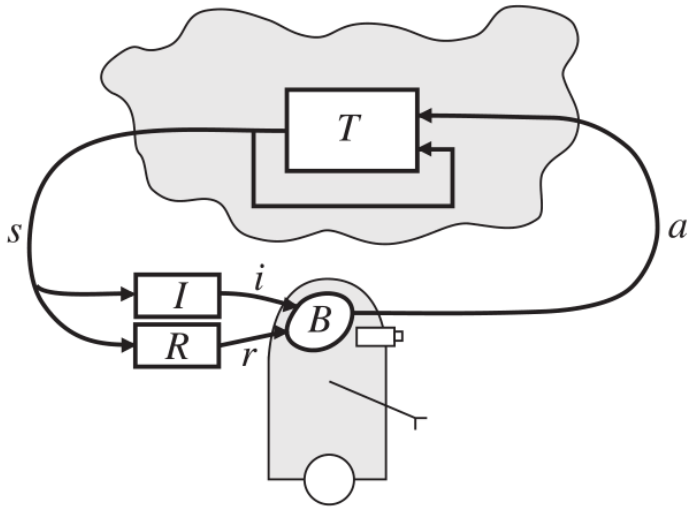

Figure 3.15: Interaction of an agent and its environment [82].

Table 3.1: Example agent dialogue of a model-free RL algorithm [108].

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.
⋮	⋮

gorithms [82]. An example dialogue to aid in the understanding of a RL is shown in Table 3.1. Such a dialogue may aid in the understanding of how the model-free RL algorithm learns from trial-and-error.

RL has been applied to a variety of problems such as the games of Chess, Shogi, and Go where an RL algorithm proved highly successful against specialized RL algorithms that have bested the best human players [109]. The RL algorithm has also had success in ABM of robotic soccer tournaments and adaptive game AI in video games [110, 111].

RL has proven successful in many fields, but still has challenges that must be considered

when formulating an RL algorithm. The trade-off between exploration and exploitation is a common challenge of RL. Exploration is the decision to test new actions with unknown results while exploitation is the decision to leverage past decisions that provided a high reward [82]. Neither exploration or exploitation can be pursued exclusively. The two methods of searching must be simultaneously pursued in order to develop a RL algorithm of good quality. With this challenge in mind, RL still seems to be a powerful decision-making algorithm for mission action design space exploration.

Genetic Algorithms

Evolutionary Algorithms are a class of guided random search techniques that include genetic programming, evolutionary strategies, evolutionary programming, and GAs. This class of algorithm operates on a population of individuals, where each individual represents a possible solution [112]. Both evolutionary strategies and GAs have been shown to rival RL algorithms [113, 114]. GAs are of particular interest because they are a gradient-free search technique, unlike evolutionary strategies and RL algorithms, such as Q-learning or policy gradient, which are gradient search techniques that use a gradient approximation and backpropagation, respectively. The competitiveness of GAs when compared to gradient-based algorithms may suggest that a gradient-free search technique could be better suited for some search spaces [113]. The inclusion of GAs as a possible method for the decision behavior algorithm provide a gradient-free method of sampling the search space.

The basis of the GA method utilizes genetics to aid in problem solving. The population that GAs operate on are comprised of a set of individuals or chromosomes. Each chromosome contains genes that define an abstract representation of the solution. The formulation of genes can be done a number of ways to capture the entire solution space, such as binary encoding [112]. Once the gene structure is defined, a population of chromosomes is typically generated randomly. The algorithm then follows the following process steps of selection, reproduction, evaluation, and replacement.

The GA algorithm starts by evaluating the fitness of each individual in the population. These fitness values enable the ranking of each individual to determine which will be chosen as parents for reproduction during the selection phase. The reproduction phase sees the generation of new chromosomes through the crossover or mutation of parent chromosomes. The newly generated chromosomes are then evaluated to determine their fitness during the evaluation phase. The set of parent and newly generated chromosomes will then replace the old population and return to the selection phase for the next generation of the algorithm.

Some advantages of GAs include their simplicity, which enables them to be used on many different optimization problems, and their stochasticity during the selection and reproduction phases which aid in finding the global optimum [112]. However, the randomness associated with GAs results in a large number of required function evaluations when determining the fitness of each solution within each population for each generation. The algorithm's hyperparameters like population size, mutation rate, crossover rate, and selection can also have a large impact in finding the global optimum and may be one of their main limitations [115]. Overall, GAs provide a robust method to formulate a decision behavior algorithm. Their gradient-free search technique could prove successful in finding the global optimum of a decision behavior algorithm since the shape of the search space is unknown.

Evaluation of Decision Behavior Algorithms

The desired decision behavior algorithm for an agent is one that maximizes optimality of the agent's learned decision behavior without constraining the minimally defined mission. Traceability is also key in mission action design space exploration due to the desire to produce believable tactics that are a culmination of traceable actions. A decision behavior algorithm that can determine optimal decision preferences given a minimally defined mission is desired to conduct tactics exploration because of the unknown shape and complexity

of the design space. Both RL algorithms and GAs place no additional constraints on the decision space throughout the mission simulation and maintain traceability of the decisions made.

The implementation of each algorithm for a minimally defined mission must be understood. An algorithm that does not artificially favor one tactic over another and has a wide applicability to many types of missions is desired. Both algorithms meet the general applicability desired due to their board range of possible state inputs, but the reward schemes of each must be carefully defined. In order to avoid any bias in the decision behavior algorithm, the rewards for each state behavior should be carefully defined as to not favor any single tactic on the baseline vehicle. In the case of RL and GAs, the reward schemes can be defined in many different ways for each. Since both algorithms have broad applicability across mission types and have no fixed reward scheme, each qualifies as a valid solution for the decision behavior algorithm framework. However, it will depend upon the user to craft the algorithm's reward scheme to prevent bias.

Overall, both algorithms seem to meet the desired characteristics of a decision behavior algorithm. Ensuring traceability while providing a method to explore an open SoS solution space are satisfied by both. However, the method of which the algorithm samples and learns optimal decision behavior may lead to a more appropriate choice to use for a decision behavior algorithm. Both RL and GA have some degree of random search to help ensure the global optimum is found, but RL provides techniques to take advantage of it's learned knowledge and apply it to future searches. A unique decision behavior algorithm likely must be trained for each technology of interest, placing some degree of computational requirements on the exploration of the design space. Utilizing an algorithm that can take advantage of learning techniques to speed up its convergence on the global optimum could substantially reduce computational cost. RL provides a more intelligent search algorithm when compared to a GA's random search that can explore the mission action design space in a more informed way. This hypothesis is stated below.

***Hypothesis 1**: The use of reinforcement learning to formulate the decision behavior algorithm could balance optimality and sampling efficiency of a minimally defined mission.*

3.3 Mission Action Design Space Exploration

The size of the mission action design space may not be an issue if it is the only design space of concern. However, technology evaluation methodologies are concerned with the mission action design space for each technology, the mission architecture design space, and SoS interactions. The desire to investigate multiple design spaces requires that the parameters and their respective ranges within each design space be limited in size to reduce the overall dimensionality of the problem.

Research into the previous gap sought to develop a framework that enabled the exploration of the mission action design space for a minimally defined mission. However, road blocks still exist in the development of a decision behavior algorithm that can efficiently explore the mission action design space. The dimensionality of a completely open design space may be a drawback of this approach due to the computational cost required to properly explore the design space. This methodology will define the decision-state space as the mapping of the agent's state, defined by state variables and imperfect observations, to each state behavior. Similarly, the state-action space is the mapping of an agent's state with each action, such as flight control changes. As the amount of information used to define the agent's state increases and the number of actions increases, the dimensionality exponentially increases.

In a reconnaissance mission, the decision-state space could consist of continuous state variables and multiple observations images. The inputs alone define an almost limitless

decision-state space without accounting for the number of state behaviors available to the agent. The state-action space is also immense due to the temporal selection of actions. The dimensionality of both the decision-state and state-action space for a decision behavior algorithm must be addressed to ensure that it can scale with the complexity of the mission action design space. This research question is presented below.

***Research Question 2**: How can the decision behavior approach capture the mission action design space as its complexity increases?*

The complexity of the mission action design space should be explored from three directions. First, the complexity of mapping the decision-state and state-action space must be examined. Second, the number of state behaviors must be chosen in such a way as to not limit the decision-state space, but enable the reduction in dimensionality of the mission. The last aspect involves probing multi-agent solution spaces. These three avenues can each be pursued independent of each other which allows Research Question 2 to be broken down into three sub questions that can help answer the overarching question of scaling decision behavior algorithms as the complexity of the mission action design space increases.

3.3.1 The Complexity of the Decision-State Space

The state definition of an agent can vary in complexity depending on the amount of data or information required to capture the both the decision-state and state-action spaces. The mapping of the state-action space is studied and implemented by many in literature [116, 102]. Similar implementations to the decision-state space are unknown. The decision-state space is the mapping of unique states that the agent could have paired with the ranked set of state behaviors the agent believes would provide the best reward in the current state.

Each unique state is defined by perfect and imperfect information. The inclusion of continuous state variables significantly increases the dimensionality of the decision-state space. The increase in dimensionality presents a problem linked to computational cost; however, increasing computational resources is not a solution that can capture observations and imperfect information of an agent's state. POMDPs introduce observations of the agent's state into the overall decision-state space. These observations may not easily be captured by a tabular decision-state space. A method is needed to capture the complexity of POMDPs while also managing the dimensionality of the decision-state space. These findings lead to research question 2.1.

Research Question 2.1*: How can a decision behavior algorithm capture the complex decision-state space?*

Two methods can be investigated to define the mapping of both the decision-state space. The first method is a tabular list that pairs each input with an an optimal output. A tabular mapping or policy table enables a simple and easily programmable framework for developing the decision behavior algorithm. Alternatively, an Artificial Neural Network (ANN) could provide a more abstract mapping that could incorporate more complex state information without a significant increase in computational cost. Both a policy table and ANNs will be investigated to understand their ability to map the complex decision-state space.

The Policy Table

The traditional state-action table is a mapping of each unique state to its optimal action or ranked actions. Both RL and GAs are setup to operate on such a table to find the optimal mapping or policy. The Q-Learning RL algorithm has been used by many in literature

[117, 118, 119]. The Q-table is a table of Q values for each unique decision-state pair. This algorithm determines the expected reward or Q value for each action given a unique state. Once the algorithm is trained, the Q-table informs the agent which state behavior will result the highest expected reward given its current state.

Similarly to Q-Learning, the genes of a genetic algorithm also inform the action of the optimal action given its current state. Genes can be defined and encoded in many ways when using a GA. This research could define a gene as a list of integers, where each index of the list corresponds to a unique sate while the integer value corresponds to optimal action. The algorithm would then optimize the string of state-action pairs to find the optimal action for every state.

Artificial Neural Networks

ANNs are methods that have been used extensively in ABM, decision-making, surrogate modeling, image recognition, and many other fields [57, 120, 121, 122]. The structure of the model is comprised of many artificial neurons that link inputs to outputs based on their connections within the hidden layers of the algorithm. The hidden layer, illustrated in Figure 3.16, is comprised of artificial neurons that link to neurons in the next hidden layer or output layer. Each ANN can have any number of hidden layers depending on the complexity of the model desired. The artificial neurons are simple processing units that are interconnected in an intentional way [121]. Each connection has an associated weight that determines its impact on its neighboring neurons. The weights of each neuron are adjusted during learning.

The ANN would replace a decision-state table by providing a complex network mapping state inputs to the optimal state behavior output. The immediate advantage of using these models would be the ability to include image observations for the agent's state information such as those implemented in [102]. One drawback of ANN is the difficulty producing a satisfactory fit without overfitting. Overfitting the ANN is easy to do and may

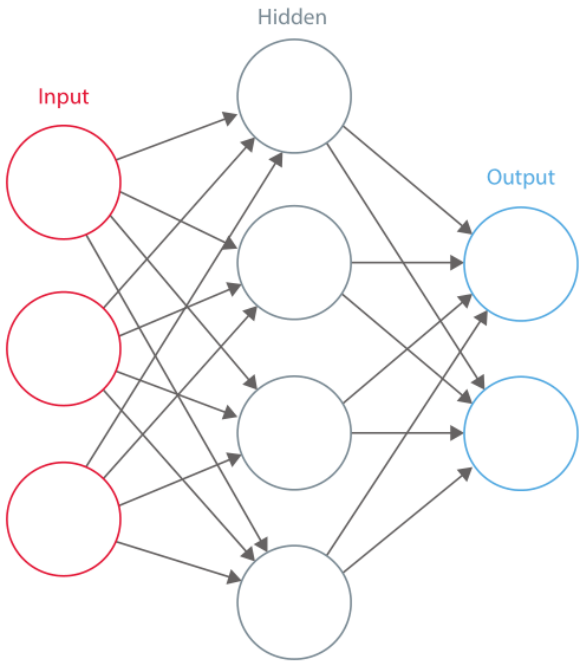

Figure 3.16: Structure of a neural network [122].

be difficult to spot. Early methods to prevent overfitting and underfitting the model focused on selecting the optimal number of neurons and hidden layers [123]. Overfitting can also be reduced by ensuring the appropriate number of samples are collected before each training update [102].

The Decision Space Framework

The dimensionality of the decision-state space can become prohibitive as the complexity of state definitions increases through the incorporation of imperfect information and agent observations. A policy table could be used as a simplistic method to capture the optimal decisions the agent should make without addressing the overfitting concerns of ANNs. However, mapping a continuous decision-state space to a discrete set of state behaviors can be difficult due to the dimensionality of the space. On the other hand, ANNs have the advantage of mapping complex and continuous inputs that could include imperfect information and observations made by the agent. Based on these findings, ANNs seem to be a better fit for the decision behavior algorithm. These models could provide the algorithm

to scale with the complex decision-state space while also enabling agent observations and imperfect state information. These findings inform Hypothesis 2.1.

Hypothesis 2.1*: The use of a Neural Network to approximate the algorithm's immense decision-state space enables the exploration of the mission action space.*

3.3.2 The Dimensionality of the Decision-State Space

The dimensionality of the mission action design space must be reduced without imposing limits on the tactical options or defining mission paths. The second avenue to pursue in assessing the dimensionality of the decision behavior algorithm is the size of the decision-state space. The dimensionality of the decision-state space should be investigated to quantify or reduce the size of the mission action design space while maintaining a minimally defined mission framework. The reduction in available state behaviors would not only decrease the decision-state space for the algorithm to choose from for each technology, but also reduce the number of state behaviors that must be trained for each technology. These findings lead to research question 2.2.

Research Question 2.2*: How can the selection of state behaviors decrease the dimensionality of the decision-state space without constraining the mission action design space?*

The state machine agent framework has the flexibility to suit a variety of possible states. Many states could be modeled without increasing the complexity of the state machine sig-

nificantly. However, not all states may provide benefits to the simulation. Some states may not even be used by an agent or have a very low probability of being chosen. In those cases, an argument could be made to filter out those states which do not provide a significant contribution to the agent's performance. Filtering out insignificant or unsuitable states in the agent's state machine could simplify the agent's decision behavior algorithm by requiring less state combinations to be sampled. Both the identification of incompatible and unsuitable state behaviors should be investigated.

Identification of Insignificant State Behaviors

One method used to identify insignificant states in an agent's state machine is a "screening test." Screening of experiments can identify important factors. One such technique is the Analysis of Variance (ANOVA) which is a statistical method used to identify significant effects of an experiment [124]. The combination of ANOVA and a Design of Experiments (DoE) can be used to determine the statistical significance of parameters on the variability of the results [7].

One method that can be used to visualize the results of ANOVA is the Pareto Chart, illustrated in Figure 3.17. The Pareto Chart was named after Vilfredo Pareto who developed the "80/20 rule" from observations of the distribution of Italian income among the population. Pareto observed that 20% of the Italian population received 80% of the income. Juran later generalized the 80/20 rule for manufacturing defects in the field of quality assurance and described it as the "vital few and trivial many" [125]. This distribution of effects is visualized through a Pareto Chart. Figure 3.17 illustrates a generic Pareto Chart which shows four variables contributing to 80% of the response. Although Pareto's decision to split the distribution at 80/20 may be arbitrary, his choice does provide a starting point for determining what constitutes an insignificant state.

The Pareto Chart has since been applied to the field of conceptual design to determine the impact of technologies or design parameters on the overall objective function [7]. The

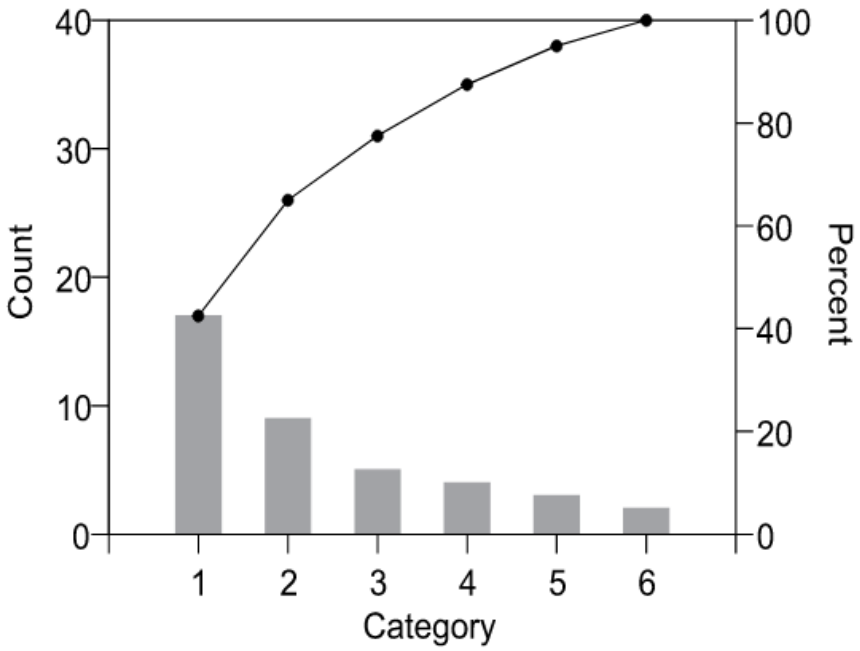

Figure 3.17: [125].

CBD for SoS methodology, discussed in Section 2.1.3, used ANOVA and a Pareto Chart to determine the impact design parameters had on a mission campaign. Figure 3.18 illustrates the Pareto Charts for the contribution each design parameter has on the variability of the response.

Note that the highlighted area in Figure 3.18 outlines 80% of the contribution to the response variability. For the response of Time Critical Targets (TCT) killed, design parameters below the 80% threshold could be removed to reduce dimensionality; however, some of the parameters that are below the 80% threshold for one response, may be vital for another response. For example, TCT dwell time is the largest contributer to the TCT killed response, but contributes almost no variability to the supersonic platforms lost response. A similar issue could occur in technology evaluation in the mission action design space. Some states defined in the agent's state machine may impact the mission effectiveness of one technology while having little impact on another technology. These considerations must be addressed when choosing which states to remove from the agent's state machine framework.

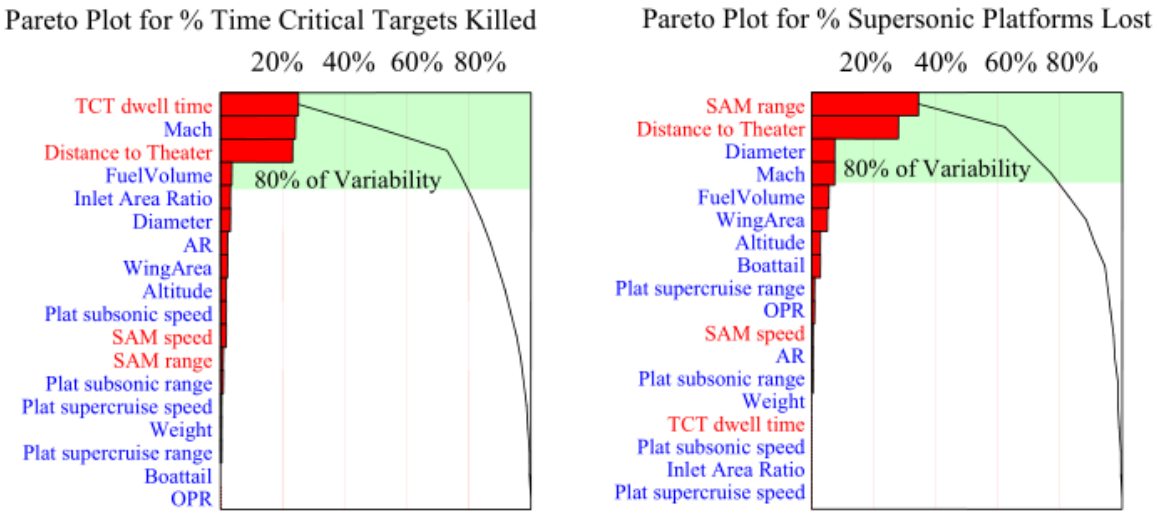

Figure 3.18: Pareto Chart for two responses from a mission campaign [7].

Identification of Unsuitable State Behaviors

The definition of state behaviors should be a broad procedure to capture the mission action design space. However, not all state behaviors should be included for each technology. Each state behavior should capture a certain aspect of the mission, such as an engagement state behavior, while avoiding targeted behaviors for specific technologies. Once state behaviors are identified, then each should be reviewed to determine if they are suitable for a given technology. However, a method to determine the suitability of each behavior for each technology must be determined to minimize the introduction of bias during tactics formulation.

The TIES for SoS methodology utilizes a Technology Compatibility Matrix (TCM) to identify compatible and incompatible technologies when conducting a DoE. This matrix compares the compatibility of each technology with another technology. The identification of incompatible technologies does not just eliminate one case from the DoE, but eliminates all technology combinations that contain both of the incompatible technologies. These incompatible technologies are those that are not realistically possible due to current physical constraints [25]. Figure 3.19 illustrates a TCM of 11 technologies for a high speed civil

transport. A full factorial DoE results in $2,048$ combinations. However, the identification of 12 incompatibilities reduces the number of possible combinations to 272.

Technology Compatibility Matrix of a High Speed Civil Transport (1:compatible, 0:incompatible)	Composite Wing T1	Composite Fuselage T2	Circulation Control T3	HLFC T4	Environmental Engines T5	Flight Deck Systems T6	Propulsion Materials T7	Integrally, Stiffened Aluminum Airframe Structures (wing) T8	Smart Wing Structures (Active Aeroelastic Control) T9	Active Flow Control T10	Acoustic Control T11
Composite Wing	1	1	1	0	1	1	1	0	0	0	0
Composite Fuselage		1	1	1	1	1	1	1	1	1	1
Circulation Control			1	1	1	1	1	1	1	1	1
HLFC				1	1	1	1	0	0	0	1
Environmental Engines					1	1	1	1	1	1	0
Flight Deck Systems						1	1	1	0	1	1
Propulsion Materials							1	0	1	1	1
Integrally, Stiffened Aluminum Airframe Structures (wing)								1	0	1	1
Smart Wing Structures (Active Aeroelastic Control)									1	1	1
Active Flow Control										1	1
Acoustic Control											1

Symmetric Matrix

Figure 3.19: Technology compatibility matrix for a high speed civil transport [25].

The TCM is a valuable tool to use during technology evaluation studies to reduce the dimensionality of the concept space. A similar tool could be developed for technology and state behavior combinations to possibly reduce the dimensionality as well. The suitability of a state behavior for each technology is more appropriate than determining compatibility due to the general behavior defined by each state behavior. A state behavior may be compatible for all technologies, but its use with some may not be suitable for that type

of technology. The engagement state behavior in the contested reconnaissance mission is compatible with all technologies, but may not be suitable for a bundle of endurance technologies that aim to increase the time the aircraft is flying. Similarly, the concept space may include offensive and defensive platforms that may or may not be suitable for an engagement behavior. The implementation of a matrix that can map the suitability of each state behavior to each technology could address the dimensionality of the decision-state space.

Reduction of State Behaviors

The use of ANOVA and a Pareto Chart may provide a method to identify insignificant state behaviors in an agent's state machine framework to reduce the dimensionality of the decision-state space; however, its use would require the training of each state behavior for each technology. A reduction in the number of state behaviors before training an agent with each technology alternative may reduce the overall complexity of the methodology by eliminating state behaviors for technology alternatives. The determination of a state behavior's suitability for each technology alternative could provide a more practical reduction in the decision-state space while also reducing the number of state behavior algorithms required to train. Technology alternatives that are not suitable for some state behaviors could reduce the dimensionality of the decision-state space for those alternatives. The development of a matrix that maps each technology's suitability with each state behavior could address the dimensionality of the decision-state space. The identification of unsuitable state behaviors would simplify the decision behavior algorithm and maintain a minimally defined mission. This hypothesis is stated below.

Hypothesis 2.2*: If a state behavior's suitability is determined for each technology with unsuitable behaviors removed, then the dimensionality of the decision behavior algorithm would be reduced.*

3.4 Exploration of Tactics

The third step of the proposed methodology seeks to train the agents using the decision behavior approach and disturb the ways the agent uses each technology alternative. The agent's logic must be redefined for an algorithm based approach to incorporate training methods for both the state behaviors and the decision behavior. The order in which each behavior is trained is important to define for the methodology. Once the order is defined, the decision behavior approach will be investigated for cooperative multi-agent simulations. The incorporation of cooperative agents expands the mission action design space significantly to enable more tactics exploration. Further definition of the agent's algorithm will also be addressed to enable the next step in the methodology to conduct tactics exploration.

3.4.1 Training the Decision Behavior Agent

The inclusion of a learning algorithm in the decision behavior approach requires the agent's logic to be revised for the state behaviors and the decision behavior. Revision of each state behavior logic must be addressed first to ensure each behavior is trained independently in a manner that does not limit its respective mission action design space. The general procedure for training the decision behavior algorithm requires each state behavior's algorithm to be trained or logic defined. Since state behaviors can be defined multiple ways, their

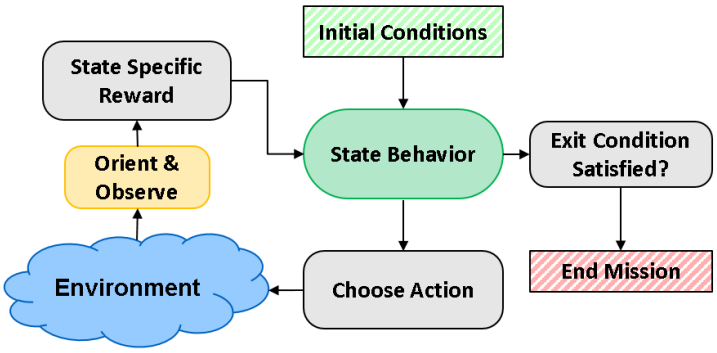

Figure 3.20: Generic agent logic for implementation of a state behavior.

training logic should be defined first.

State behavior logic could be defined as a set of logical tasks that define dropping a supply payload or firing a missile or by an algorithm that defines a trained behavior. The general formulation of an agent's state behavior defines a mission segment that represents the tasks and situations the agent is expected to encounter in the higher level mission of interest. This mission segment should encompass all variable states the agent is expected to encounter during the execution of that state behavior. The generic formulation of an agent's state behavior logic is illustrated in Figure 3.20.

The agent's state behavior logic is structured similarly to a POMDP framework. The agent's state behavior is given a set of initial conditions. The agent then uses the initial information to choose an action and takes that action in the environment. The environment reacts to the action from the agent and changes accordingly. The new environment is then oriented to the agent's frame of reference and observed to determine the state specific reward to receive for taking the initial action. This logic loop is continued until the exit criteria is satisfied. The state behavior implements this agent logic to train the algorithm by sampling the environment over many mission simulations. For state behaviors that do not require an algorithm, the agent's logic does not need to be sampled once it has been defined.

Once each state behavior is defined and each corresponding algorithm is trained, the

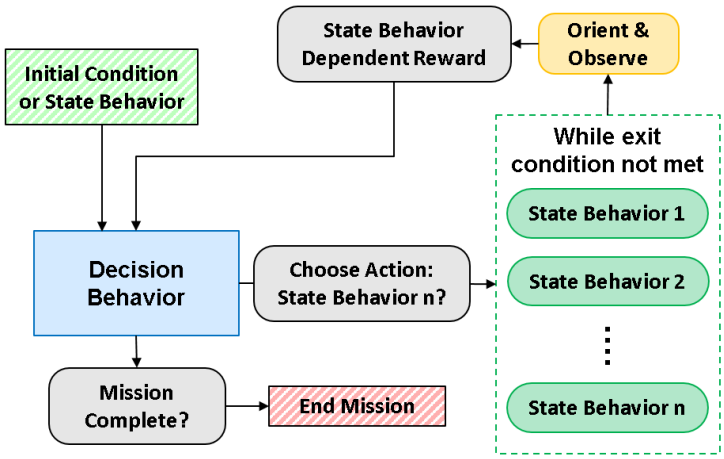

Figure 3.21: Generic agent logic for implementation of its decision behavior algorithm.

decision behavior logic and training can begin. The decision behavior logic defines the agent's logic for the entire mission. Care must be taken in defining the agent's decision behavior logic to ensure it maintains a minimally defined mission. The generic decision behavior logic, illustrated in Figure 3.21, does not explicitly define an initial condition. The initial condition of the agent could be defined as either an initial starting condition or an initial state behavior. Once the mission begins, the decision behavior logic proceeds similarly to the state behavior logic; however, the decision behavior does not directly interact with the environment every time step, but instead receives its reward from the change in the environment between when the chosen state behavior began and ended its tasks. The decision behavior uses its observations of the environment and continuous state variables to determine the algorithm's inputs for the next decision.

The order of training each behavior is necessary to define due to the dependency of the decision behavior on each state behavior. The state behavior's logic and decision behavior logic can be defined together to form the collective mission logic, illustrated in Figure 3.22; however, each state behavior algorithm needs to be trained before the decision behavior algorithm. Each state behavior should remain independent from each other to ensure the algorithm's can be independently trained. Once trained, the state behaviors are used during training of the decision behavior algorithm. The decision behavior algorithm samples the

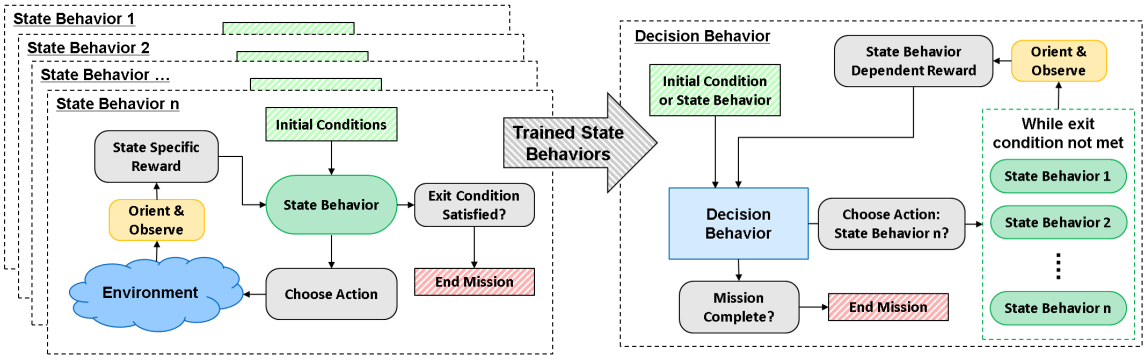

Figure 3.22: Generic implementation of an agent's state behavior algorithms with its decision behavior algorithm.

decision-state space by selecting a state behavior and receiving a corresponding reward once the state behavior has met its specific exit criteria. The entire mission is simulated during each training episode of the decision behavior algorithm, which results in a set of decisions made at each decision point, a set of rewards that resulted from each decision, and a final reward that quantifies the reward for a single mission execution.

The agent's training logic provides a framework to define and train state and decision algorithms for a broad set of missions. However, the implementation of the decision behavior approach for multi-agent missions is unknown. Cooperative multi-agent tactics are difficult to model due to the potential complexity of interactive agent behavior and the ever increasing dimensionality of the model. In order to truly explore an open mission action design space, cooperative, competitive, and mixed multi-agent systems should be considered. This research will investigate implementing agents with a decision behavior algorithm in a cooperative environment. It is suspected that the methods to implement the decision behavior algorithms for a cooperative multi-agent system can be extended to competitive and mixed multi-agent systems. Once a cooperative environment is developed, the quantification of mission effectiveness of a multi-agent system will be discussed. The findings in this section will ultimately enable further exploration of the mission action design space.

3.4.2 Cooperative Decision Behavior Algorithm

Cooperative multi-agent systems have been investigated by many in literature [126, 127, 119, 128, 129, 118]. Multi-agent systems are a sub category of Distributed Artificial Intelligence (DAI) and can be further broken down into cooperative and/or competitive systems [130]. These multi-agent systems typically face many challenges due to the complex nature of interacting agents in a changing environment. These challenges lie in the dimensionality and complexity of the problem.

The curse of dimensionality is always a challenge that needs to be addressed in complex multi-agent systems. Even for agents with a reduced state-action space, dimensionality can exponentially increase as the number of agents increase [131]. The introduction of competitive agents into an already complex cooperative multi-agent system would continue to increase dimensionality and become prohibitive for most algorithms used to explore the state-action space. Some algorithms have successfully modeled large multi-agent systems [119]. Cooperative multi-agent systems will be investigated for the initial implementation of the decision behavior algorithm in multi-agent systems. Competitive and mixed multi-agent systems are excluded from the initial implementation due to the scope of the problem and should be considered in future work. The algorithm choice and behavior framework for the cooperative system can help to manage the dimensionality of the problem as well as the impact cooperation has on the agent's mission logic.

Another challenge in cooperative multi-agent systems arises during the formulation of the learning algorithm. Care should be taken in defining a good goal for the multi-agent system. Maximizing the response of a single agent cannot be done independently since their responses are correlated [131]. A goal that considers both the stability of learning and adaptation to changing behavior should be chosen. Along with defining a goal, the learning algorithm needs to handle the nonstationarity of the problem [132]. In other words, the current optimum agent behavior changes as other agent's behaviors change.

Although challenges exist in cooperative multi-agent systems, the resulting tactics pro-

vide a critical contribution to the mission action design space. Methods to implement the decision behavior algorithm for cooperative agents are desired. Successful implementation would result in a decision behavior algorithm that does not significantly increase the dimensionality. The challenges of cooperative multi-agent systems inform Research Question 2.3.

Research Question 2.3*: How can the decision behavior approach scale as the number of agents increases without a prohibitive increase in dimensionality?*

Two aspects of multi-agent systems can be investigated to inform the hypothesis for this research question. The choice of learning framework used to train the cooperative agents will be addressed to determine if a reduction in sampling requirements for complex agent interactions is possible. The choice of centralized versus decentralized learning will be investigated to evaluate its impact on state behaviors.

Multi-Agent Learning Frameworks

Research into cooperative systems lead to a significant portion of studies whose agents depend on learning algorithms, such as RL [131]. The use of RL in multi-agent systems has been successful due to the complexity that arises for cooperative agents. Optimal cooperative agent behavior may be difficult or even impossible to design. Even if cooperative behavior is known *a priori* and preprogrammed into the agents, that same behavior may become obsolete or non-optimal in a changing environment. Learning algorithms allow for agents to learn new behaviors in a changing environment to improve the goals of the algorithm.

The choice of learning algorithm may enable its ability to be extended to multi-agent

systems. Some algorithms, such as Q-learning, have been shown to scale well with agents using modularity [117]. Other single agent algorithms, such as those based on policy gradient, temporal difference error, and actor-critic methods, have also been successful in scaling from a single agent to large agent simulations [119]. These multi-agent RL algorithms have varying success for large complex systems, but learning frameworks may provide more weight in managing the dimensionality of the problem.

Cooperative agent learning methods have the potential to learn the optimal policy faster than a single independent agent [133, 128]. Although multi-agent systems may have more complex behavior, the ability of agents to share knowledge from trial-and-error exploration speeds up their learning phase. Some examples of learning techniques involve the exchange of information, learning from an expert agent, imitating an expert agent, or utilizing curriculum learning. Early studies found that learning from an expert agent outperformed the exchange of information or the imitation of expert agents [126]. However, as the number of agents increase from a handful to 20 versus 20 or larger, then methods such as curriculum learning may be required. This type of learning seeks to learn policies for basic tasks, then progressively increases the complexity of the task [119]. However, the mission action space is already discretized into decision and state behaviors, which may not require learning techniques such as curriculum learning.

Learning frameworks that have high data sampling efficiency may provide a more appropriate implementation than advanced learning techniques. Multi-agent system frameworks are typically implemented with centralized or decentralized learning and control. The centralized view incorporates joint intentions and actions for all agents in the environment [134]. Decentralized agents make local decisions based on local or global information. Global joint cooperative learning has been shown to outperform decentralized learning in some cases, but results in an exponential growth in communication costs with the number of agents [135]. The advantages of decentralized agents are their scalability and robustness [136]. As the number of agents increase in a decentralized system, their

communication costs do not.

Decentralized agent frameworks have been successfully implemented many with RL algorithms such as Q-learning and Proximal Policy Optimization (PPO) [102, 135]. The sharing of global information of the environment enables the cooperation of agents without the need to make joint decisions. One example of information sharing is the creation of a global map updated by each agent during each time step. The map is then used by each agent to make local decisions [102]. The sharing of information can also enable higher data efficiency during training. Since each agent is simultaneously sampling the environment, the learning algorithm receives more data about the environment during a single time step increasing the algorithms sample efficiency [135]. However, the increased frequency of sampling could lead to a noisy high variance of the policy during training. The advantages and disadvantages of both centralized and decentralized agent control should be considered when choosing an multi-agent learning framework.

Extending the Decision Behavior algorithm

The literature review found that many single agent reinforcement learning algorithms can be extended to cooperative multi-agent systems through centralized and decentralized learning. This finding suggests that the algorithm chosen for the decision behavior algorithm will be extensible to cooperative multi-agent systems. However, the choice of learning framework may have a larger contribution in managing the extensibility of the decision behavior algorithm. A prohibitive increase in dimensionality is defined as one that requires every decision and state behavior algorithm to be retrained for each unique set of agents and technology combinations. The choice of a decentralized agent framework could distinguish two classifications of state behaviors. The first class would be cooperative state behaviors that depend on information from each agent such as its heading or position. The second class of state behaviors are those that may use global information, but do not explicitly depend on cooperative agent information.

Implementation of a decentralized cooperative agent mission with the decision behavior algorithm will be addressed to quantify the scalability of the method. Decentralized agents could improve data sample efficiency and reduce the number or cooperative state behaviors required to train. The reduction in cooperative state behaviors would decrease the approaches dimensionality, thereby increasing its scalability. These findings inform Hypothesis 2.3.

Hypothesis 2.3*: If a decentralized agent framework is utilized to enable agent-independent state behaviors, than dimensionality of a cooperative decision behavior approach will be reduced.*

3.5 Evaluation of Technologies and Tactics

The two motivating technology evaluation methodologies – EBD and TIES for SoS – propose unique technology evaluation and selection methods. EBD seeks to maximize an objective function defined by measures of effectiveness to determine the most effective solution [34]. On the other hand, TIES for SoS explores more formal techniques to account for the complex nature of the multi-attribute and multi-objective design problem [39]. Technology selection typically requires a Multi-Attribute Decision-Making (MADM) method, such as the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method. This MADM method, which seeks to identify the closest solution to an ideal solution, has been applied to many similar problems [137, 138, 139]. The challenge in technology selection does not reside in a lack of methodology, but in generating data that can be compared with such a method.

The comparison of technologies and their respective optimal tactics would provide criti-

Table 3.2: Example Technology Impact Matrix (TIM) modified from [24].

"k"–factor vector	T1	T2	T3
k_1	$+4\%$	$\sim$	-10%
k_2	$\sim$	-3%	$\sim$
k_3	-1%	$\sim$	-2%
k_4	-2%	-2%	3%

cal decision-making information in the grand scheme of a technology evaluation methodology. Difficulties may arise in comparing fundamentally disparate technology-tactic combinations due to the unique solutions generated from the decision behavior algorithm. These challenges must be addressed to utilize the data produced from the decision behavior agent framework. Proper evaluation techniques are critical in correctly identifying the set of optimal technology-tactic solutions.

Traditional technology evaluation methods have successfully compared the performance and effectiveness of a technology trade study, albeit with a fixed mission. Methods to address the impact technology alternatives have on performance can be quantified using a Technology Impact Matrix (TIM) [25]. This matrix, illustrated in Table 3.2, defines technical "k"–factors that can capture the broad range of impacts each technology has on the aircraft's performance. Each "k"–factor element defines the expected impact a technology could have on a disciplinary metric as a $\Delta\%$ where a value of 1.0 or 0% denotes no impact on the baseline disciplinary metric.

Implementation of the TIM enables the performance metrics of the aircraft to be directly linked to which technologies are included in a given alternative. The "k"–factors from each technology within an alternative are summed to determine the overall impact a set of technologies may have of the aircraft's performance. In most cases, the application of "k"–factors can capture the effects of each technology alternative across many disparate technologies [26]. Similar methods have been proposed for tactical alternatives; however, no current methods have been implemented to evaluate the tactical alternative inherent

in a minimally defined mission. The evaluation of tactical alternatives must be further investigated to enable the evaluation of disparate technology-tactic combinations. These findings inform Research Question 3.

***Research Question 3**: How can tactical alternatives be evaluated for a technology alternative using the decision behavior approach?*

Two methods are investigated to explore tactical alternatives within a minimally defined mission. Implementation of the proposed "λ"–factors are discussed and contrasted with the use of a stochastic policy for the decision behavior algorithm.

3.5.1 Extension of Existing Methods

Since their introduction, "k"–factors have been applied to a variety of studies, such as a supersonic business jet, military campaign analysis, and uncertainty analysis for a hybrid wing body aircraft [140, 141, 142, 143]. The extension of "k"–factors to tactical alternatives could provide a similar $\Delta\%$ impact on operational effectiveness metrics. The application of "λ"–factors with a DES model for a humanitarian aid and disaster relief mission have been proposed with no implementation [39]. Several gaps that exist in the proposed technique are the difficulty in defining each tactic, determining each "λ"–factor, and identifying which operational effectiveness metrics each factor would operate on. Assuming those limitations can be addressed, the dimensionality of sampling both "k"– and "λ"–factors can be prohibitive. A notional tactic impact matrix is illustrated in Table 3.3.

Exploration of n technology alternatives using a full factorial DoE would result in 2^n alternatives. The introduction of m tactical alternatives would require a separate DoE for each technology alternative. A full factorial DoE on the technology and tactical alternative space

Table 3.3: Notional Tactics Impact Matrix (TIM) [39].

"λ"–factor vector	Tc1	Tc2	Tc3
λ_1	$+7\%$	$\sim$	-5%
λ_2	$+3\%$	-1%	-2%
λ_3	$\sim$	$\sim$	$+4\%$

would results in 2^{n+m} alternatives. The investigation of six technologies alongside six tactics would results in $4,096$ alternatives. The hypothetical implementation of "λ"–factors with the decision behavior approach would require the training of unique state behaviors and a decision behavior for each alternative. The exponential increase in alternatives is prohibitive when implemented with the decision behavior approach.

3.5.2 Stochastic Agent Policy

Investigation of the learning algorithm's policy, introduced in Section 3.2.2, may provide an alternative area to explore tactical alternatives. The algorithm's policy, π, defines the agent's behavior. Once trained, the agent's policy inputs state information such as the agent's health, heading, or observations, and outputs a corresponding action. Altering the policy once it is already trained could quickly create a suboptimal policy and may invalidate the model. Instead, the type of policy used for the decision behavior algorithm should be purposefully chosen before training each behavior.

Deterministic policies, such as a greedy policy used in Q-learning, always return the same action given the same policy inputs. The greedy policy selects the action that has the highest action-value or expected reward, $\pi(s) = arg\max_a Q(s,a)$ [144]. This results in no variation of the outputs from the agent's policy during testing, which in many cases is desired. In other cases, such as problems with continuous actions spaces or partially observable environments, deterministic policies may not be easily implemented or desired. The deterministic policy set, $\pi : S \rightarrow A$, is sufficient for optimality of a MDP [145]. However, this guarantee has been shown to not hold for POMDPs [146]. This is in part due to

perceptual aliasing where two states may be perceptually identical, but require different action responses [147]. The confounding of states may significantly diminish the the agent's policy [148].

Stochastic policies provide a probability distribution over the actions, $\pi(s) = P[a|s]$. These policies are arbitrarily better than deterministic ones due to their inherent flexibility of action choices [145]. For any given state, the stochastic policy returns a probability distribution that defines the learned optimality of each action. These distributions enable on-policy exploration which provide a built in mechanism of balancing exploration and exploitation throughout training [149]. The issue of perceptual aliasing can also be be eliminated thanks to non-deterministic action selection. For example, this behavior enables the policy to be trained to equally weight the probability of choosing one optimal action over an equally optimal action given perceptually identical states. The stochastic policy can be trained using many RL algorithms such as extensions of traditional algorithms like Q-learning or modern algorithms like Trust Region Policy Optimization (TRPO) and PPO [149, 150, 151].

Action selection using a stochastic policy may result in different tactical outcomes for the same mission. From this perspective, tactical alternatives could be defined as the set of actions chosen over the course of a mission. The variation of tactical alternatives that result from a stochastic policy may provide the desired outcome. However, further investigation is needed to ensure enough variation is present to result in sufficiently unique tactical alternatives.

3.5.3 Evaluation of Tactical Alternatives

Quantification of the MoEs from a fixed mission environment is not sufficient for the exploration of each technology-tactic alternative. Such an environment would not enable an apples-to-apples comparison due to the inherent suboptimal behavior of a fixed mission. The mission environment cannot be fixed for such a methodology that seeks to compare

the technology-tactic synthesis. In reality, the mission environments should be stochastic, noisy, and complex [57]. Such a mission environment would enable each technology-tactic combination to find their optimal mission effectiveness.

The definition of a stochastic mission environment is not the only step required to enable the comparison of technology-tactic combinations. The solutions from the stochastic mission environments may aid in understanding effective tactics for a technology, but may not be the key enabler for the variation of tactical alternatives. The proposed "λ"–factors have many unknowns that must be addressed in order to utilize such a method. Futher research into the existing gaps must be addressed before such a method can be implemented. The use of a stochastic policy provides a well understood method that could be applied to varying the tactical alternatives of each technology. The use of a stochastic mission environment provides some variability to quantify the MoEs, but no variation of tactical alternatives. Based on this research, tactical alternatives of a technology could be captured through stochastic policy. These findings inform the hypothesis below.

Hypothesis 3*: If a stochastic agent policy is used for the decision-state space, then tactical alternatives of a technology can be evaluated.*

3.5.4 Quantification of Mission Effectiveness

The final step in the proposed methodology seeks to evaluate the technology-tactic combinations. Evaluation of the alternatives will first compare the MoEs and then quantify the mission effectiveness of each alternative. Comparison of the alternatives may not be possible if each technology-tactic combination is tested using a unqiue set of stochastic mission environments. Instead, the same set of stochastic mission environments should be used to

test each alternative. This will ensure any 'easy' or 'challenging' stochastic mission, which may produce outliers in the results, is tested with every alternative.

Evaluation of the MoEs is typically conducted by evaluating the effectiveness of each technology alternative from a fixed mission, resulting in a set of point designs. The inclusion of both a stochastic mission and stochastic policy will not only produce the traditional mean response for each MoE, but also a set of individual responses that capture the variation in tactics within a stochastic mission. This additional level of data requires a new method to evaluate the MoEs.

The TIES for SoS methodology proposed the *means* and *ways* plot for the comparison of technology-tactic alternatives. This plot, illustrated in Figure 2.8, aims to identify Pareto frontiers that quantify the upper limit of effectiveness when comparing MoEs. Frontier plots enable the visual identification of the set of non-dominated solutions – the Pareto Frontier – within the entire set of results [152]. This method of data evaluation will be implemented to compare the results from the technology-tuned behavior algorithms.

Once the MoEs are evaluated, the mission effectiveness is quantified using the results from the batch of test missions. Mean and standard deviation of each alternative is calculated to pair with visual results from the means and ways plot. This information is then used to aid in the technology selection process for further investment into research and development of the selected technology.

CHAPTER 4

PROPOSED METHODOLOGY

The Stochastic Agent Approach (SAA) provides a methodology to quantify the mission effectiveness of technologies through tactics formulation and exploration. However, after addressing gaps in the methodology, a new methodology can be proposed to better suit the needs of the overarching technology evaluation methods used for development and acquisition of advanced military capabilities. The proposed four step method enables the exploration of the mission action design space by means of a minimally defined mission. The quantification of mission effectiveness can aid in the technology evaluation process by enhancing the design knowledge and design freedom early on in the conceptual design process. The Technology-tuned Decision Behavior Algorithm for Tactics Exploration (Tech-DEBATE) is defined in this chapter. This methodology, illustrated in Figure 4.1, consists of the problem definition, tactics formulation, tactics exploration and technology-tactics evaluation steps.

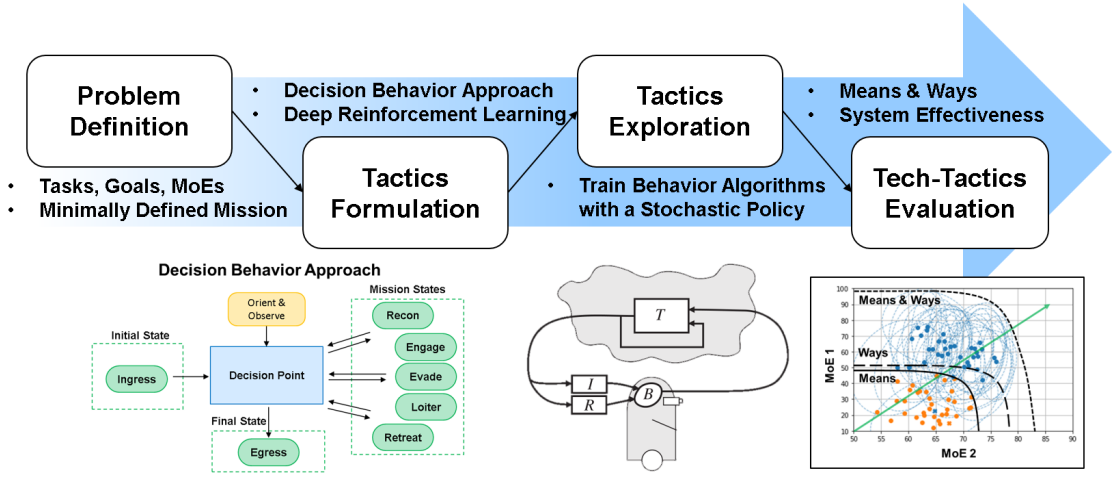

Figure 4.1: The Tech-DEBATE Methodology.

4.1 Step 1: Problem Definition

The first step in the Tech-DEBATE methodology defines the problem to be explored. The problem, consisting of technologies of interest and the mission definition should be clearly identified in this step. Both the technologies of interest and mission attributes can be defined using a morphological matrix. Care should be taken when determining the mission behaviors to avoid limiting the mission design space a priori. The degree of fidelity desired for tactics exploration will depend more on the fidelity chosen in the physics-based M&S environment than for the Tech-DEBATE methodology. This methodology is tool agnostic, so the modeling fidelity desired depends on the choice of ABM software.

Another aspect to problem definition is determining the outputs of interest. These are specified through Measures of Effectiveness (MOEs), and metrics of success. The choice of MoEs to output depends on the goals of the mission. Care should also be taken during MoE selection to minimize bias in the direction a solution may take [64]. Processes to reduce bias in the formulation of MoEs exist that focus on critical operational issues and iterate on the formulation of MoEs.

Once the problem is defined, the modeling framework should be determined. An appropriate ABM should be chosen for the mission of interest. Background research determined that an agent framework consisting of a state machine that can observe and orient itself in a changing environment would suit mission action design space exploration. Specifically, ABMs that are based on POMDPs provide an appropriate agent framework for the decision behavior approach. The decision-state and state-action spaces of each behavior must be defined while accounting for unique states that each technology of interest may require. Each technology may have unique state machines due to the possibility of discrete capabilities between the technologies of interest. An example of discrete capabilities in a contested reconnaissance mission would be the investigation of a reconnaissance aircraft compared to a degraded satellite network. Aircrafts and satellites would have seldom shared states. For

each unique state machine, no assumptions should be made at this point on the significance each state may have on the overall tactics of the agent. Once the mission-action space is defined for the set of technologies, the agent's state machine framework can be developed for the ABM. The three key aspects of the problem definition step are summarized below.

Tech-DEBATE Step 1: Problem Definition

i. Define the mission, measures of effectiveness, and metrics of success

ii. Identify technologies and mission variables using a morphological matrix

iii. Develop an agent-based model using the decision behavior approach

---

4.2 Step 2: Tactics Formulation

The second step seeks to formalize the agent's framework by defining the decision and state behavior logic within the state machine. Each state behavior's logic can be defined by a predetermined optimal behavior or a state behavior algorithm that corresponds to the level of fidelity desired. For example, an aircraft modeled in a lower fidelity simulation may utilize an optimal control algorithm while a higher fidelity simulation may desire an algorithm-based flight controller to evade a defender. The logic that triggers the decision behavior algorithm is also defined.

The logic that triggers the decision behavior algorithm will provide traceability of the decisions made throughout the mission. The decision and state behavior algorithms will be defined by an RL algorithm with a stochastic policy. This methodology does not depend on a specific RL algorithm due to the ever advancing field of machine learning.

Tech-DEBATE Step 2: Tactics Formulation

i. Develop agent logic for minimally defined mission

ii. Define a stochastic policy reinforcement learning algorithm for the agent's decision and state behaviors

4.3 Step 3: Tactics Exploration

The exploration of the mission action design space is enabled by the decision behavior approach. Each algorithm will be tuned for a given technology through learning about its environment. Before the learning phase begins, the user should have an understanding of the size of both the decision-state and state-action space. If the dimensionality of the decision behavior approach is prohibitive, then a tactics compatibility matrix should be used along side the identification of unsuitable state behaviors for each technology alternative.

Once an appropriate dimensionality is determined, each state behavior should be trained using a representative mission to enable the algorithm to sample its entire state-action space. The mapping of both the decision-state space and state-action space should be done through deep neural networks to capture its complexity. The models are then infused into the decision behavior's logic. The decision behavior algorithm can then be trained using the overall mission environment.

In order to compare the results of each technology-tuned decision behavior algorithm, a set of stochastic missions should be determined. Once generated, the set of missions will provide a fixed set of scenarios to compare the technologies in. This is done to main-

tain the same mission sample set across technologies, but avoid a deterministic mission environment that may unintentionally bias a technology.

Tech-DEBATE Step 3: Tactics Exploration

i. Train the behavior algorithms using deep neural networks

ii. Perturb the ways by sampling the stochastic mission space with the trained stochastic policy from each technology-tuned agent

---

4.4 Step 4: Technology-Tactics Evaluation

The key enabler for comparing the synthesis of technology and tactics lies in the quality of distribution metrics produced. Distribution metrics should be determined for each technology-tuned algorithm based on the results from the set of sample missions. These metrics enable the quantification of mission effectiveness and provide an apples-to-apples comparison of technology-tactic combinations. The distribution metrics can be visualized by identifying frontiers in the mission action design space. These Pareto Frontiers can be multi-dimensional depending on the number of MoEs investigated. No single optimal solution can be identified at this point due to the multi-attribute and multi-objective nature of the *means* versus *ways* trade-off.

The critical information produced from the frontiers plot will identify a set of solutions that are all uniquely optimal in comparison to each other. This information can be infused back into the overarching technology evaluation methodology by providing quantitative results on the mission effectiveness of each technology of interest. The mission effectiveness is then combined with each alternative's measures of performance to quantify the overall

system's effectiveness.

Tech-DEBATE Step 4: Technology-Tactics Evaluation

i. Compare technology-tactic alternatives through the identification of *means* and *ways* frontiers

ii. Evaluate system effectiveness

4.5 Overall Research Hypothesis

The proposed methodology aims to improve the quantification of mission effectiveness to better inform technology selection for further investment during the conceptual and preliminary stages of design. The emphasis on a minimally defined mission enables variation of the ways each technology is used, potentially opening up the design space by identifying new ways to use an evolutionary technology or novel ways to use a revolutionary technology. The Tech-DEBATE methodology discretizes the mission action space into decision and state behaviors to enable exploration of a minimally defined mission for each alternative of a technology evaluation study. The proposed methodology provides an improvement in the quantification of mission effectiveness that past methods could not capture due to a fixed mission behavior or stationary mission environment. These findings inform the overall research hypothesis.

Overall Research Hypothesis: *If the proposed Tech-DEBATE methodology is implemented for technology evaluation in a contested environment, than the improvement in the quantification of mission effectiveness can provide more information for technology selection.*

CHAPTER 5
DETERMINATION OF METHOD

Implementation of the Tech-DEBATE methodology must be conducted in order to confirm or deny the hypotheses presented in Chapter 3. A contested reconnaissance mission was investigated for validation with a set of relevant technologies. This choice in mission aligns with the desire to achieve air superiority in contested environments. The quantification of mission effectiveness through a minimally defined mission could aid in the determination of technology investment thereby reducing the risk associated with development and acquisition costs of a new or advanced technology. A need exists to address ISR capabilities in future contested environments [18]. The contested reconnaissance mission for this thesis draws from the Stochastic Agent Approach (SAA) mission scenario developed by Gordon and previously discussed in Section 2.2.4. Figure 5.1 illustrates a notional contested reconnaissance mission simulation of four reconnaissance agents surveying a contested area with ten fixed defenders.

The complexity of the contested reconnaissance mission is suitable for a proof-of-concept implementation of the Tech-DEBATE methodology. The choice of a more complex mission would not provide further insight into the implementation of the methodology, but merely complicate the experimental setup and procedures required. For these reasons, the contested reconnaissance mission provides an appropriate level of complexity to investigate the proposed methodology.

5.1 Definition of the Vehicle and Mission Concept Space

The first step in the Tech-DEBATE methodology is to define the baseline vehicle configuration, mission variables, and their respective alternatives. Alternative vehicle configurations can be determined using the morphological matrix introduced in Section 3.1. Three vehi-

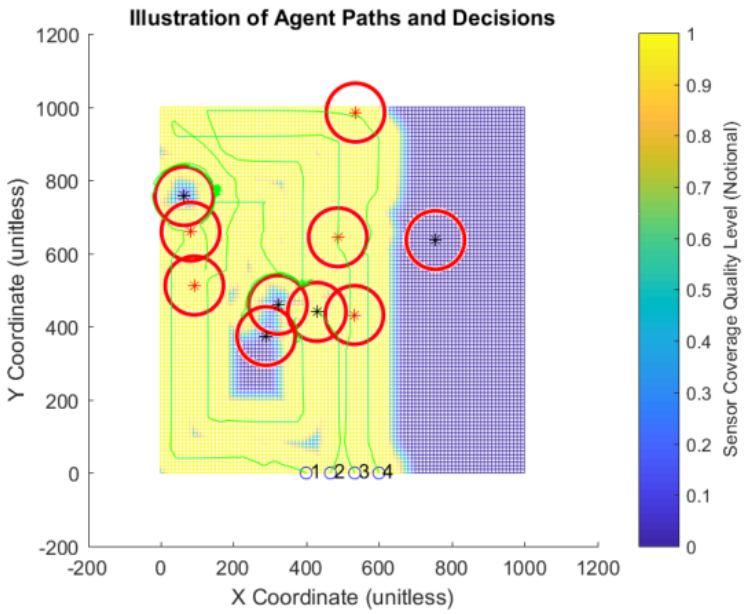

Figure 5.1: Notional contested reconnaissance mission simulation [57].

cle attributes are identified as the vehicle's configuration, payload, and propulsion. Each attribute is then further decomposed into a set of variables that enable the identification of specific alternatives. The morphological matrix for an ISR platform is illustrated in Figure 5.2. This analysis focuses on high level vehicle alternatives as to not over complicate the implementation of the methodology.

Once vehicle alternatives are enumerated, a baseline vehicle is identified. The baseline vehicle, highlighted in blue, is chosen to represent a conventional single engine wing and tail aircraft. Conventional sensors and optics are also selected as the main payload with conventional munitions. The baseline fuel volume of 1500 is unitless for this study. The performance of the baseline configuration is assumed to be inadequate from the start, which motivates the need to either identify technologies of interest from the alternatives within the matrix or identify alternative mission tactics from the matrix.

The morphological matrix is utilized to identify the mission alternatives, illustrated in green in Figure 5.2. The mission attributes define the mission type, mission behaviors, and mission teaming. The mission type is given its own category due to the applicability of the decision behavior approach to many missions using the same agent-based models; how-

Attributes		Alt 1	Alt 2	Alt 3	Alt 4
Config	Vehicle	Quad Copter	VTOL	Wing & Tail	Low RCS
	Planform	Low Speed	High Speed		
	Structure	Traditional Structure	Adv. Structures	Low Cost Structures	
Payload	Fuel Volume	<1000	1500	>2000	
	Sensors/Optics	Traditional Optics	Advanced Optics	COTS Optics	
	Munitions	Conventional Munitions	Adv. Munitions	Directed Energy	
Propulsion	Engine Type	High Supersonic	Supersonic	Subsonic	
	# Engines	6	4	2	1
	Engine Position	Forward Fuselage	Aft Fuselage	Under Wings	Over Wing
	Energy Source	Electric	Traditional Fuel		
Mission	Mission Type	Reconnaissance	Strike	Support	Cargo
	Mission Behaviors	Recon	Evade	Engage	Retreat
	Teaming	Single A/C	Multiple A/C		

Figure 5.2: Vehicle and mission alternatives for a contested reconnaissance mission. The baseline alternative is highlighted in blue.

ever, the extension of state behaviors to multiple missions is not addressed in this work. The mission behaviors are the enumeration of state behavior alternatives that apply to the mission types. For this example, only the mission behaviors for the reconnaissance mission are shown. Mission teaming is given its own category due to the complexity of multi-agent teaming and multi-domain teaming that one might find in SoS. Similar to the vehicle configuration, the baseline mission configuration is highlighted in blue. The baseline mission attributes utilize a single agent conducting a reconnaissance mission with only the 'recon' mission behavior.

The Measures of Effectiveness (MoEs) and Measures of Performance (MoPs) require refinement throughout the development of the ABMS environment. Initial MoEs and MoPs are derived from the successes of ISR platforms from uncontested environments which consists of endurance, payload, integration, and connectivity [18]. However, ISR platforms developed for contested environments also must quantify survivability and lethality. The MoEs can be derived from these findings from successful ISR platforms. The MoEs that are chosen to quantify the mission effectiveness of each technology alternative are endurance which corresponds to the percent of the contested area surveyed and both survivability and lethality which corresponds to the damage incurred during the mission. The MoPs that are chosen to quantify the performance of each technology are the range, probability of kill,

and cruise velocity. However, the initial implementation of the Tech-DEBATE methodology focuses on the MoEs to highlight the novel quantification of mission effectiveness. Connectivity can also be quantified for multi-agent simulations. This MoE is defined as the percent of overlap encountered during reconnaissance, where less overlap translates into better connectivity.

Several baseline ISR systems exist that can be modeled. Systems such as the MQ-1 Predator, MQ-9 Reaper, RQ-4 Global Hawk, RQ-170 Sentinel, U-2, and AC-130 are examples of successful ISR platforms used in uncontested environments [18]. Based on this research, the MQ-1 Predator drone could provide a baseline technology for a reconnaissance mission. The additional systems identified highlight different technologies that each excelled in a single MoE. For example, the MQ-9 Reaper had strike capabilities, the RQ-170 Sentinel enhanced its stealth capability, while the legacy U-2 aircraft conducted reconnaissance at high altitude. However, their success is measured for uncontested environments. Quantifying their effectiveness in a contested environment has yet to be determined. Due to the sensitive nature of military aircraft, the baseline agent is a generic concept that is not necessarily representative of any current UAV system.

Implementation of the contested reconnaissance mission is first defined in a discrete state space to allow for a simplistic mission environment to investigate the decision behavior algorithm. Once an appropriate algorithm is determined, the complexity of the mission and state space can be increased. The complexity of a continuous contested reconnaissance mission will be investigated alongside the implementation of the decision behavior approach to address the second research question outlined in Section 3.3.

5.2 The Discrete Contested Reconnaissance Mission

The reconnaissance mission selected in the morphological matrix is further defined before the development of the agent-based model. In this investigation the mission is conducted in an adversarial environment with stationary defenders. The defenders are immobile objects

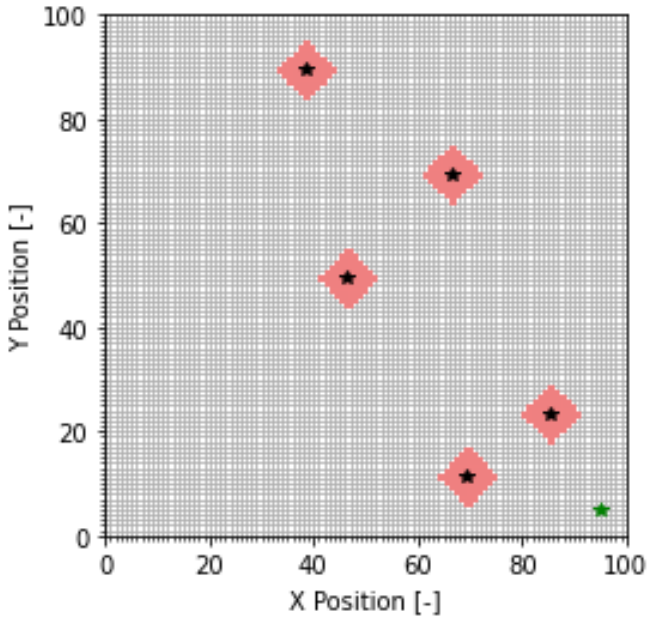

Figure 5.3: The contested reconnaissance mission discretized into a 100×100 map with 5 defenders.

that represent an adversary's asset. The friendly surveying agent must locate the defenders without unnecessarily putting itself at risk. The friendly agent's risk is recorded as damage. If the agent incurs too much damage it is destroyed. Similarly, if the agent encounters too much risk, then the probability of mission success is unlikely. The total damage an agent incurs throughout a mission is analogous to the agent's survivability. The initial contested reconnaissance mission, illustrated in Figure 5.3, is discretized into a 100×100 grid. Each block within the grid is a single space the agent can be located at. This discretization defines a finite set of locations the agent can have.

The defender locations, shown as black stars, are stochastically generated within the contested map. Each defender has an initial field of view, highlighted in red, equal to five spaces in all directions. If the reconnaissance agent is within the defender's field of view, it will incur damage each time step. The reconnaissance agent will have an initial starting location, marked by a green star, near the bottom of the contested area. The agent will survey the region that is within its field of view, which is also equal to a radius of five spaces from the agent's current location.

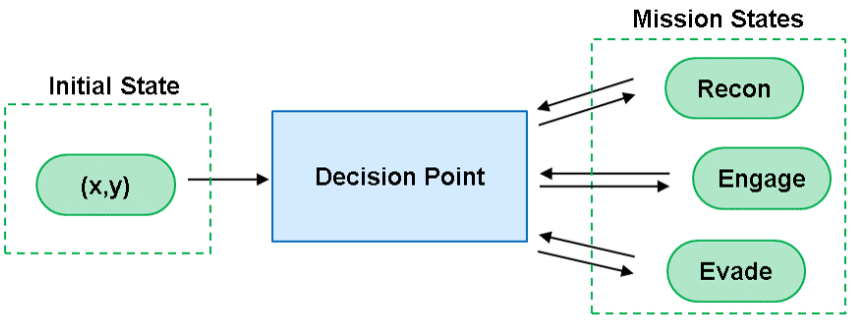

Figure 5.4: The agent's framework for the contested reconnaissance mission.

The ABM for the discrete contested reconnaissance mission is developed in the Python coding language to provide a simple and efficient modeling framework to validate the methodology. The mission environment is modeled as a 2-dimensional area where the defenders are assumed to be located on the ground while the surveying agent is flying at a constant altitude and speed above the defenders. Variations of this contested reconnaissance mission are investigated in the context of each hypothesis presented in this thesis.

5.2.1 The Decision Behavior Agent

This section will formally define the agent framework used for the investigation of hypothesis 1 from Section 3.2.2. The experiment implements a single reconnaissance agent to search a contested environment where defenders are residing at unknown locations. The reconnaissance agent's state machine framework is illustrated in Figure 5.4. The initial goal of the reconnaissance agent is to survey the predefined area of interest while minimizing damage inflicted by defenders. This mission will be explored using an initial starting location defined by an (x, y) coordinate and three predefined state behaviors.

The three state behaviors are identified from the morphological matrix. The chosen mission type, behaviors, and teaming capabilities are highlighted in green in Figure 5.5. The three identified state behaviors – including the baseline reconnaissance behavior – are described in Table 5.1. Each behavior should be defined as a high level behavior that has some generality to the execution of the behavior. The high level behavior defined for each

Attributes		Alt 1	Alt 2	Alt 3	Alt 4
Mission	Mission Type	Reconnaissance	Strike	Support	Cargo
	Mission Behaviors	Recon	Evade	Engage	Retreat
	Teaming	Single A/C	Multiple A/C		

Figure 5.5: Identification of mission attribute alternatives from the morphological matrix.

Table 5.1: Definition of state behaviors for a reconnaissance agent.

State Behavior	Description
Recon	The agent is actively surveying the area while avoiding known defender locations
Engage	The agent is engaging the defender with predefined offensive capabilities
Evade	The agent maneuvers away from the defender in an optimal manner

state reduces the bias SMEs may have on the overall technology-tactic solutions by defining a general behavior versus a specific tactic. From this perspective, the decision to evade or engage a defender would be a tactic, while the actual evasion behavior is an action that can be defined by a function or algorithm.

Each state behavior definition is outlined further to define the agent's logic during their execution. Simplistic state behaviors were chosen to reduce the complexity of the initial implementation of the decision behavior algorithm during the investigation of the first hypothesis. The reconnaissance behavior is defined as an optimal path planning algorithm while the engagement and evasion state behaviors are predefined deterministic behaviors.

State Behavior Logic

The reconnaissance state behavior explores the contested area without information on defender locations. Once a defender is found, the decision behavior algorithm will choose which state behavior to perform next. If the reconnaissance behavior is chosen, the agent will continue exploring the contested area without regard for the newly found defender. In some cases, choosing to continue reconnaissance when the agent just grazes the defender would provide the agent with the same outcome as evading the defender; however, in other cases, the agent will fly straight over the center of the defender if reconnaissance is chosen.

Reconnaissance. The field of path planning algorithms was investigated to determine an optimal search algorithm for the reconnaissance behavior. Informed and Uninformed path planning algorithms provide methods to solve the shortest path problem. This problem type seeks to find the shortest path from a start node to a goal node by minimizing the sum of individual distances between each intermediate node [153]. Informed algorithms are typically more useful than uninformed search algorithms due to their improved heuristics. Uninformed search algorithms, such as breadth-first or depth-first, systematically expand nodes in the path tree until the goal is found. In contrast, informed search algorithms, such as A^* or best-first, quantify the path length or goodness to prioritize paths that have a higher likelihood of being the optimal path.

The A^* algorithm in particular is notable since it can be admissible while providing a comparatively simple and efficient method when compared to other informed search algorithms. An admissible algorithm guarantees that the search will determine the optimal path from a starting node to a goal node [154]. If the search heuristic is nondecreasing along any path, than the algorithm is considered admissible. These properties are useful to understand when developing the heuristics for the path planning problem. The reconnaissance state behavior must define its heuristics in a way to ensure optimality of the path. The A^* algorithm provides a sufficient path planning algorithm to defined the reconnaissance behavior. This algorithm, defined by Algorithm 1, can determine the optimal search path for the agent to explore the contested area. Given and initial starting point, S, the algorithm will expand each adjacent node relative to its current node and evaluate some function $f(n)$, where n is the node being evaluated. The A^* algorithm will then choose the partial path N from the set of partial paths Q that provides the minimum summation of $f(n)$. Each expanded node is tracked in an array, E, to ensure sub-optimal paths are not reevaluated.

The reconnaissance state behavior uses the A^* algorithm to choose the optimal path as the agent explores the contested environment while not within a defender's field of view. The specific logic that is utilized when the reconnaissance behavior is selected is illustrated

Algorithm 1 A^*

1: Initialize Q with the initial starting node S
2: If Q is empty, end algorithm; else choose N with minimum $f(N)$
3: If n = goal, algorithm complete; else remove N from Q
4: If N is in E, return to step 2; else add N to E
5: Extend N to adjacent nodes not contained in E and evaluate $f(n)$ for each node
6: Add all partial paths N to Q
7: Return to step 2

in Figure 5.6. The agent is given an initial start node, S, and no information about the defenders. The behavior then starts by selecting an action using the evaluation function, $f(n)$. This function, defined by equation 5.1, defines the estimate cost of the path from S to the goal through the node n. The heuristic, $h(n)$, defines the estimated cost from n to the goal and the cost, $g(n)$, defines the cost from S to n.

$$f(n) = g(n) + h(n) \tag{5.1}$$

The heuristic used to determine the optimal node depends on the expected area the agent will explore while the goal 'node' is defined as exploring 100% of the contested area; however, the contested area will be considered sufficiently explored once coverage is greater than 70%. This heuristic is consistent, always greater than zero, and nondecreasing which will ensure admissibility of the algorithm. The mission will end once the 70% goal is reached or the agent's health drops to zero. If neither of those end conditions are met, the agent will continue to explore until a defender is found. The discovery of a new defender will end the state behavior and trigger the decision behavior algorithm to decide how the agent should act while within the defender's field of view.

If the agent decides to continue reconnaissance within the defender's field of view, the A^* algorithm will continue to choose an optimal path based on the amount of area explored for each potential action without regard for any damage incurred. Some paths may only be within the defender's field of view for a single node while others may cross straight over the center of the defender. This information is given to the decision behavior algorithm in

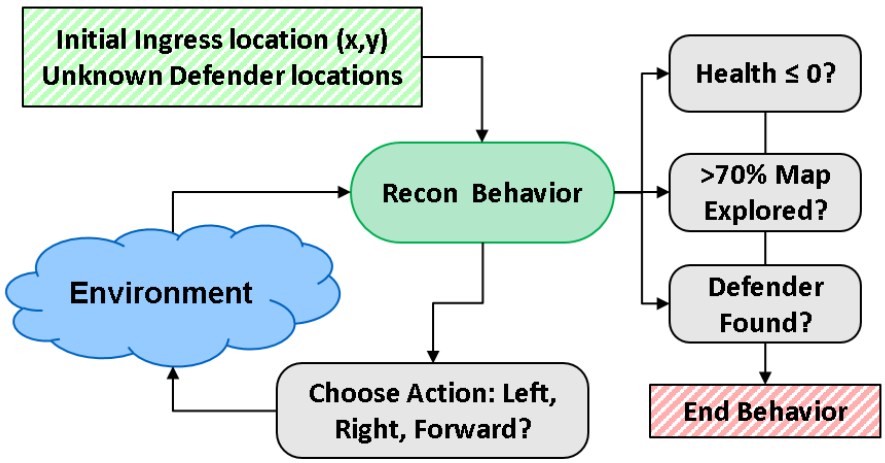

Figure 5.6: The agent's reconnaissance state behavior logic.

hopes that it will learn to choose the appropriate situations to select and when not to select the reconnaissance behavior.

Evasion. The optimal path to evade a defender is to exit the defender's field of view in the shortest number of steps possible. The defined mission environment allows the agent to immediately exit the defender's field of view once it has entered. This behavior, illustrated in Figure 5.7, can be defined by a single action. The choice of action – left or right – is deterministic depending on the agent's relative position to the defender. If the defender is to the right of the agent relative to the agent's point of view, than the agent will evade left relative to its current heading. Alternatively, the agent will move right relative to its current heading if the defender is to the agent's left.

The evasion state behavior logic, illustrated in Figure 5.8, only requires a single loop to successfully evade the defender for the discretized contested area. Although the agent only takes a single step, the agent's health is still checked to ensure it has not incurred too much damage to continue the mission. Once the agent has successfully evaded the defender, the evasion state behavior will end and transition back to the reconnaissance behavior until a new defender is found.

Engagement. Similar to the evasion state behavior, the engagement state behavior has

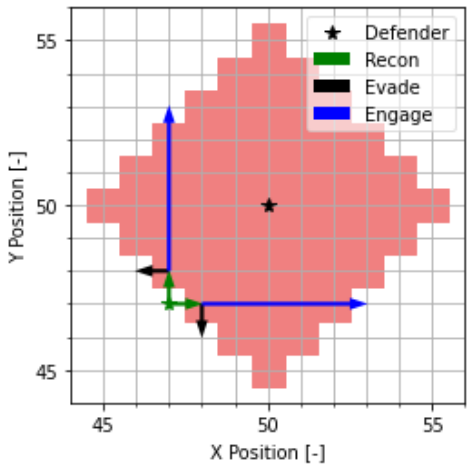

Figure 5.7: The agent's evasion and engagement behavior relative to its heading once a defender is found. The engagement behavior assumes the agent has missed and/or is out of ammunition.

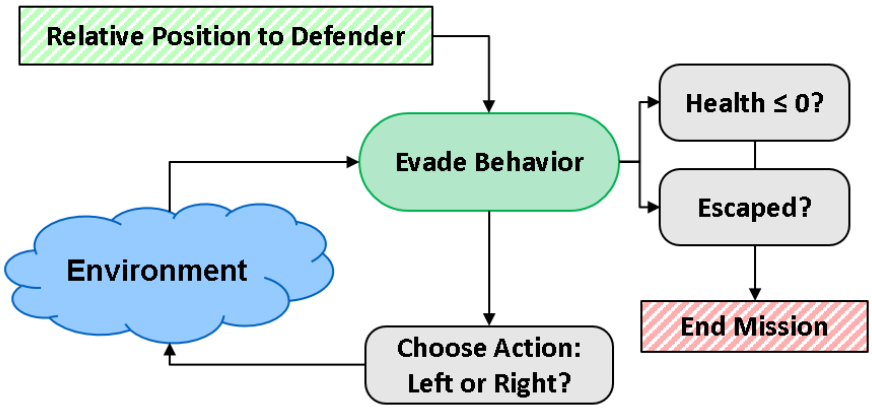

Figure 5.8: The agent's evasion state behavior logic.

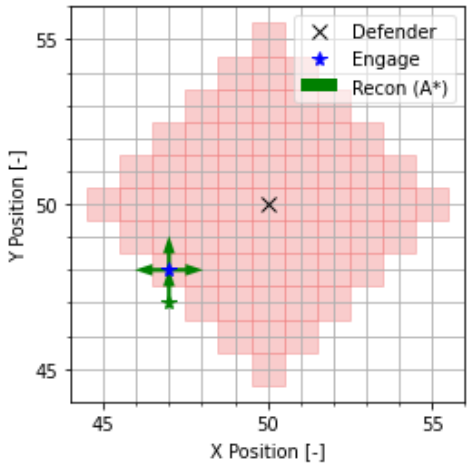

Figure 5.9: The agent's successful engagement behavior once a defender is neutralized.

a deterministic set of actions it can take to neutralize or escape a defender. The probability of kill, P_k, defines the likelihood of a successful engagement. Once the engagement state behavior is chosen, a random number is drawn from a uniform distribution between 0 and 1. If the random number is less than P_k, the engagement is successful, while a random number greater than P_k will result in an unsuccessful engagement. Figure 5.7 illustrates an unsuccessful engagement where the agent missed and has no more ammunition to engage the defender again. If the agent missed and still has ammunition, it may engage the defender again by drawing a new random number; however, once the agent has no more ammunition, it will continue on its path until the agent is outside of the defender's field of view. Figure 5.9 illustrates a successful engagement behavior where the agent neutralizes the defender and transitions back to the reconnaissance behavior. The reconnaissance behavior will then choose to move the agent forward, left, or right based on the A^* algorithm.

The agent's logic, illustrated in Figure 5.10, is initially provided with the agent's current ammunition amount. If the agent has ammunition left, it will always engage with a $P_k = 0.85$ of success. if the agent is out of ammunition, the agent will carry on with its current heading until it has left the defender's field of view. The decision to maintain the agent's heading if no ammunition is available will help to discourage the decision behavior algorithm from choosing to engage with low ammunition or no ammunition. During

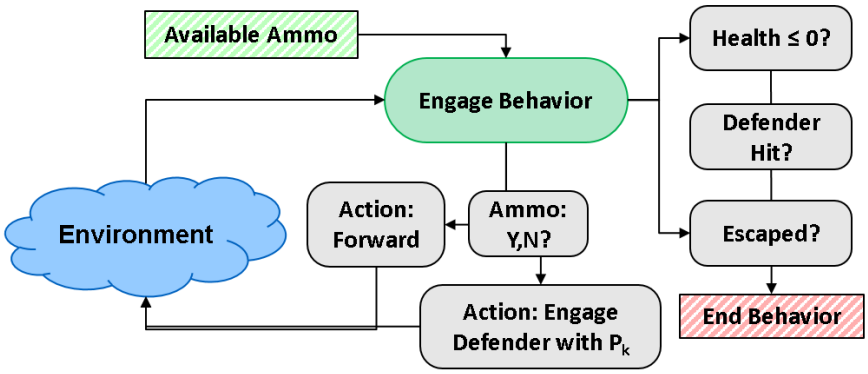

Figure 5.10: The agent's engagement state behavior logic.

each step, the agent evaluates its health, distance from defender, and if it neutralized the defender. If any of the exit conditions are met, the agent will end its engagement behavior and either end of mission if no health is left, or return to the reconnaissance state behavior until a new defender is found.

Decision Behavior Logic

Once the state behaviors are formally defined, the agent's decision behavior logic can be defined through further definition of the state machine framework. The decision behavior algorithm will sample the mission through the development of a representative mission. Since the decision behavior is only triggered when a new defender is found, then the algorithm can be trained by starting from an initial state where the defender was just found. Figure 5.11 illustrates the agent's decision behavior logic. Since the state behaviors require no training to define their behavior, they can be integrated immediately into the agent's decision behavior logic. The agent is given a set of initial conditions that define its relative location to the defender and if ammunition is available. The decision behavior algorithm uses this information to select the state behavior that is expected to provide the highest reward. The agent performs the selected state behavior until the defender is neutralized or the agent has escaped. During training of the decision behavior algorithm, the reconnaissance behavior maintains its heading when selected. During evaluation of the overall mission,

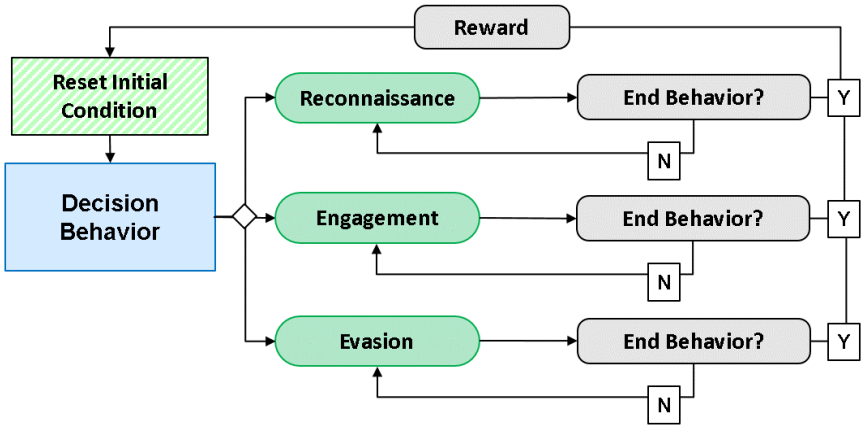

Figure 5.11: The agent's decision behavior logic for training.

the reconnaissance behavior will be driven by the A^* algorithm. This is a conservative assumption since the A^* algorithm is provided no information about defender locations. After the completion of the selected state behavior, the decision behavior algorithm receives the actual reward for that behavior given the initial state.

Several rewards were defined for the decision behavior algorithm. Some rewards were state specific while others were tied to global information. Both the evasion and engagement state behaviors return a reward for evading the defender and neutralizing the defender respectively. These state specific rewards equally encouraged use of both state behaviors. If the engagement behavior was selected and the agent was unable to neutralize the defender, then the agent was not rewarded once it left the defender's field of view. This logic discourages the agent from selecting engagement when it has zero ammunition available. An additional reward was returned for the amount of damage incurred during the simulation which encourages the agent to reduce incurred damage. The damage is subtracted from the total reward so far. If the agent's health also drops below zero at any point, a penalty is applied to the total reward which discourages risky behaviors if the agent has low health.

In order to ensure the algorithm learns which situations return the highest reward, the initial conditions of the agent must be sampled such that the entire decision-state space is covered. The inputs of the decision-state space can be defined by the agent's health, the

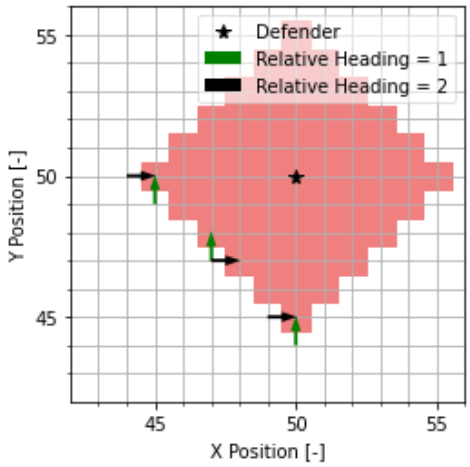

Figure 5.12: Representation of the two relative headings sampled during training of the decision behavior algorithm.

agent's available ammunition, the agent's relative heading, and the agent's relative distance to the defender. The agent's health is discretized into 20 states in increments of 5 from 5 to 100. The agent's available ammunition is a binary value where a value of one is a state in which the agent has ammunition left and a value of zero corresponds to no ammunition available. The heading of the agent is defined as a heading of one which corresponds to a state where the defender is directly in front of or to the right of the agent. In contrast, a relative heading of two corresponds to a state where the defender is directly in front of or to the left of the agent. The definition of the agent's relative heading, illustrated in Figure 5.12, does not change as the agent is rotated around the defender. The final input of the decision-state space is the agent's distance to the defender. This distance is defined as a one dimensional distance parallel to a relative heading of two. The set of three black arrows illustrated in the figure correspond to distances from left to right of 5, 2, and 0 while the set of green arrows correspond to distances of 5, 3, and 0. The total number of discrete distances the agent can be from the agent is in the set $\langle 0, 1, 2, 3, 4, 5 \rangle$.

The total number of possible input states that the decision behavior algorithm can have is 480. The dimensionality of the decision-state space can then be quantified as the number of input states multiplied by the number of actions, resulting in a size of $1,440$. The

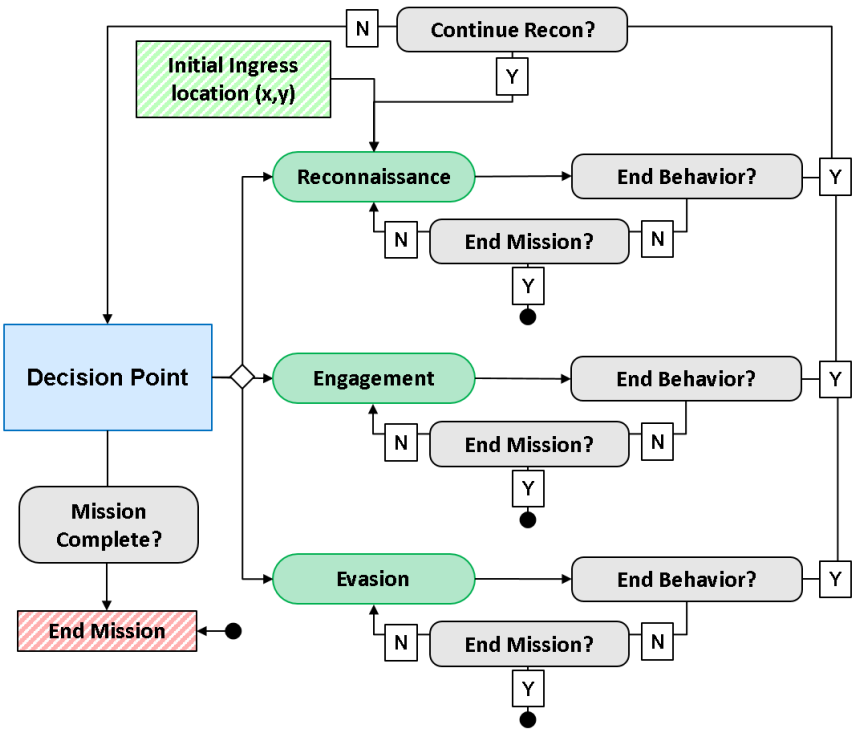

Figure 5.13: The agent's contested reconnaissance mission logic.

size of the decision-state space is very manageable for both Reinforcement Learning (RL) algorithms and Genetic Algorithms (GA) using a policy table implementation.

Mission Logic

The agent's mission logic integrates the logic from each state behavior and the decision behavior. This logic is used once the decision behavior algorithm has been trained to quantify the agent's mission effectiveness. The agent's ingress state is defined as a single starting point specified by the coordinate (x, y). Once the simulation starts, the agent will conduct reconnaissance until it finds a defender. The decision behavior is then triggered requiring the agent to choose between the three state behaviors given its current input state. Throughout the mission, the agent's logic will check if incurred damage surpasses 100, in which case the mission will end and be classified as failed. Successful completion of the mission is achieved when 70% of the contested area has been surveyed.

5.2.2 The Decision Behavior Algorithm

The desired decision behavior algorithm is one that efficiently maximizes the optimality of the agent's decision framework without constraining the minimally defined mission. The framework and logic of the mission defined in the previous sections ensure the mission is minimally defined. The literature review found that both RL and GA could be appropriate algorithms for the decision behavior without placing any constraints on the minimally defined mission. Now that the ABM has been defined, both an RL algorithm and a GA can be identified and implemented for the decision behavior algorithm.

Reinforcement Learning Implementation

There are many RL algorithms that are suitable for the decision behavior algorithm applied to a discrete MDP problem. The simplicity of the mission enables the applicability of many modern techniques. An appropriate algorithm is one that is robust to the type of problem, simple to implement, applicable to multi-agent systems, and performs well compared to other state-of-the-art RL algorithms.

Algorithm Selection. Q-learning is one of the more common algorithms that has been studied by many for decades [155]. The algorithms popularity is due in part to its simplicity to implement in a tabular settings and its success in optimally solving MDPs [116].

The Q-learning algorithm defines a Q value for every state-action pair, (s, a). The Q function – or action-value function – defines the mapping of the decision-state or state-action spaces. The learning algorithm updates its Q values after every *episode* [116]. Such a learning scheme consists of n episodes that have the following steps:

- Observes its current state s_n,
- selects and performs an action a_n,
- observes the subsequent state y_n,
- receives an immediate payoff r_n, and
- adjusts its $Q_{n-1}(s, a)$ values using a learning factor α_n [116]

The optimal Q values are determined by sampling the environment and updating the Q function. The function for updating the Q-learning table is shown by equation 5.2 where $Q_t(s, a)$ is the Q value at update episode n for action a given state s [156]. Each training step uses the reward, $r(s)$, to update the Q function.

$$Q_n(s_n, a_n) = Q_{n-1}(s_n, a_n) + \alpha(r(n) + \gamma \max_a Q_{n-1}(s_n, a) - Q_{n-1}(s_n, a_n)) \tag{5.2}$$

where α is the learning rate $\alpha \in [0, 1]$ and γ is the discount factor $\gamma \in [0, 1)$. The learning rate smooths the update average due to potential randomness in the rewards. The discount factor improves the algorithms convergence by discounting future rewards. This hyperparameter is one of the most important parameters for tuning an RL algorithm. It is typically set to 0.99 and reduced if convergence is not sufficient [157]. The optimal Q function, shown by equation 5.3, follows the Bellman equation [102].

$$Q^*(s, a) = r(s) + \gamma \sum_{s' \in S} p(s'|s, a^*) \max_{a' \in A} Q^*(s', a') \tag{5.3}$$

where $p(s'|s, a)$ is the probability of transitioning to state s' from state s given the optimal action a^*. The convergence of such an algorithm can be done through the trade-off of exploration and exploitation. One common method is to use the ϵ-greedy approach which provides a threshold for trading off the selection of a random action or exploiting an optimal action given state s [102]. If a randomly drawn value is less than ϵ, a random action will be chosen, otherwise the current Q function will be evaluated to determine the action that results in the highest Q value. The ϵ variable is initially set to $\epsilon = 1$, but decreases logarithmically as the training episodes increase, which transitions the action selection from exploration to exploitation. Exploration can be further extended by holding $\epsilon = 1$ for the first N number of training episodes. Once $\epsilon = 0.01$, its value will not decrease further.

Since Q-learning was introduced, many additions to the algorithm have been proposed to address the potential of overestimating action values due to the algorithm's update pol-

Algorithm 2 Double Q-Learning [156]

1: Initialize Q^A, Q^B, s
2: **repeat**
3: Choose a, based on $Q^A(s, \cdot)$ and $Q^B(s, \cdot)$, observe r, s'
4: Choose (e.g. random) either UPDATE(A) or UPDATE(B)
5: **if** UPDATE(A) **then**
6: Define $a^* = arg\max_a Q^A(s', a)$
7: $Q^A \leftarrow Q^A(s, a) + \alpha(s, a)(r + \gamma Q^B(s', a^*) - Q^A(s, a))$
8: **else if** UPDATE(B) **then**
9: Define $a^* = arg\max_a Q^B(s', a)$
10: $Q^B \leftarrow Q^B(s, a) + \alpha(s, a)(r + \gamma Q^A(s', a^*) - Q^B(s, a))$
11: **end if**
12: $s \leftarrow s'$
13: **until** end

icy of always choosing the maximum Q value [156, 158, 159, 160]. The overestimation of action values can produce suboptimal policies where overestimation is not uniform [161]. The potential for overestimation is found to be more prevalent in large action spaces, which is particularly concerning for its use with a more complex decision-state space [160]. Double Q-learning has found success in reducing the overestimation of action values [161]. This algorithm uses two Q functions to decouple action selection and evaluation. The double Q-learning algorithm is outlined in Algorithm 2.

The main contribution of double Q-learning is the Q-learning update function, shown in steps 7 and 10. Each update function uses one Q function to select the action and the other to evaluate the action. This decoupling results in a more robust algorithm that not only reduces overestimation, but also decouples the overestimation error from the size of the action space. The comparison of overestimation error with a single Q-learning and the double Q-learning update function versus the number of actions is illustrated in Figure 5.14 where Q and Q' are the two value functions and $V_*(s)$ is the optimal value function for any state $s \in S$. The significant decrease in overestimation error can be seen for action spaces as small as 4 actions. The occurrence of error in smaller action spaces is significant since the current problem formulation defines 3 state behaviors for the 'actions' in the decision-state space.

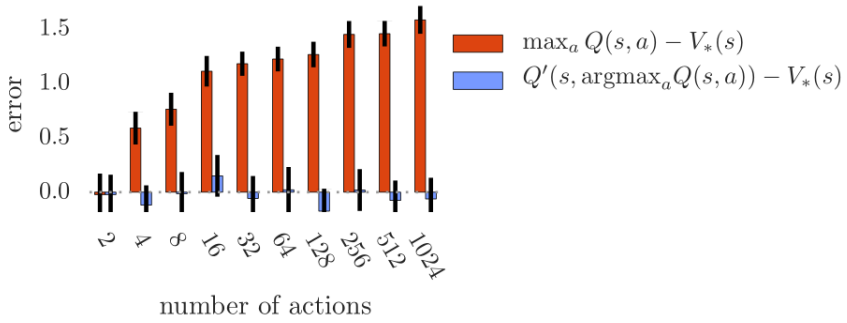

Figure 5.14: Overestimation error of Q-learning and double Q-learning versus the number of actions [161].

The performance of Q-learning and Double Q-learning has proven to be competitive when compared to other state-of-the-art RL algorithms [158, 159]. Both algorithms have also been successfully applied to cooperative multi-agent systems [118, 117, 119, 102]. Based on these findings, double Q-learning is selected as the appropriate RL algorithm for the decision behavior algorithm due to its robust application to many problem types, simple implementation, extensibility to multi-agent systems, and competitive performance with other state-of-the-art algorithms.

Algorithm Implementation. Due to the relatively small size of the problem's decision-state space, training of the double Q-learning algorithm can be done on a personal computer (PC) equipped with a standard number of CPUs. The algorithm's hyperparameters are set to $\alpha = 0.001$ and $\gamma = 0.99$ to provide sufficient convergence within $15,000$ training episodes. The double Q-learning algorithm is updated using experience replay, which randomly selects a batch of N number of experiences that were previously encountered. The replay memory capacity is set to maintain the $1,000$ most recent sample sets of the state, action, reward, and next state defined as $[s, a, r, s']$. Once the replay memory reaches N number of experiences, the Q tables are updated after each new training episode. Each update draws N random sample sets from the replay memory where each sample set is used for training one of the randomly selected Q functions. A batch size of $N = 64$ is used. The convergence is tracked by evaluating each state in the Q function and summing the reward from action a. The action that is chosen for evaluation is determined by taking the

maximum of the average of the two Q functions. Convergence of the double Q-learning algorithm are presented in Section 5.2.2.

Genetic Algorithm Implementation

The desired characteristics of a GA algorithm for the decision behavior algorithm are the same as an RL algorithm in that a robust, simple, extensible, and one that is competitive with state-of-the-art GAs is desired. Many advances in the field of evolutionary algorithms have been made that allow GAs to rival RL algorithms. Recently, techniques and methods that seek to encourage population diversity as opposed to population fitness have provided improvements [162]. These improvements prolong the exploration phase of the algorithm to increase the probability of finding the global optimum. Some techniques that increase diversity include increasing the mutation rate of individuals when rate of fitness increase plateaus, using genetic diversity for parent selection criteria, and altering the structure of the population. However, even with these techniques, search algorithms may still converge to a local optimum early on.

One method that has proven competitive with RL is Novelty-Search (NS). The NS algorithm rewards individuals for discovering novel behaviors during evolution as opposed to higher fitness values. This type of GA has proven to be robust to deceptive problems in which the rewards may encourage behavior that avoids the global optimum [113]. However, the emphasis on novel behaviors limits the effectiveness of NS on large behavioral spaces which may present limitations as the decision-state space increases in complexity [162].

Another limitation that plagues most GAs is the number of parameters required to define the algorithm. These parameters include but are not limited to the population size, the number of parents for reproduction, the number of mutations, and the number of generations. The parameter tuning problem (PTP) is defined as the problem of designing the parameters for a GA implementation [115]. Determining the perfect set of parameters – assuming they exist – before solving the algorithm is almost impossible [163]. Convergence

Algorithm 3 Simple Genetic Algorithm

Input: population size N, number of selected individuals T, number of mutations M,number of generations G, fitness function F.
for $g = 1, 2..., G$ generations **do**
  if $g = 1$ **then** initialize population
  for $i = 1, ..., N$
    evaluate $F(i)$
  end for
  Select top T individuals for reproduction using truncation
  for $i = 1, ..., N - T$
    generate offspring via crossover
  end for
  for $m = 1, ..., M$
    mutate a randomly selected gene
  end for
end for
Return: Elite Individual(s)

of the algorithm can vary widely depending on the specification of parameters. Parameters that provide sufficient improvement must be determined in order to effectively optimize the decision behavior genetic algorithm.

Due to the challenges presented by the PTP, a simple GA is chosen for the decision behavior algorithm to reduce the complexity of implementation. The reduction in complexity provides an algorithm that has a low number of parameters required to tune. The simple GA utilized is defined by Algorithm 3. This algorithm incorporates the main components found in GAs such as selection, reproduction, evaluation, and replacement [112].

The main components of the GA can be specified a number of ways. Many methods for selection criteria are available and debated in literature [107]. However, truncation selection is chosen to determine the parent chromosomes for reproduction due to its simplicity. This selection method chooses the top T individuals for mating [113]. Two parents identified from selection are chosen at random to reproduce. The reproduction step uses crossover to generate the offspring's chromosome by using a single crossover point at the center. This results in the first parent to provide the first half of its genes while the second parent provides the second half of its genes to the offspring. Mutation of the offspring is

conducted by mutating M genes within their chromosome. Once mutation is complete, the T parents are combined with the $N - T$ offspring to create the new population for fitness evaluation.

Algorithm Implementation. The PTP is first set up as a separate GA that treats Algorithm 3 as its fitness function. The same components specified for the simple GA are used for the tuning algorithm. The parameters used to specify the PTP's GA highlights the inherent complexity. Value encoding is used to define the chromosome for both the parameter tuning and decision behavior algorithm problem. The parameter tuning chromosome is a set of 4 numbers drawn from a uniform random distribution from the intervals $N \in [1, 1000]$, $T \in [0, N]$, $M \in [0, S]$ where S is the number of states defined in Section 5.2.1, and $G \in [1, 50]$. The parameters within the chromosome are ordered as follows $[N, T, M, G]$. The chromosome used for the decision behavior genetic algorithm defines one action for all states as an integer in the set $[0, 1, 2]$, which represents reconnaissance, engagement, and evasion respectively. The mutation of this chromosome requires a unique mutation to ensure the solution space is explored. Each gene in the chromosome adds a randomly selected integer chosen uniformly from the set $[-1, 0, 1]$. This mutation logic results in a 33% chance the gene does not mutate.

The initial PTP defined its own parameters as $N_{PTP} = 10$, $T_{PTP} = 4$, $M_{PTP} = 1$, and $G_{PTP} = 10$. These parameters provided sufficient improvement in the decision behavior genetic algorithm. The optimal parameters to use for the decision behavior genetic algorithm were found to be $N = 169$, $T = 15$, $M = 25$, and $G = 50$. However, further convergence of the algorithm was achieved by increasing the number of generations to $G = 90$. Convergence and results for the decision behavior genetic algorithm are presented in Section 5.2.2.

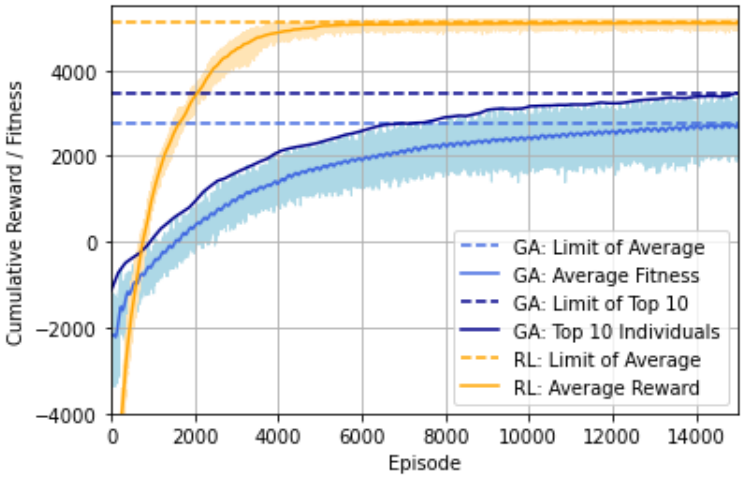

Figure 5.15: Convergence of both the RL and GA decision behavior algorithms.

Selection of The Decision Behavior Algorithm

The appropriate decision behavior algorithm is one that efficiently maximizes optimality of a minimally defined mission. This criteria seeks to quantify the optimality of each algorithm's converged solution along with the sampling efficiency required to achieve it. Both algorithms are applied to an identical agent that returns the same reward for a given decision-state pair. This enables the direct comparison of each algorithm's performance. Along with their convergence, further analysis is conducted to investigate state behavior selection preferences of each solution and quantification of each algorithm's overall mission effectiveness.

The convergence of each algorithm is determined using five samples, each with a unique random seed. The choice of only five random seeds has been shown to produce statistically different results [164]; however, more investigation is needed to properly identify a sufficient random seed sample size for the comparison of multiple algorithms. The RL reward and GA fitness history is illustrated in Figure 5.15. The five random seed samples are averaged and smoothed using Savitzky-Golay filtering with window size 169 [165]. In addition, the fitness of the top 10 individuals from each generation of the GA are similarly averaged and smoothed with a window size of 7.

The RL algorithm not only converges to a higher reward, but also does so in less training episodes than the GA. The reward and fitness values for the RL and GA implementation converged to $5,001.2$ and $2,748.8$ respectively. However, since the average fitness values for the GA are calculated using the whole population, a higher fitness can be obtained by determining the average fitness from a set of elite individuals within the population. The average fitness of the top 10 individuals from each generation result in a maximum average fitness of $3,474.7$. A single individual is not preferred due to its lack of diversity in behaviors. A set of elite individuals may have similar fitness values, but provide a much more diverse set of behaviors. The elite individuals from the GA significantly improve its average fitness, but is still well below the RL algorithm's average reward. The difference in average rewards for the two algorithms suggest that an increase in sample seed size may not have increased the GA's fitness reward enough to be competitive with the RL algorithm's reward.

The computational expense of each algorithm is important to consider as well when determining sampling efficiency of each algorithm. The average computational time for RL to conduct $15,000$ episodes is 169.0 seconds while the GA completed the same amount of episodes in an average of 128.0 seconds. Both algorithms proved to have a very low computational expense, but a high sampling efficiency is still important to ensure the algorithm can scale as complexity increases without reaching a prohibitive computational expense. Based on the average computational time, the GA seemed to be more computationally efficient; however, the computational expense for each algorithm to reach convergence tells a different story. The RL algorithm is determined to converge after $6,000$ episodes while the GA converges after $14,000$ episodes. The resultant computational time for each algorithm to converge is 73.0 and 117.8 seconds for RL and GA respectively. These findings show that the RL algorithm is not only more sample efficient, but also can achieve convergence with a slightly lower computational expense than the GA.

Further analysis of the results is conducted to ensure each decision behavior algorithm

performs as expected and to quantify the mission effectiveness each algorithm can achieve. The decision behavior algorithm's preference of state behavior for each decision-state can be investigated to understand their learned behavior's. Figures 5.16 and 5.17 illustrate the frequency each state behavior is chosen for a given decision-state. The results are determined by sampling each state and recording the action that results from the maximum average of the Q tables or the most likely action specified by the top 10 individuals from the converged solutions for the RL algorithm and GA, respectively.

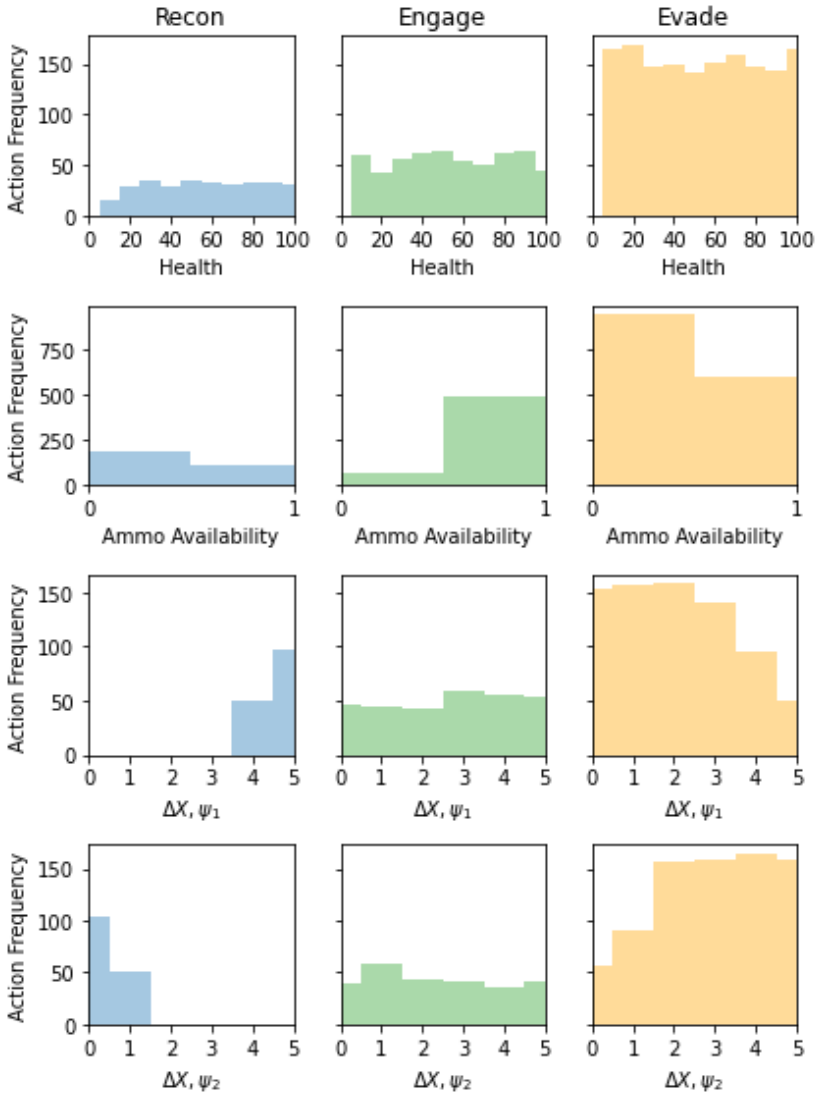

Figure 5.16: The RL algorithm's decision frequency for each decision-state.

The three actions, reconnaissance, engagement, or evasion, are plotted against different

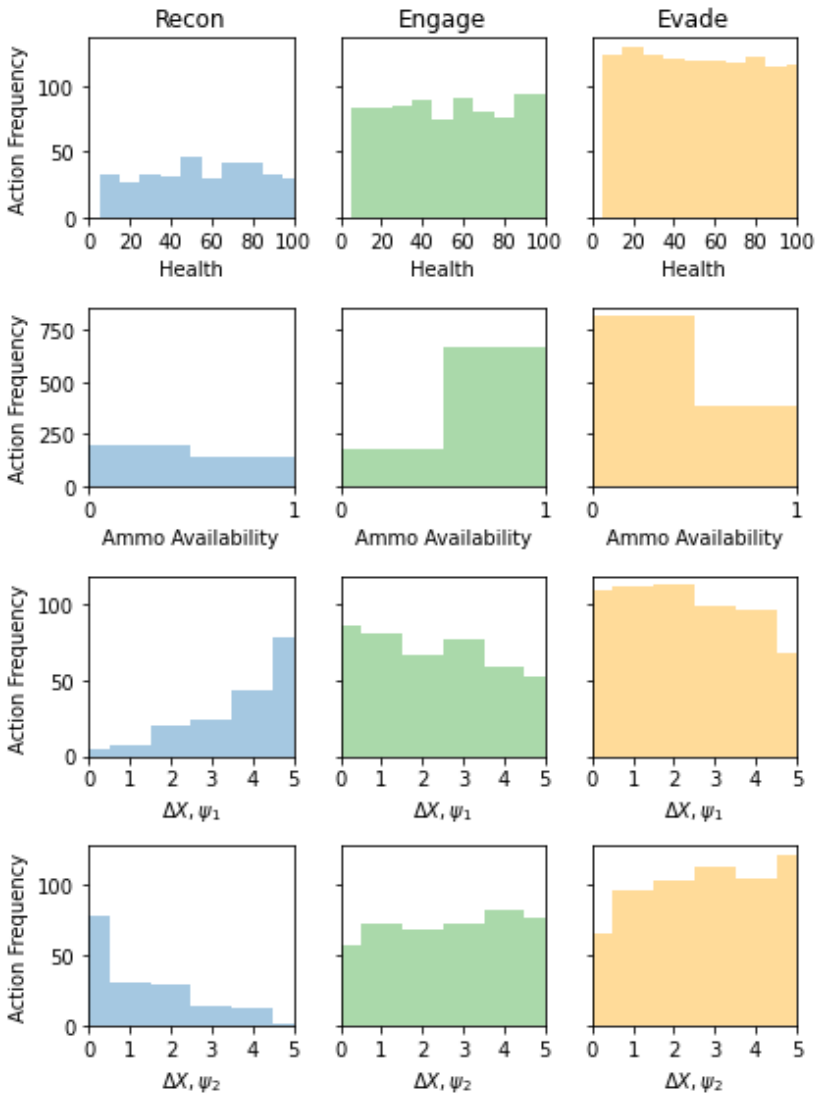

Figure 5.17: The GA algorithm's decision frequency for each decision-state.

instances of the four discrete state variables, health, ammunition availability, and the two distances ΔX to the defender with respect to the two relative headings ψ_1 and ψ_2. The state behavior histograms for both algorithms favor the evasion state behavior more than engagement while both the evasion and engagement behavior is favored over reconnaissance. This finding is expected since reconnaissance never results in a higher reward than either the evasion or engagement behavior. The engagement behavior is almost only selected when the agent has available ammunition while the agent favors evasion once the available ammunition equals zero. The selection of engagement versus ammunition availability verifies that

the algorithms are learning expected behaviors. The choice of engagement when ammunition availability is zero illustrates that the algorithms may not be at the global optimum, but the RL algorithm's behavior is more optimal for engagement due to the reduced preference to engage when no ammunition is available.

The final two rows of Figures 5.16 and 5.17 illustrate the agent's distance to the defender for the two possible relative headings the agent could encounter. The two possible relative headings are illustrated in Figure 5.12. The RL algorithm seemed to only prefer the reconnaissance state behavior when the agent was passing the defender near the tangent of its field of view. This result was expected and provides confidence in the learned behavior of the RL algorithm. The GA seemed to learn the same behavior, but it was not able to perfectly learn that a path near the defender's center resulted in a high penalty due to the incurred damage. These findings suggest that RL is able to find a more optimal solution for the defined decision-state space.

Further comparison in the optimality of each algorithm can be conducted through the quantification of mission effectiveness. Each algorithm type is used for an agent's decision behavior for the contested reconnaissance mission. The sole MoE used to quantify mission effectiveness is the agent's health, which is analogous to survivability. Each of the five randomly seeded solutions for each algorithm type was tested on 20 stochastic missions. The same set of 20 missions are used for each algorithm's random seed pair. In total, each algorithm type was tested on 100 stochastic missions resulting in an average of 73.15 and 63.35 health for RL and GA, respectively. Figure 5.18 illustrates one mission simulation where the RL algorithm's decisions resulted in a final health of 75 while the GA's decisions resulted in a final health of 40 for the same mission environment.

The higher convergence reward and mission effectiveness achieved by the decision behavior RL algorithm suggest that it is a more optimal algorithm to use for this problem. The decision-state histograms verified that both algorithms learned similar behaviors that were also expected behaviors. The RL algorithm was also determined to be more sample

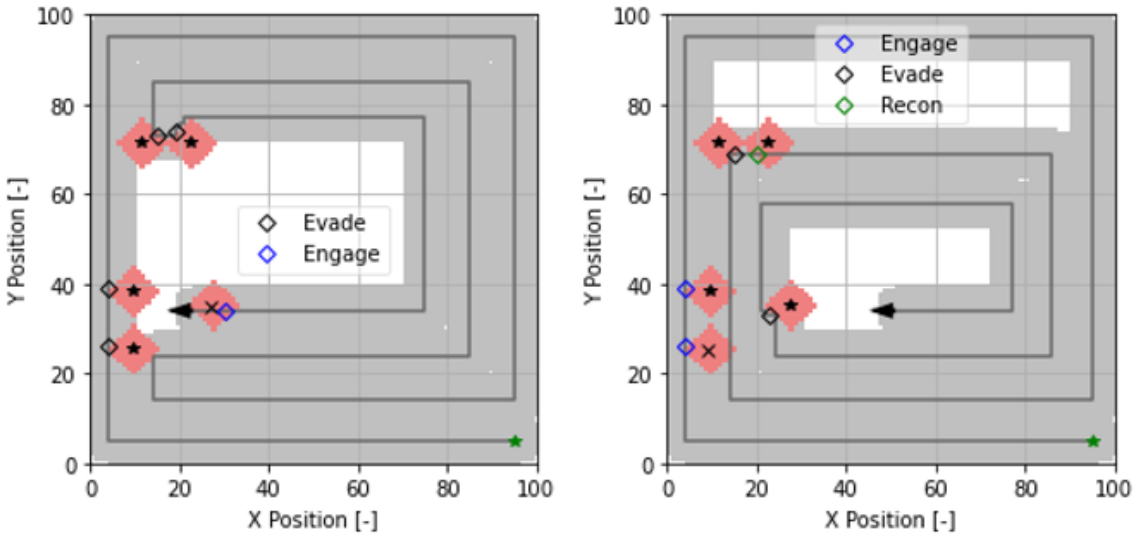

Figure 5.18: Overall mission simulation of the decision behavior RL algorithm (left) and GA (right).

efficient than the GA. These findings inform a conclusion for Hypothesis 1.

Research Conclusion 1*: The use of reinforcement learning to formulate the decision behavior algorithm efficiently balances optimality of a minimally defined mission.*

5.3 The Continuous Contested Reconnaissance Mission

The discrete contested reconnaissance mission provided a simple mission environment to explore the application of the decision behavior algorithm. However, a more complex mission action space is required to explore alternatives for aerospace technologies. The previously developed mission can be extended from a discrete state variable space to a continuous one. The continuous state variables, such as the agent's heading or bank angle, are continuous while the possible actions remain discrete. The actions within the decision-state space still remain discrete state behaviors while the actions in the state-action space

change from primitive motions (forward, left, & right) to discrete changes in the agent's heading. The continuous contested reconnaissance mission will be defined in this section along with the agent's equations of motion and partially observable framework.

The continuous contested reconnaissance mission utilizes the same building blocks as the discrete version. Five defenders are randomly placed throughout the map at unknown locations. Their influence extends only as far as their field of view allows. The baseline field of view for each defender is a radius of $R_{def} = 80$ meters. The agent incurs damage when it is within a defender's field of view at a rate that is a function of the distance the agent is from the defender, ds_{def}. The damage per time step, D, is defined by equation 5.4.

$$D = 1 - (\frac{ds_{def}}{R_{def}})^2 \tag{5.4}$$

The discrete map is enlarged with arbitrary units of 1000×1000 meters. Figure 5.19 illustrates an example of the initial mission map where the green star is the agent's starting location while the black stars and red circles depict the defender locations and field of view, respectively. The agent initially has no knowledge of each defender's location. This mission is used as the baseline mission environment moving forward.

The ABM for the continuous contested reconnaissance mission is also developed in the Python coding language and the PyTorch package for neural networks. The continuous mission environment carries over some assumptions from the discrete one in that it is modeled as a 2-dimensional area where the defenders are assumed to be located on the ground while the surveying agent is flying at a constant altitude and speed above the defenders.

5.3.1 The Continuous Agent Model

The agent's position and velocity can be defined using Dubin's kinematic equations [166]. Equation 5.5 defines the agent's x and y position as a function of the velocity, v, and the agent's heading, ψ. Also, the flight path angle's rate of change is defined as a function

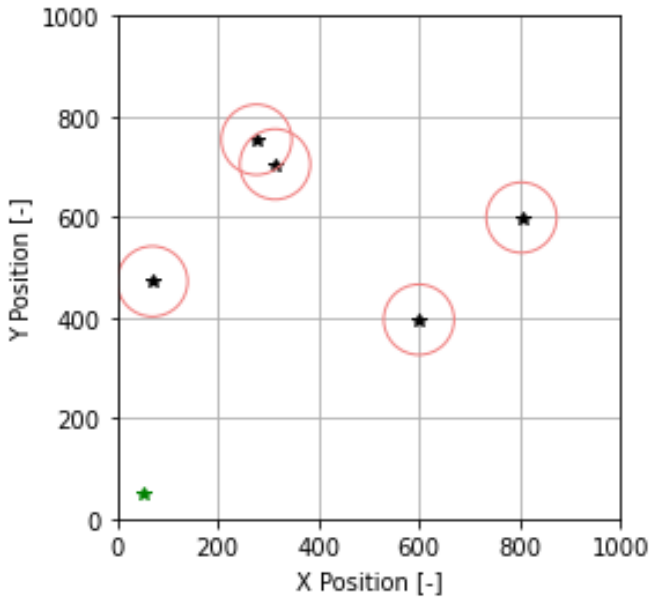

Figure 5.19: The baseline mission environment with 5 defenders.

of gravity, g, the agent's velocity, and the agent's bank angle, ϕ. The kinematic equations assume steady-level flight at a constant velocity [102].

$$\dot{x} = v\cos\psi, \quad \dot{y} = v\sin\psi, \quad \dot{\psi} = \frac{g\tan\phi}{v} \tag{5.5}$$

The bank angle is adjusted at every time step to control the agent's heading. The agent can increase or decrease its bank angle by $\Delta\phi$. The value of $\Delta\phi = \pm 5$ is chosen such that it is not too small or large to balance precise changes in bank angle and the agent's ability to quickly change bank angles [102]. The choice of $\Delta\phi = 0$ is not included as an available action for the agent to ensure the dimensionality of the state-action space remains manageable. The agent can maintain its current heading by switching back and forth between the two actions. The bank angle is adjusted at a frequency of 10 Hz.

These kinematic equations enable the use of continuous state variables such as ϕ and ψ. However, the increase in fidelity also increases the dimensionality of the state-action space significantly. The number of states an agent could be in is now infinite due to the continuous nature of the state variables. The use of a neural network is necessary for both

the state behavior and decision behavior to approximate the mapping of the infinite states to a set of discrete actions.

Environment Observation & Orientation

The POMDP problem is formulated using continuous state variables, of which the agent has perfect knowledge, and observation images, of which represents the source of imperfect information. The observation images map the surveyed area and known defender locations. Each observation image is a $100X100$ grid where each cell, c, is assigned a normalized value, $c \in [0, 1]$. The survey map assigns each cell that the agent has visited a value of 1, where a visited cell is defined as any cell that is currently or was within the agent's field of view. The baseline field of view for the agent is defined as a radius of $R_{A/C} = 50$ units. The defender map assigns each cell within the defender's field of view a value corresponding to the damage that would be incurred if the agent entered the cell. This map definition results in cell values close to 0 near the edge of the defender's field of view while a cell which contains a defender's location would have a value of 1. The overlap of defenders results in the overlapping cells to be assigned a value equal the summation of damage that would be incurred if the agent entered the cell up to a maximum value of 1. Overlapping cells that are equal to 1 may not reflect the true damage that would be incurred if the agent enters that cell. All cells that fall outside the true map boundary are assigned a value of 1. This is to discourage the agent from leaving the contested area.

The observation maps are generated by rotating and centering the true map relative to the agent's location and heading. This orientation step improves the agent's learned policy [102]. The observation maps are a 100×100 grid, where each cell represents a 10×10 square on the true map. Progression of the agent's observations are illustrated in Figure 5.20. The agent initially starts at the coordinates $x = 50$ and $y = 50$ with an initial heading of $\phi = 45$ degrees. Initially, there are no defenders in the agent's defender image because their locations are unknown to the agent. Once the agent enters a defender's field of view,

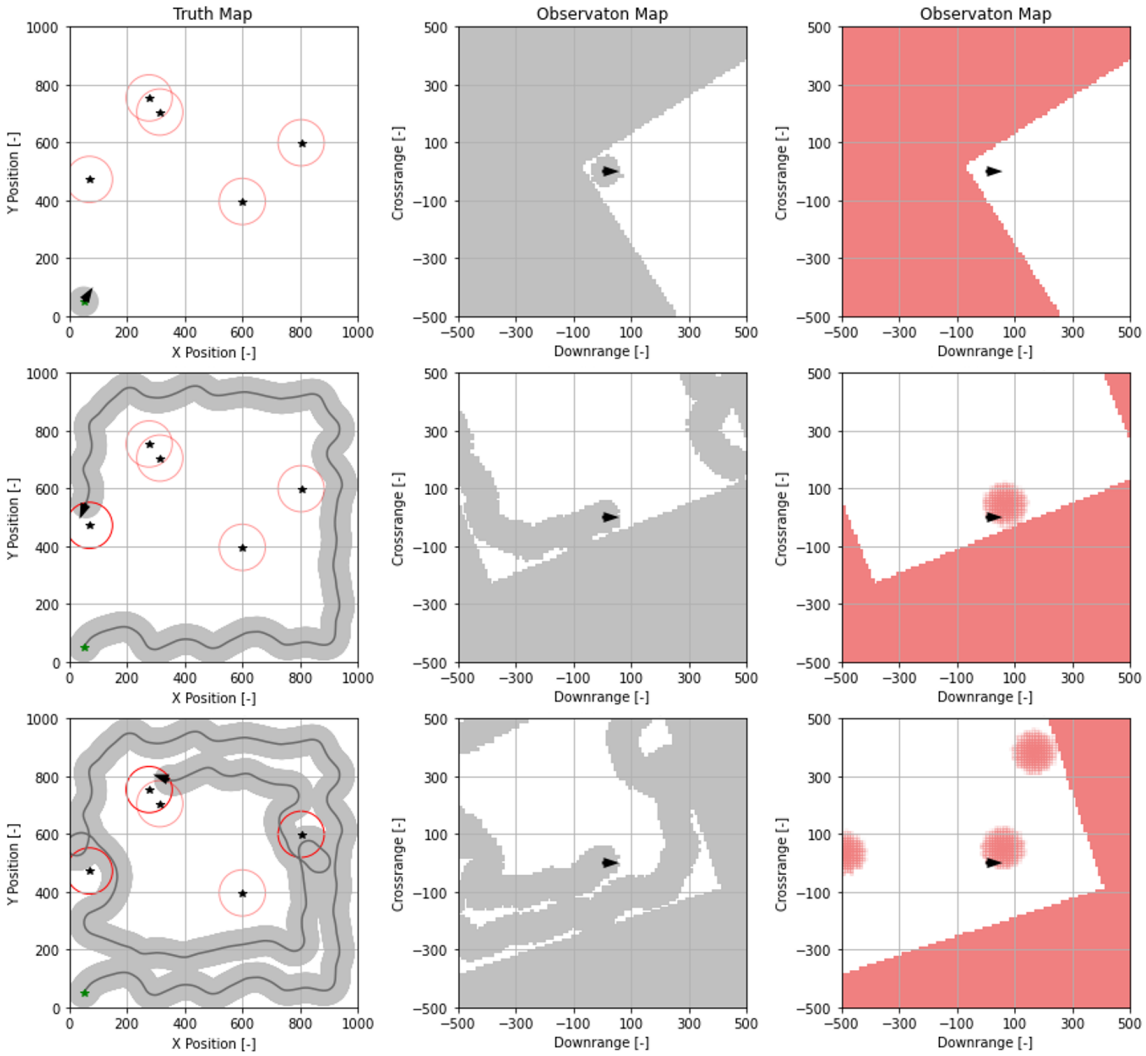

Figure 5.20: Progression of the true mission map and corresponding observation maps.

the defender's entire area of influence is known and updated accordingly on the observation image. Alternatively, if the agent's field of view is greater than the defender's, the agent gains full knowledge of a defender's location and area of influence once the defender is within the agent's field of view.

The incorporation of the observation maps during the formulation of the POMDP state machine is critical to include. The agent's ability to observe and orient itself during each time step allows it to constantly adapt to the mission environment. Agent observations, carried over from the OODA loop, allow the agent to learn better policies in a dynamic mission environment.

5.3.2 The Deep Decision Behavior Algorithm

In order to map continuous state variables and observation maps to actions, the decision-state and state-action spaces need a NN to approximate the agent's policy. The NN architecture must be defined in such a way as to combine both continuous state variables and observation maps to inform action selection. The dimensionality of both the action spaces requires deep learning. The identification of an RL algorithm that can efficiently handle deep learning is first identified during the formulation of a deep decision behavior algorithm. The architecture of the NN is then outlined to capture both continuous state variables and observation maps.

Deep Reinforcement Learning

The identification of a deep RL (DRL) algorithm is required to address the complexity of the decision-state space. The double Q-learning algorithm outlined in Section 5.2.2 utilized the ϵ-greedy policy, which is *quasi*-deterministic. Similar Q-learning algorithms have been successfully implemented on problems using a POMDP for the state-action space [102]. However, as the problem becomes more non-markovian, a deterministic policy's guarantee to produce an optimal policy does not hold [146]. Identification of a DRL algorithm that utilizes a stochastic policy is desired to learn optimal policies for problems using a POMDP.

Deep Q-learning (DQL) not only carries over the same action value overestimation it had with a tabular policy, but its use with a non-linear function approximator, such as a NN, can cause the algorithm to diverge [167]. Divergence of the DQL algorithm during training commonly occurs due the algorithm's method of updating and evaluating its policy using the same Q-network [144]. Many of the proposed techniques to address overestimation and policy divergence have been successfully implemented for DQL on MDPs with non-linear function approximators [158]. The *quasi*-deterministic ϵ-greedy policy has also found success in problems with POMDPs; however, a stochastic policy is desired due to its flexibility in action selection and potential to produce an arbitrarily better policy than a

deterministic one [145]. In addition, an algorithm that is robust and data efficient would enable the deep decision behavior algorithm for a complex decision-state space.

Recent developments in policy gradient RL algorithms have shown effective optimization of large NNs [150]. However, some model-free RL algorithms, such as Trust Region Policy Optimization (TRPO), tend to have a high sample complexity when using large NNs [168]. The popular Proximal Policy Optimization (PPO) algorithm builds on knowledge gained from TRPO to develop methods that maintain their stability and reliability while providing better performance with a much simpler implementation [151]. Two key aspects of PPO are its surrogate objective function, L, and its policy update method, which performs stochastic gradient ascent over K epochs. The clipped surrogate objective function, L^{CLIP}, is shown by equation 5.6.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right], r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (5.6)$$

where π_θ is a stochastic policy, θ is the vector of policy parameters, $r_t(\theta)$ denotes the probability ratio, $\hat{A}_t$ is the advantage function, and ϵ is a hyperparameter. The hyperparameter is selected as $\epsilon = 0.2$ based on the results presented in [151]. The surrogate objective uses a pessimistic bound by always taking the minimum of unclipped and clipped objective function. Finally, the empirical average is taken over a batch of samples, denoted by $\hat{\mathbb{E}}_t$.

The clipped surrogate objective function adds two additional terms to improve sampling efficiency. The first is an entropy term, S, that encourages exploration. The second term incorporates a loss function that enables parameter sharing between the policy and value network. These additional terms are combined into the final surrogate objective function as defined by equation 5.7

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right] \quad (5.7)$$

where $c_1 = 0.5$ and $c_2 = 0.005$ are coefficients, L_t^{CLIP} is the clipped surrogate objective

function at time t, and L_t^{VF} is the mean-squared loss at time t. The decay of the entropy coefficient, c_2, can significantly influence the performance of PPO [169]. The entropy decay is similar to methods that smooth the surrogate objective function to enable easier optimization early in training and then gradually increases the functions complexity as the algorithm learns more about the solution space [170]. Smoothing of the objective function can also enable the use of a larger learning rate, α, which can speed up convergence. The entropy is decayed from the initial value of $c_{2,i=1} = 0.005$ to a minimum value of $c_{2,\min} = 0.001$ at a rate $\beta = 0.996$. The entropy decay follows the equation $c_{2,i} = \max(\beta c_{2,i-1}, c_{2,\min})$.

The final surrogate objective enables the use of parameter sharing, which helps to reduce the number of policy parameters required to learn for large NNs. The advantage estimator, $\hat{A}$, rolls up the discounted rewards from the value network over a finite sample set from time step t to T. Equation 5.8 defines the advantage estimator.

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + ... + \gamma^{T-t+1} rT - 1 + \gamma^{T-t} V(s_t) \tag{5.8}$$

where the discount factor is to $\gamma = 0.99$. Implementation of the algorithm also utilizes a learning rate of $\alpha = 0.001$. The PPO method, outlined by algorithm 4, collects NT samples each iteration and optimizes the surrogate objective function for K epochs using Adam [171]. For the single agent contested reconnaissance mission, $N = 1$. After the optimization steps are complete, the policy network parameters are updated using a soft update method similar to target network updates used for Q-Learning [158, 172]. Soft network updates use a target value smoothing coefficient, τ, to reduce instabilities during training [173]. The value network parameters, θ_{old}, are updated with a smoothing coefficient of $\tau = 0.05$.

The selection of PPO for the deep decision behavior algorithm provides a simple yet sample efficient algorithm to map the complex decision-state space. This algorithm is considered state-of-the-art within the RL community; however, further experiments with PPO

Algorithhm 4 PPO, Actor-Critic Style [151]

for i=1,2,... **do**
  for actor=1,2,...,N **do**
    Run policy $\pi_{\theta_{old}}$ in environment for T time steps
    Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{old} \leftarrow \theta_{old} + \tau(\theta - \theta_{old})$
end for

have continued to improve the algorithm's robustness [174]. Implementation of the PPO algorithm is done using PyTorch and builds off of an open source python code for a simple NN architecture [175]. The field of RL is rapidly advancing, so many more algorithms that provide a better solution for the decision behavior algorithm may be proposed, but PPO is determined to be more than sufficient for this proof-of-concept implementation of the decision behavior approach.

Deep Neural Network Architecture

The appropriate NN architecture is determined through evaluation of the inputs for both the decision and state behaviors. The immense size of both the decision-state and state-action spaces requires the use of a Deep NN (DNN). Recent advances have enabled the use of DNNs to efficiently map raw input data to outputs, automatically capturing complex behaviors by identifying features at various abstraction levels [176]. DNNs are typically those which have more than three hidden layers. Determining the number of layers can be difficult since too few layers can be detrimental to learning while too many layers can be too computationally expensive to train.

The DNN architecture chosen for the decision and state behaviors are identical in structure, except for the number of inputs fed into each network. The structure of the DNN, illustrated in Figure 5.21, is comprised of a set of fully connected layers, illustrated in Figure 3.16, and a set of convolutional layers that capture the continuous state variables and the observation image(s), respectively. The reconnaissance and evasion state behaviors in-

put two continuous variables into the fully connected layers, denoted by solid lines, while the decision behavior inputs all four variables depicted. The state variables are the agent's bank angle, ϕ, the agent's heading, ψ, and amount of ammunition available, A, and the agent's health, $H = 100 - D$. This information passes through two hidden layers, each with 100 neurons, before merging with the outputs from the convolutional layers.

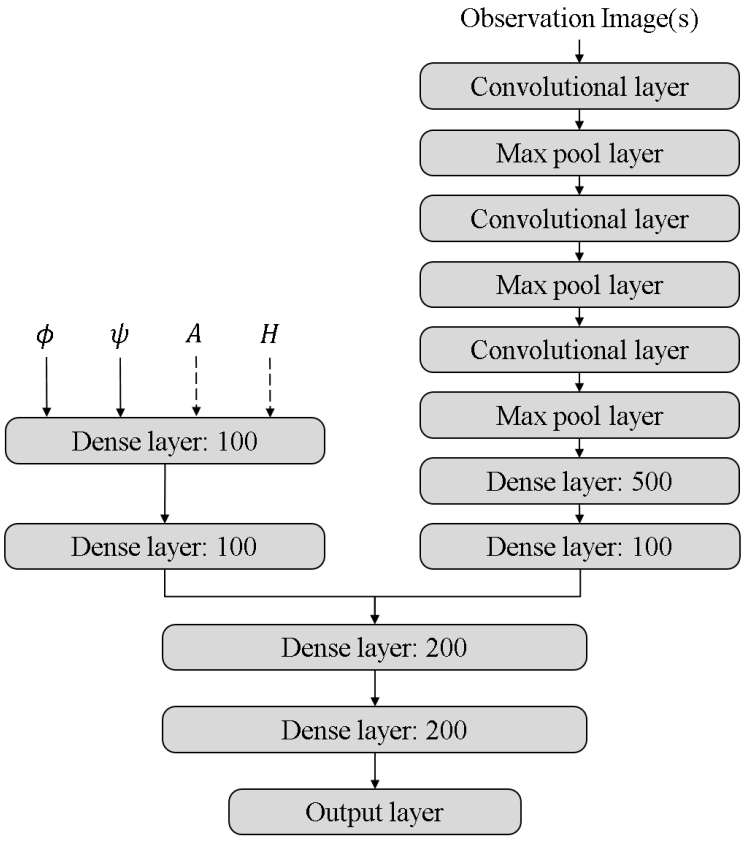

Figure 5.21: Neural network architecture modified from [102].

The observation image(s) are input into the convolutional NN (CNN) as raw 100×100 arrays of binary numbers. The reconnaissance behavior inputs both the survey and defender observation maps or images as two separate channels while the evasion behavior only inputs the defender observation image into a single channel. CNNs are a type of NN that have a reduced set of neuron connections and parameters compared to feedforward NNs [177]. The reduction in size contributes to the grouping of neighboring pixels which

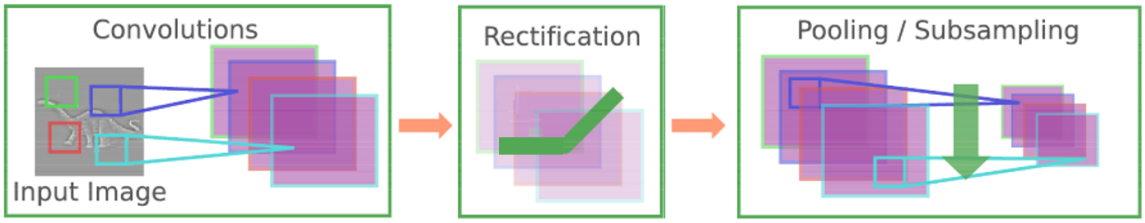

Figure 5.22: Notional convolutional layers including the convolution, rectification, and pooling layers. Modified from [178].

aims to identify features within each grouping. This grouping of pixels allows CNNs to be more easily trained for high dimensional object recognition tasks [102].

The CNN is comprised of three sequential sets of a convolutional layer, rectification layer, and max pooling layer. Each set, illustrated in Figure 5.22, conducts feature extraction from the image (initial layer) or previous CNN layer. The convolutional layers use a 3×3 kernel size, which defines the grouping of pixels, a stride length of 1, which defines the overlap of each group, and a padding of 0, which adds zero blank pixels to the images edge. The rectification layer utilizes a Rectified Linear Unit (ReLU) activation function. This choice of function has been very successful in improving the training time of CNNs [177], but also worth noting is that the $\tanh$ activation functions may provide better performance [157]. The max pooling layer reduces the number of parameters needed for sequential layers by taking the maximum value in 2×2 non-overlapping groups of pixels. The CNN reduces the input image(s) from a size of $100 \times 100 \times n_{ch}$ to $10 \times 10 \times n_{ch}$ where n_{ch} is the number of channels or images.

Once the observation image(s) are passed through the convolutional layers, their outputs are flattened and passed through two fully connected layer of 500 and 100 neurons, respectively. The two sets of data – one from the continuous state variables and one from the observation image(s) – are combined and passed through two additional fully connected layers with 200 neurons each.

The selection of the PPO RL algorithm utilizes a policy and value NN. The use of two NNs doubles the number of parameters that are needed. In order to manage the total size of

the problem, just the CNN parameters are shared between the two NNs. This is acceptable thanks to the mean-squared loss term in the surrogate objective function which enables parameter sharing between networks. The policy output layer returns the categorical action log probabilities while the value output layer uses a softmax activation function to return a normalized conditional probabilities of the state values [179].

Overall, the policy and value NNs have a total of $3,386,975$ parameters for each state behavior and the NNs for the decision behavior algorithm have $3,387,476$ parameters. The number of parameters is acceptable for the initial implementation of the decision behavior approach. Improvements in the architecture were not investigated due its success modeling a similarly sized state-action space for UAV wildfire surveillance [102].

Deep State Behavior Logic

The increase in fidelity of the contested reconnaissance mission enables the inclusion of state behavior algorithms that must be defined and trained before the decision behavior algorithm. Similar to the discrete contested reconnaissance mission, the three state behaviors of the agent are defined in Table 5.1. The logic used for each state behavior must be defined along side a representative training mission. Once the state behaviors are defined and trained, the decision behavior algorithm can be trained using the state behavior algorithms.

Reconnaissance. The reconnaissance state behavior seeks to learn how to efficiently survey the contested area while avoiding known defender locations. The definition of the representative mission must ensure the agent can uniformly sample the state-action space. This state behavior is given inputs of the two observation maps along with two continuous state variables. The observation maps enable the agent to see where it has explored and where defenders are located at. The agent's continuous state variables are its true heading and bank angle which are contained in the intervals $\psi = [0, 360)$ and $\phi = [-50, 50]$ degrees. Each continuous variable is standardized to the interval $[0, 1]$ to help convergence of the NN.

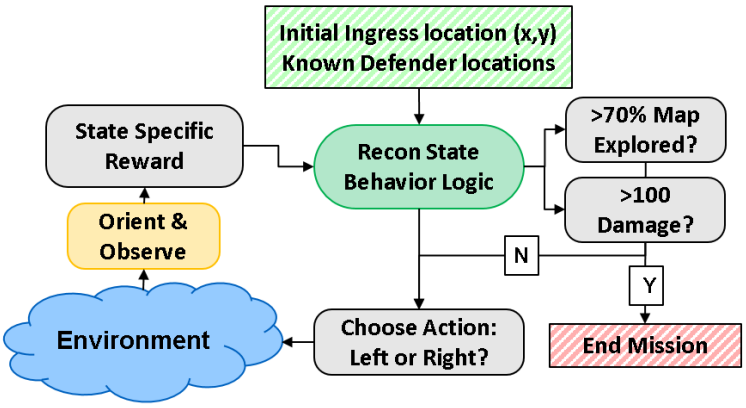

Figure 5.23: The agent's reconnaissance state behavior logic for the continuous mission.

The initial condition of the reconnaissance state behavior includes an initial coordinate and the location of all defender locations. Inclusion of the defender's locations in the agent's initial knowledge allows the policy to learn to avoid known defenders and move towards areas of the map that have not yet been surveyed. The reconnaissance agent's mission logic is illustrated in Figure 5.23. Once the mission begins, the agent executes the steps within the logical loop each time step. The agent first selects between two actions that correspond to increasing or decreasing the agent's heading by 5 degrees. The agent then takes the selected action in the mission environment and observes any changes. Changes within the environment include updating the agent's position on the true map, recording any newly visited cells, and updating the observation maps. The resulting reward is then fed back into the algorithm for learning. Once the reconnaissance agent surveys 70% of the true map or incurs 100 damage, the mission will end.

The reconnaissance agent receives three rewards after each time step. The agent receives a reward proportional to the damage incurred over one time step and the number of new cells visited. However, if the agent leaves the true map's boundary, it will receive a penalty, λ_3, for every time step until it returns. These rewards encourage mission paths that avoid a known defender's field of view, move towards unexplored regions, and stay within the limits of the true map. Definition of the rewards can be difficult due to the potential of conflicting reward signals such as the penalty of entering a defender's field of view while

receiving a reward for visiting unexplored cells. The set of reward equations, defined by equations 5.9, 5.10, and 5.11, must be tuned using the tuning parameters, λ_1, λ_2, and λ_3.

$$r_1 = -\lambda_1(D_t - D_{t-1}) \tag{5.9}$$

$$r_2 = \lambda_2 \left[\sum_{c \in C | c=1} M_t(c) - \sum_{c \in C | c=1} M_{t-1}(c) \right] \tag{5.10}$$

$$r_3 = \begin{cases} -\lambda_3 & \text{if } x \notin [0, 1000] \text{ or } y \notin [0, 1000] \\ 0 & \text{else} \end{cases} \tag{5.11}$$

where t denotes the current time step, c is a cell in the set of cells C, and $M(c)$ is the binary value from the true survey map of cell c. The tuning parameters are used to adjust each reward to an order of magnitude equal to ± 0.1. The tuning parameters are tuned to values of $\lambda_1 = 0.05$, $\lambda_2 = 0.7$, and $\lambda_3 = 0.1$. During training and testing, the agent starts at the coordinates $(x, y) = (50, 50)$ with an initial heading chosen using a random uniform distribution from the interval $\psi_0 \in [0, 90]$ degrees and an initial bank angle of $\phi = 0$ degrees.

The reconnaissance behavior algorithm is trained for $5,000$ training episodes to reach convergence. Each episode has a max iteration of $4,000$ time steps. The algorithm stores the agent's state, action, reward, and next state at each step. Once the algorithm collects $1,000$ samples, it will update its policy with minibatches of $M = 200$ over $K = 5$ epochs. The reconnaissance behavior algorithm's convergence for the baseline alternative, illustrated in Figure 5.24, required approximately 17 hours to complete all training episodes. The reconnaissance state behavior algorithm is trained using 5 separate random sample seeds. The blue line illustrates the average metric from all 5 random seeds. The results are smoothed and averaged using Savitzky-Golay filtering with a window size of 301.

The cumulative reward and total time steps for each training episode are plotted along

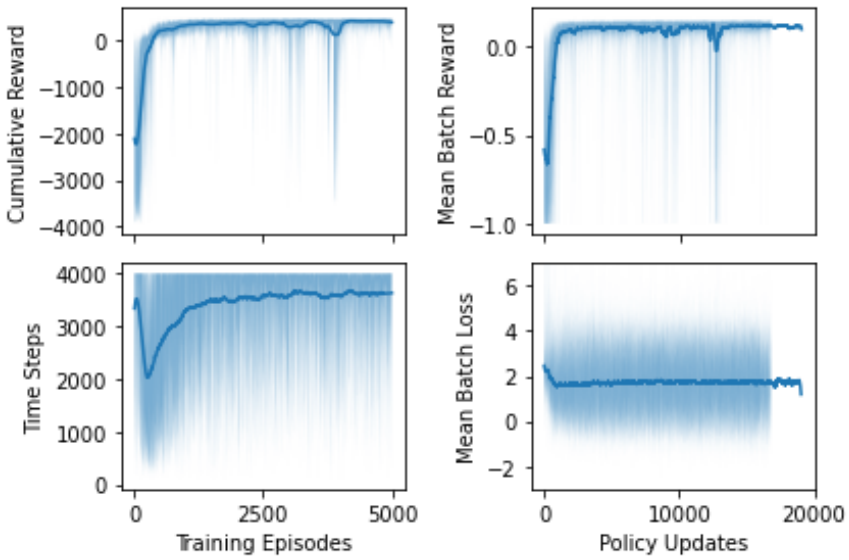

Figure 5.24: Convergence of the reconnaissance state behavior algorithm.

with the mean batch reward and loss for each policy update. The cumulative and mean batch rewards quickly converge towards an optimal reward; however, the time steps do not converge until after $2,500$ training episodes which illustrates that the algorithm is still learning. The loss for each policy update stays relatively constant throughout training and does not diverge for any of the random sample seeds. These results illustrate that the reconnaissance behavior algorithm sufficiently converges. The NN from the random sample seed with the highest average reward across 200 mission samples is chosen to use with the decision behavior algorithm. The reconnaissance behavior from the top NN is illustrated in Figure 5.25. The test mission shows that the agent successfully explores the contested map while avoiding known defender locations.

Evasion. The evasion state behavior seeks to learn how to evade a defender given a bank angle, relative heading to the defender, and the defender observation map. The representative mission used to train the evasion behavior spawns the agent on the edge of a defender. The agent's distance from the defender is chosen to be the maximum of the agent's field of view or the defender's field of view. In order to sample the entire state-action space, the agent's bank angle, heading relative to the defender, and angular placement on

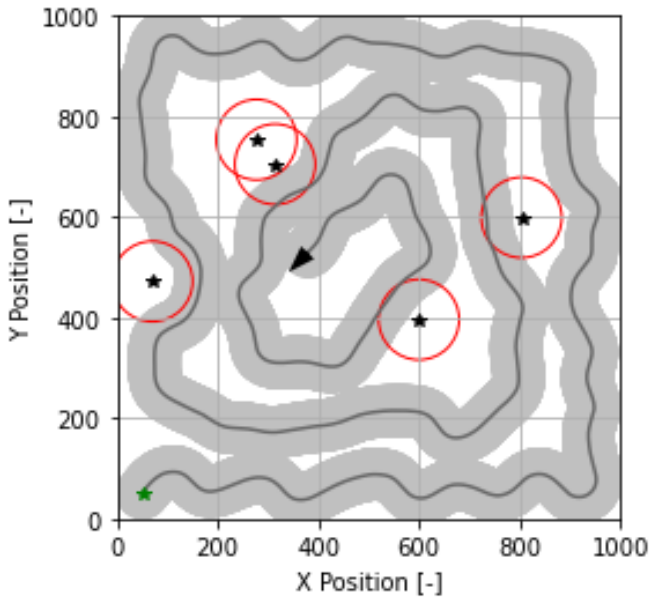

Figure 5.25: The reconnaissance state behavior surveying the baseline mission environment with 5 known defender locations.

around the defender is varied within the intervals $\phi_0 = [-50, 50]$, $\psi_{0,def} = (-90, 90)$, and $\theta_0 = [0, 360)$, respectively. The agent's relative heading relative to the defender is bounded by the interval $\psi_{def} \in [-180, 180)$ degrees.

Once the initial conditions of the agent are defined, the evasion behavior will choose between the two actions of increasing or decreasing its bank angle. The selection of the evasion behavior by the decision behavior algorithm prioritizes the survivability of the agent over the mission goals. The maximum change in bank angle during evasion is increased by a factor of $k_{\Delta\phi} = 1.2$ to reflect an increase in maneuverability due to the urgency of escaping the defender's field of view. The evasion's state behavior logic, illustrated in Figure 5.26, follows similar steps as the logic for the reconnaissance state behavior. The agent selects an action, performs the action in the environment, observes changes in the environment and updates its observation map, returns a reward based on from interacting with the environment, and then restarts the logical loop. After each loop, the agent checks to ensure its damage did not fall below 100 or if it has escaped the defender. Escaping the defender requires that the agent is outside of the defender's field of view and $|\psi_{def}| > 90$ degrees.

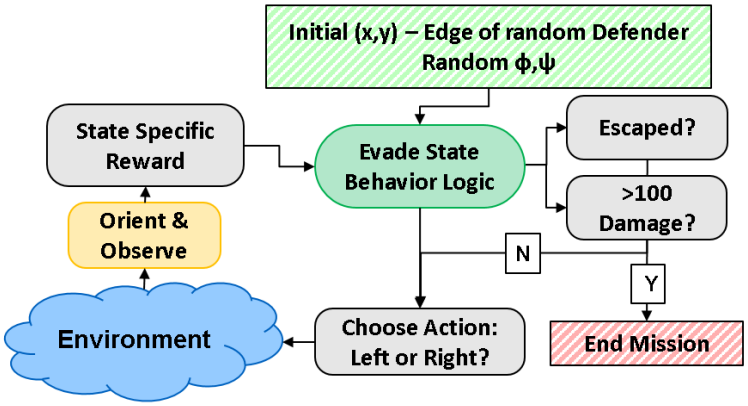

Figure 5.26: The agent's evasion state behavior logic for the continuous mission.

The evasion state behavior receives a single reward, r_1, after each time step. This sole reward prioritizes the agents survivability by encouraging policies that minimize total time steps required to escape the defender which also minimizes the incurred damage. The state behavior is trained over $15,000$ episodes. The maximum time steps of each episode is capped at 200. The algorithm is updated every 192 time steps which corresponds to $K = 3$ and a minibatch size of $M = 64$ samples. The convergence of the evasion state behavior, illustrated in Figure 5.27, is determined using five trained algorithms, each with a unique random seed. Average training time for each algorithm took 30 minutes. The cumulative reward, mean batch reward, episode time steps, and mean batch loss are averaged and smoothed using Savitzky-Golay filtering with a window size of 301.

Convergence of the evasion state behavior is considered sufficient due to the minimization of time steps and steady-state reward average. The loss is plotted to ensure the algorithm does not diverge during training. The behavior is tested using 200 stochastic missions to quantify its performance and compare it to the other random seed samples. The algorithm with the highest average cumulative reward is chosen to use with the decision behavior algorithm.

After convergence of each randomly seeded algorithm is reached, the behavior is checked to ensure the algorithm is behaving as expected. Many test cases of the evasion behavior are checked for the evasion of a single defender and evasion of two defenders. Two test cases

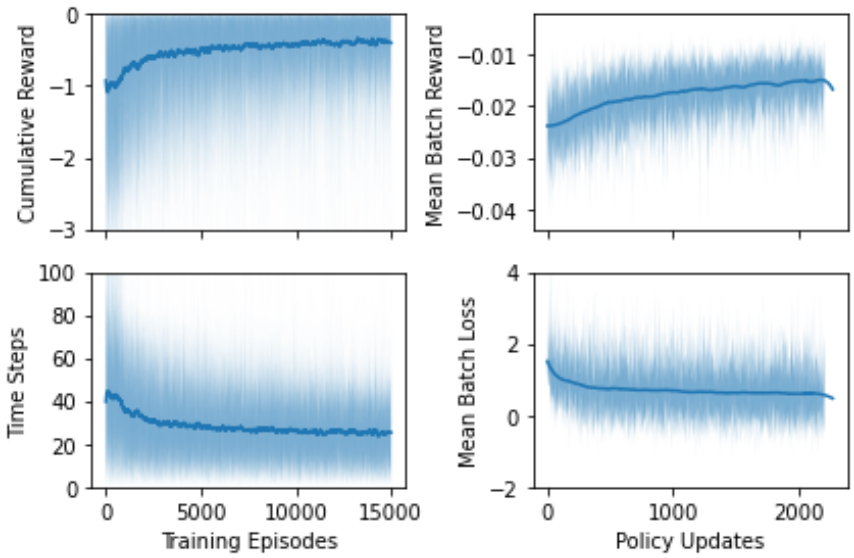

Figure 5.27: Convergence of the evasion state behavior algorithm.

are illustrated in Figures 5.28 and 5.29 for the single and double defender, respectively. In both test cases, the agent initially enters the defender's field of view with a positive bank angle and negative heading relative to the defender, ψ_{def}. The agent decides that continuing a counter-clockwise bank would result in a suboptimal evasion. The action to decrease its bank angle is chosen almost every time step to ensure the agent quickly reaches the maximum clockwise bank angle. The agent successfully evades the defender(s) in both test cases. The resultant behavior from the algorithm.

The evasion behavior algorithm successfully converged to a sufficient behavior in the defined environment. The resultant behavior made decisions that were expected given the initial state. These findings provide confidence that the evasion behavior is ready to use for the larger mission environment with the decision behavior algorithm.

Engagement. The behavior of engagement is chosen to be a single action as opposed to a behavior algorithm. Initial definition of an engagement behavior algorithm resulted in a significantly noisy algorithm due to the chance of neutralizing the defender in a single time step or running out of ammunition and evading the defender over the course of many time steps which resulted in rewards and episode lengths of different magnitudes. Selection

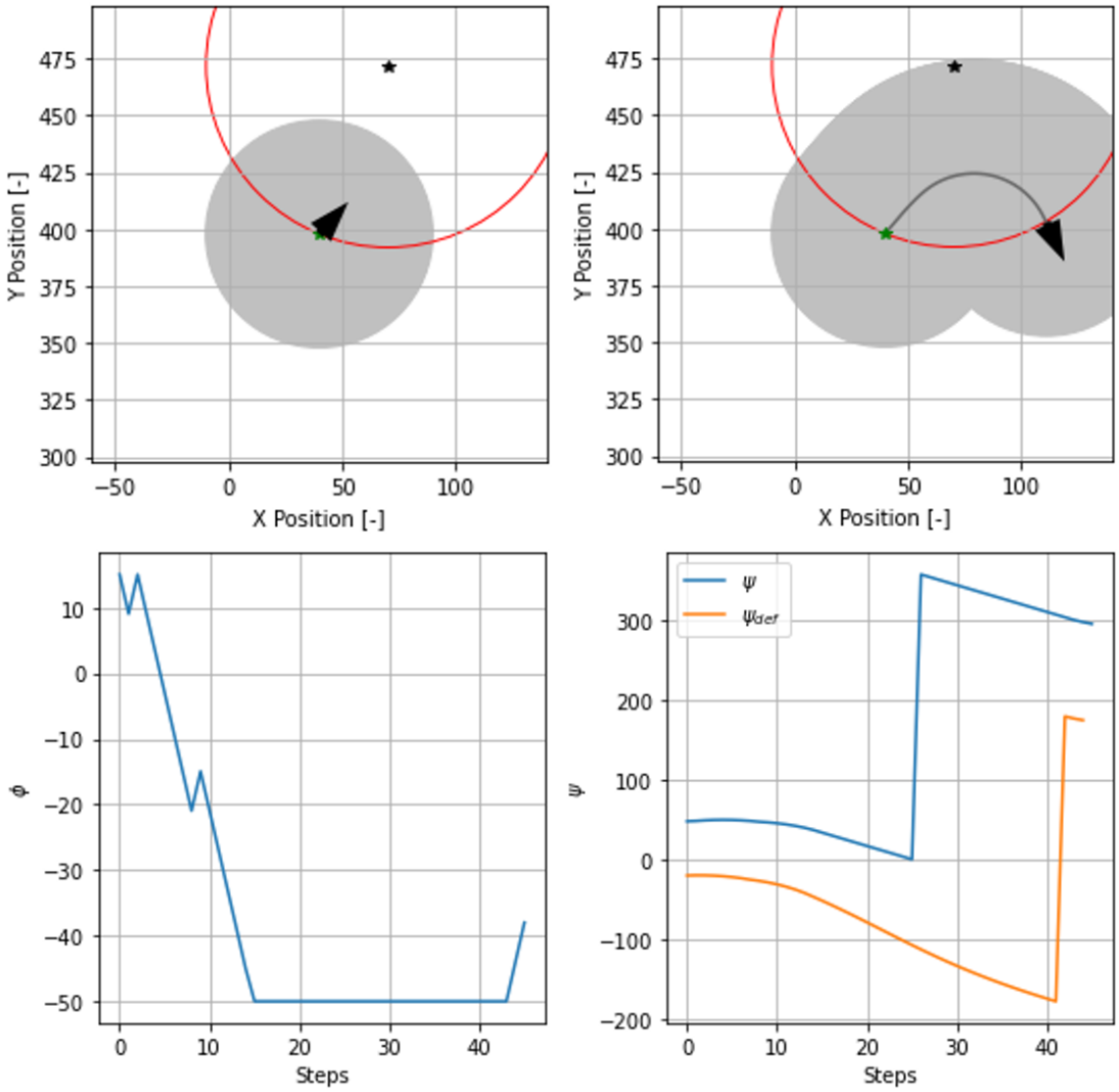

Figure 5.28: Initial and final state of the agent during the evasion of a single defender along with its corresponding flight path angles.

of the engagement behavior logic, illustrated in Figure 5.30, results in engagement of the defender over a single time step. The agent maintains its current bank angle and heading when this behavior is selected by the decision behavior algorithm. The defender is neutralized if a random uniform number drawn from the interval $[0, 1]$ is less than the agent's probability of kill, $P_k = 0.75$. If a defender is neutralized, it is removed from the defender map, allowing the agent to fly through its field of view without incurring any damage. Un-

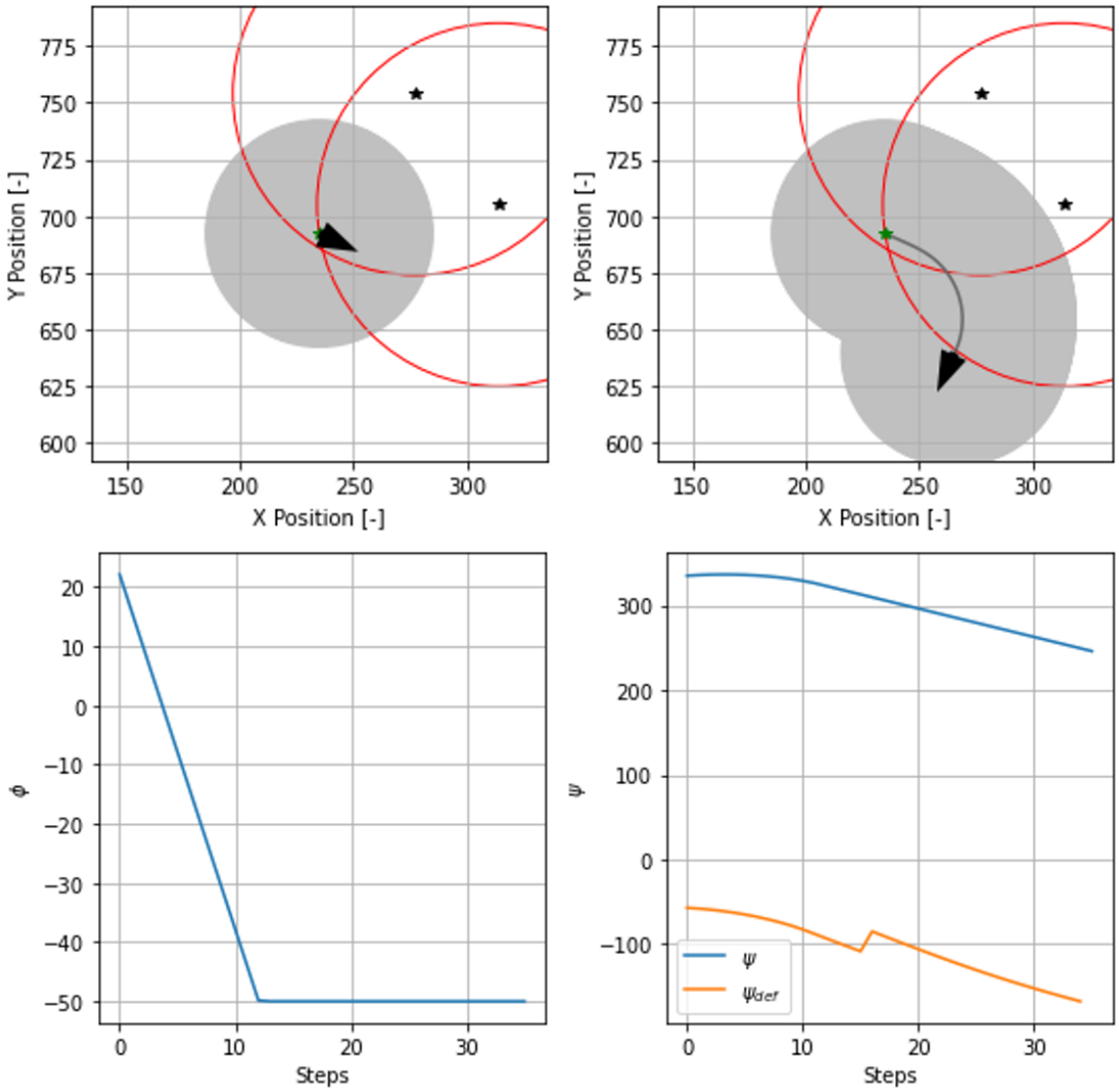

Figure 5.29: Initial and final state of the agent during the successive of two defenders along with its corresponding flight path angles.

successful engagement will not impact the defender at all. Once a single logical loop is complete, the agent will return to the decision behavior algorithm to make a new decision or return to reconnaissance.

The agent can only engage the defender when it is within the maximum of either the defender's field of view or the agent's field of view. The agent is provided a finite number of ammunition it can use for engagement during the entire mission. The baseline concept is

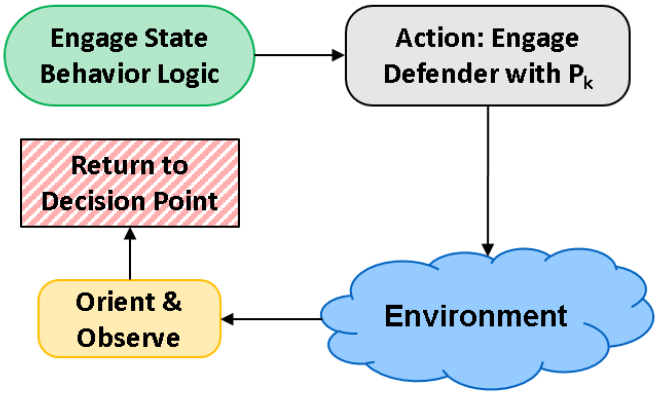

Figure 5.30: The agent's engagement state behavior logic.

equipped with an ammunition of 2 to provide some engagement capability, but prevent the agent from engaging every defender in the contested area. If engagement is selected when the agent has no more ammunition, no engagement will take place and the agent will return a penalty to the decision behavior algorithm. The simplicity of the engagement behavior provides the decision behavior algorithm with more critical decision points throughout the mission. The specific rewards linked to the engagement behavior are defined with the definition of the decision behavior algorithm's logic in the following section.

Deep Decision Behavior Logic

The decision behavior algorithm's logic, illustrated in Figure 5.31, is defined using the entire mission envelope as its representative training mission. The entire mission is chosen to provide multiple time step samples to be recorded for each training episode. The initial conditions are defined by initial starting coordinates $(x, y) = (50, 50)$ and an initial state behavior chosen to be reconnaissance. The agent conducts reconnaissance in the contested area without any knowledge of defender locations. Once the agent finds a defender, the decision behavior algorithm is triggered, requiring a new state behavior to be chosen. Once the chosen state behavior reaches its end criteria, the agent will either return to reconnaissance if a safe area is reached, or return back to the decision behavior to choose a new state

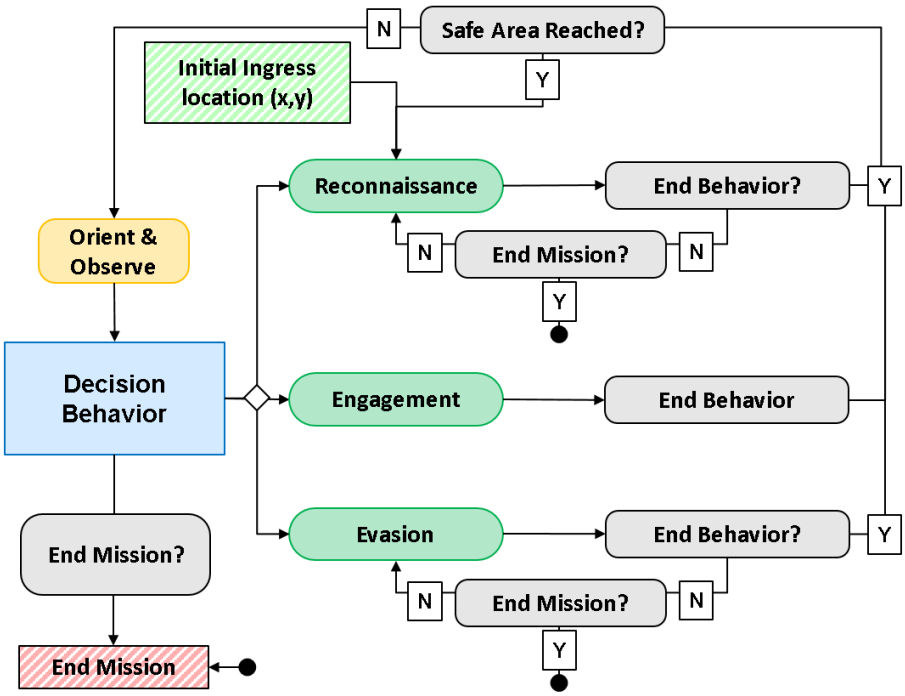

Figure 5.31: The agent's decision behavior logic for the continuous mission.

behavior. The decision behavior logic also uses both the survey and defender observation maps to observe and orient itself within the environment before each decision is made. Throughout the mission, the state behaviors and the decision behavior have the ability to end the mission depending on the agent's incurred damage or the mission's surveillance goal of 70% is achieved.

During the training of the decision behavior algorithm, a reward is returned after the completion of each state behavior as well as the completion of the entire mission. Completion of a the reconnaissance and evasion state behaviors is defined between the point at which a defender is initially discovered until the agent either finds a new defender within the region or completes the mission. This defines the evasion behavior mission segment as the evasion behavior required to evade the defender plus the reconnaissance behavior implemented after successful evasion. The decision behavior is given a reward, r_4, based on the change in damage during the mission segment normalized by the initial remaining

health of the agent, $H_i = 100 - D_i$.

$$r_4 = 1 - \frac{D_f - D_i}{H_i} \tag{5.12}$$

where D_f is the final incurred damage at the end of each mission segment and D_i is the initial damage already incurred by the agent at the start of the mission segment. The engagement behavior is provided three conditional rewards, defined by equation 5.13, based on the agent's available ammunition and drawn random number, N_{eng}. If the agent successfully engages the defender with an initial available ammunition of $A_i \geq 1$, than it is given a reward. If the the agent does not successfully engage the target with the same initial available ammunition condition, than it is neither rewarded or penalized. However, if the decision behavior algorithm chooses to engage the defender with an no ammunition available, it is given a small penalty to discourage this choice.

$$r_5 = \begin{cases} \lambda_{5a} & \text{if } A_i \geq 1 \text{ and } N_{eng} \leq P_k \\ 0 & \text{if } A_i \geq 1 \text{ and } N_{eng} > P_k \\ -\lambda_{5b} & \text{if } A_i \leq 0 \end{cases} \tag{5.13}$$

where the tuning parameters are defined as $\lambda_{5a} = 0.9$ and $\lambda_{5b} = 1$. Once the mission is complete, the decision behavior algorithm receives a final reward based on the average damage per defender found. The number of defenders found, $N_{def|found}$, can vary between each mission since the agent's goal is to just survey a certain percentage of the map. Equation 5.14 defines the decision behavior algorithm's final reward, which is at a higher order of magnitude than the individual decision rewards.

$$r_6 = 30 - \frac{D}{N_{def|found}} \tag{5.14}$$

The best performing state behavior algorithms combined with the engagement behavior

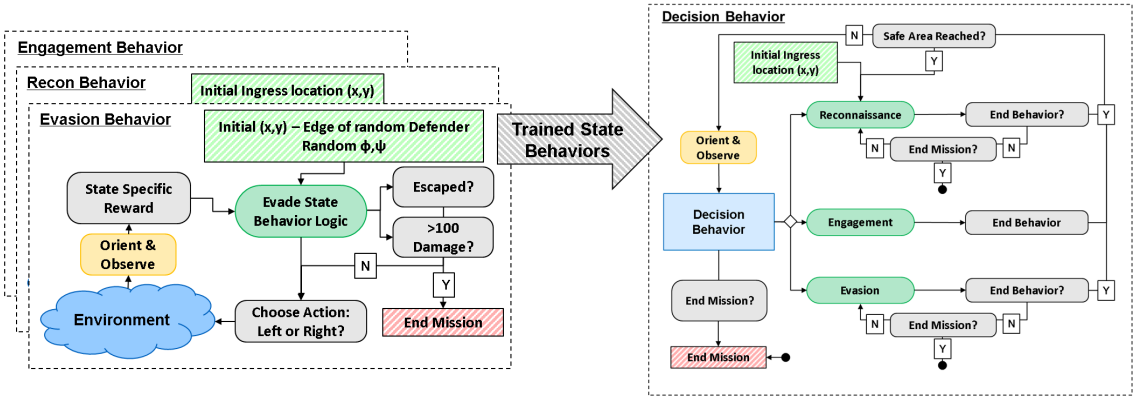

Figure 5.32: The training logic of the decision behavior algorithm.

logic is inserted into the decision behavior algorithm before training. The decision behavior algorithm then allows each state behavior algorithm or logic to 'drive' the agent between each decision point. The training logic, illustrated in Figure 5.32, requires that the state behavior algorithms be trained prior to the decision behavior algorithm's training.

Training of the decision behavior algorithm is conducted over $3,000$ training episodes. Each episode has a maximum number of $4,000$ cumulative state behavior time steps. The maximum number of decisions the agent can make is set to 15 to prevent the agent from getting trapped choosing engagement on every time step after its ammunition is depleted. The algorithm's convergence, illustrated in Figure 5.33, is evaluated for five random seed samples to ensure a satisfactory behavior is achieved. The required training time for each algorithm is approximately 7 hours. Additional training episodes did not noticeably improve the agent's policy. Each policy update during training uses a minibatch size of $M = 16$ and $K = 3$ epochs. Minimum values were used for both the minibatches and epochs to ensure the agent does not gather too much information between policy updates.

The decision behavior algorithms cumulative reward and mean batch reward converges to a steady-state value within the set number of training episodes. The convergence statistics are averaged and smoothed using Savitzky-Golay filtering with a window size of 101. The reward, time step, and loss statistics are sufficient in showing that the behavior converges. Now that the decision behavior algorithm has converged to a satisfactory point, the

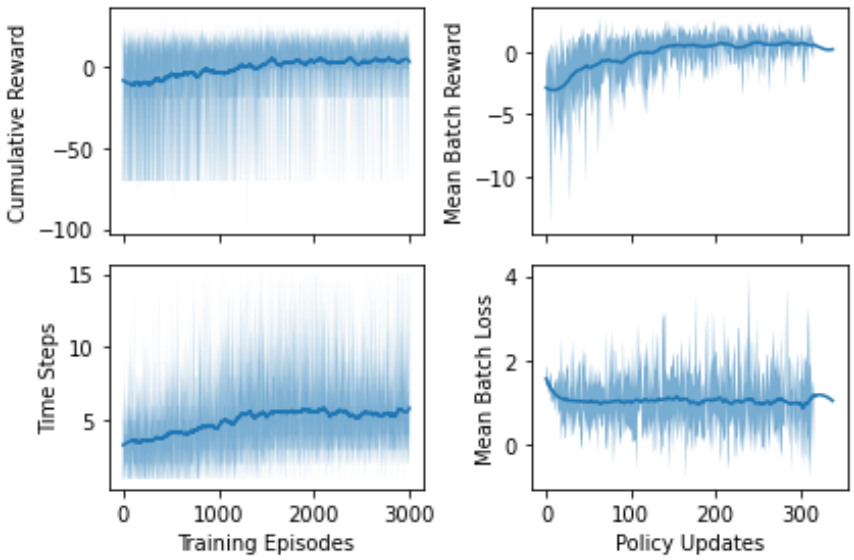

Figure 5.33: Convergence of the decision behavior algorithm for the continuous mission.

Table 5.2: Definition of the baseline technology alternative.

Baseline Alternative	**Performance Metrics**
Cruise Speed	20 m/s
Field of View	50 m
Defender's Field of View	80 m
Maneuverability Factor	1.2
Probability of Kill	0.75
Available Ammunition	2
Range	7,000 m

use of the DNN to approximate the decision-state space is evaluated to confirm of deny Hypothesis 2.1.

5.3.3 Evaluation of the Deep Decision Behavior Algorithm

The immense size of the decision-state space for partially observable systems requires a non-linear function approximator for the agent's policy. Application of a DNN architecture similar to that used for the state-action space was investigated for use with the decision-state space. The decision behavior algorithm was investigated to verify that the learned behavior successfully completes the contested reconnaissance mission. The baseline technology alternative's specifications are defined in Table 5.2 with arbitrary units.

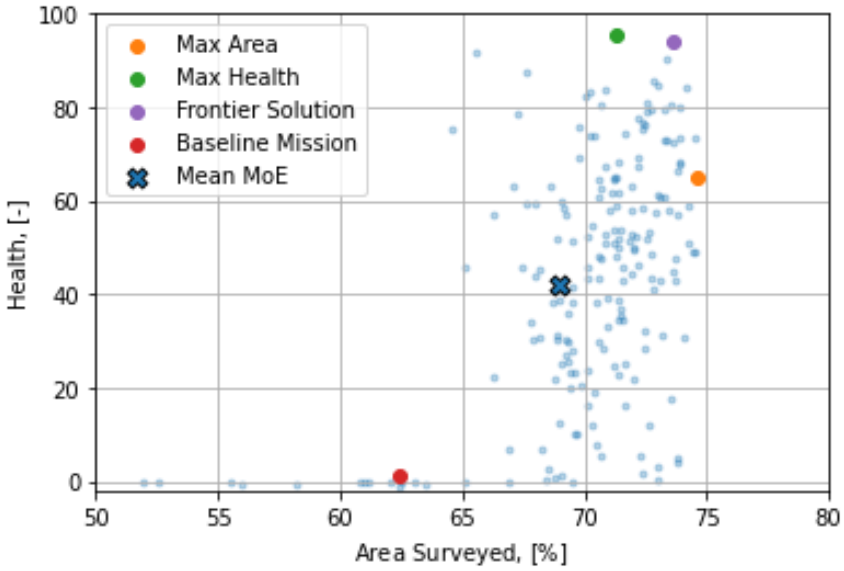

Figure 5.34: The agent's remaining health and area surveyed for each mission in the sample set.

The same set of 200 stochastic missions used to evaluate the reconnaissance behavior were also used to evaluate the baseline alternative's MoEs, illustrated in Figure 5.34. The quantification of the agent's mission effectiveness can now be evaluated by comparing the MoEs for each mission sample. The baseline agent successfully completed the mission by reaching its maximum range without incurring 100 damage for 176 out of the 200 mission samples. This corresponds to a mission survivability of 88%. The average remaining health of the baseline agent is a value of 41.91 ± 26.96 while the average area surveyed is $68.96\% \pm 7.19\%$ of the map. The agent's remaining health is found to be much more sensitive to the decisions the agent makes throughout the mission than the percent of the area surveyed.

The decisions the agent makes throughout the mission were investigated for specific solutions to further investigate the results. Four solutions were chosen from the set of 200 missions including three high performing solutions and a single failed solution. The mission path for each highlighted point is illustrated in Figure 5.35. The maximum health and maximum area surveyed solutions were selected to investigate the best solution of each MoE. The frontier solution is defined as the solution that lies on the Pareto frontier and is

chosen by determining the solution that provides the maximum equally weight combination of the MoEs. The last solution chosen is the baseline mission, which is illustrated earlier for the reconnaissance state behavior in Figure 5.25.

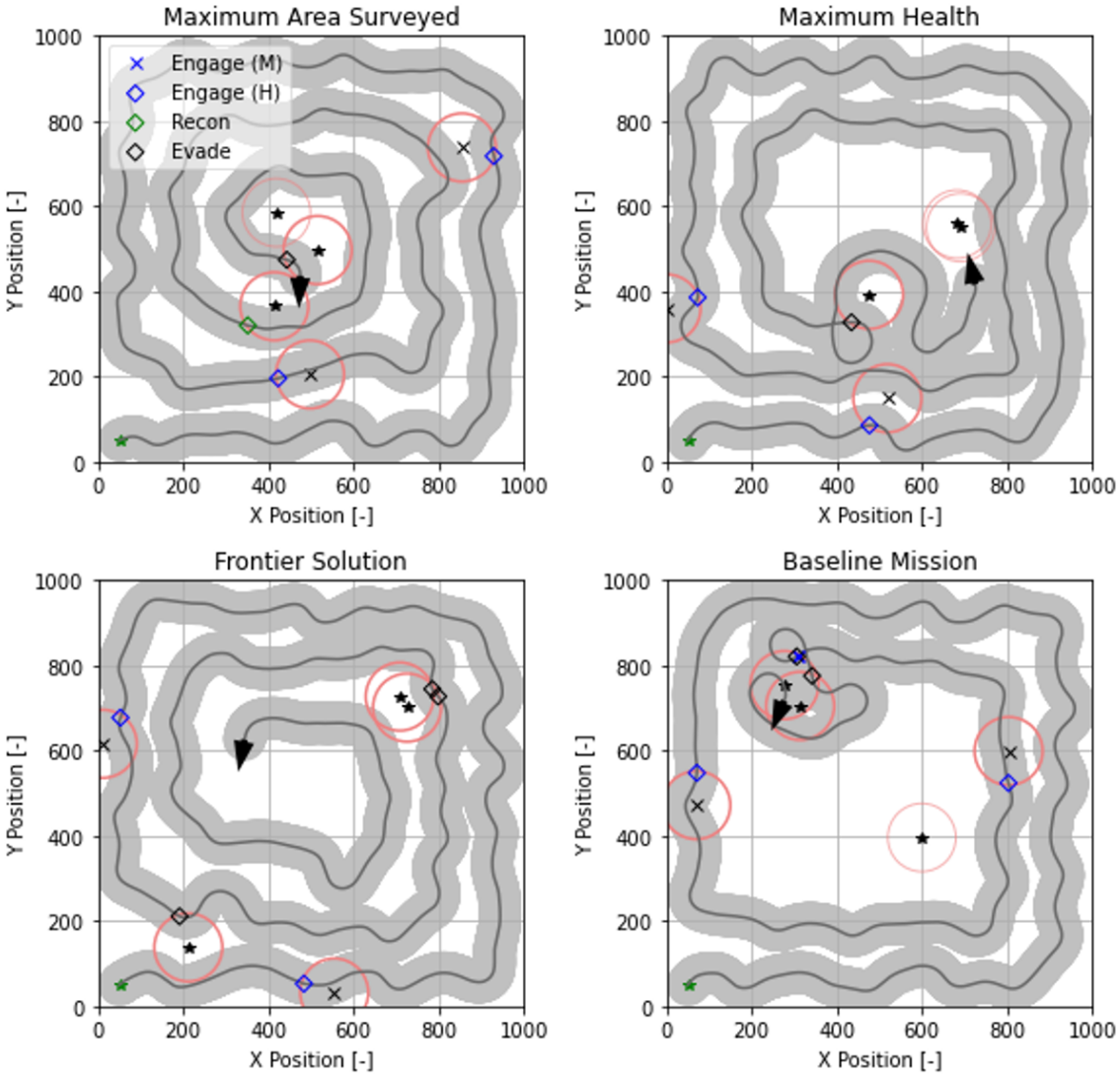

Figure 5.35: The solutions of four missions using the decision behavior algorithm.

One Notable tactic that the decision behavior uses for all four solutions is the choice to engage the first two defenders, which corresponds to the agent's available ammunition. Once the agent's ammunition is depleted, the agent almost always evades the defenders. The selection of engagement when ammunition is available results in a hit (H) if the drawn random number within the engagement logic is above the probability of kill. Neutralized

Table 5.3: The MoEs of the highlighted solutions for the baseline technology alternative.

Mission	**Health**	**Area Surveyed**
Mean MoE	41.91 ± 26.96	$68.96\% \pm 7.19\%$
Maximum Area Surveyed	64.95	74.59%
Maximum Health	95.42	71.34%
Frontier Solution	94.14	73.61%
Baseline Mission	0	62.43%

defenders are denoted by a black 'x' while active defenders are denoted by a black star. The baseline mission solution makes one mistake by choosing to engage when no ammunition is available, which is marked as a miss (M). However, the fatal mistake is due to the reconnaissance behavior's decision to cross over a known defender. Although the baseline mission solution highlights the chance of sub-optimal choices to be made by the agent's state behaviors, it provides an example of the traceability that the decision behavior approach has when evaluating the mission action design space.

The overall performance of the four solutions can be further investigated by the comparison of MoEs shown in Table 5.3. All four mission solutions achieve similar coverage of the contested area, with the baseline mission solution slightly lower. These results show further shows the sensitivity of agent's health as opposed to the area surveyed. The mission that resulted in the maximum area surveyed had a much lower health than the frontier solution while only achieving about 1% more area coverage.

The decision-state space can also be sampled to understand the learned behaviors of the decision behavior algorithm. The algorithm is sample $10,000$ times with uniform random state variables. The minimum amount of relevant information is defined by the observation maps which includes no area surveyed and a single defender visible. The behaviors chosen for each random sample are compared using a histogram, illustrated in Figure 5.36. The agent's health and available ammunition are found to be significantly more influential in the decision behavior's choice of state behavior than the agent's bank angle or heading relative to the defender. The frequency of choosing reconnaissance or evasion decreases as the agent looses health. Alternatively, the engagement behavior is more preferred when

the agent is at a lower health. This result is expected due to the higher consequence the agent may receive for evasion or reconnaissance which increases the agent's probability of reaching 0 health.

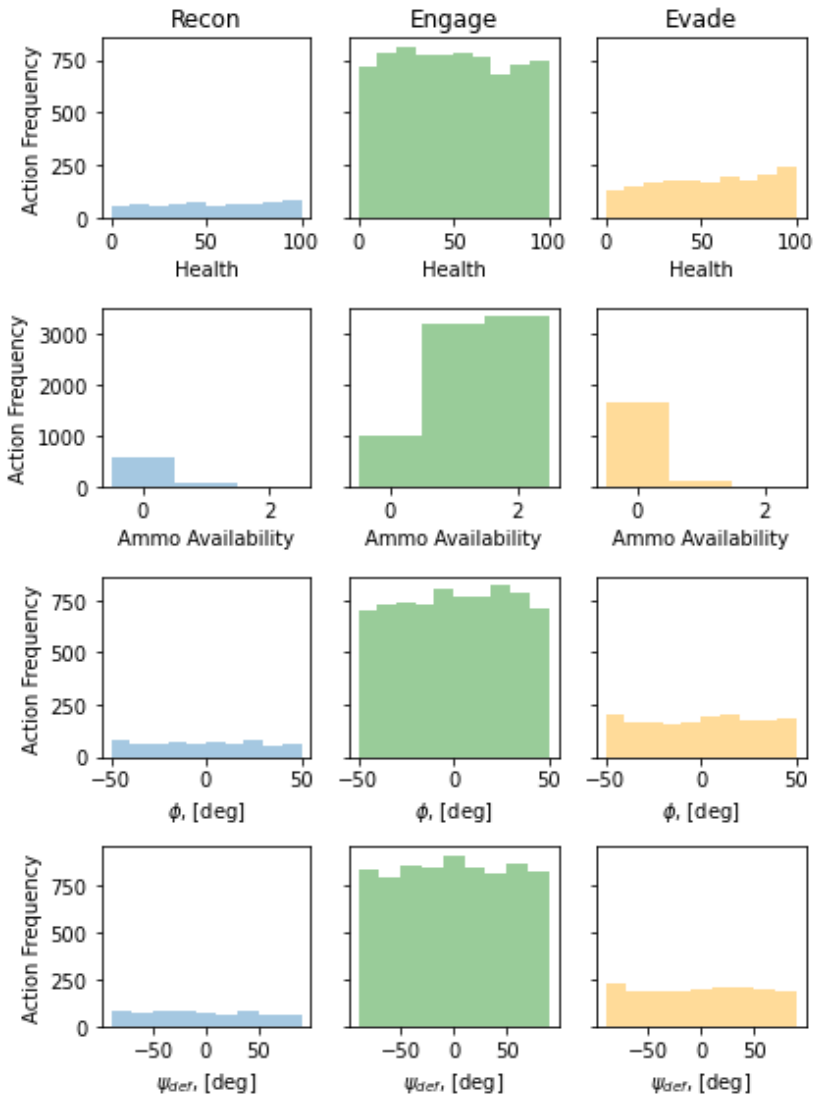

Figure 5.36: The decision behavior algorithm's frequency of state behaviors selected given an initial state.

The frequency each state behavior is chosen for the available ammunition is also expected; however, the algorithm seems to still select engagement when no ammunition is available. Additional training did not noticeably reduce the frequency of choosing engagement with no ammunition. Further reward tuning may be necessary to ensure the agent

learns not to engage in those states. The agent's bank angle and heading seem to be invariant to each state behavior. This result may suggest that the bank angle and state variables do not effect the agent's decisions; however, further investigation could be done to ensure the DNN is properly sized.

Overall, the use of a DNN with the decision behavior approach enabled the exploration of a complex partially observable decision-state space. The decision behavior algorithm successfully learns to make decisions that lead to higher MoEs for the contested reconnaissance mission. The results succeed in determining that the decision behavior approach can be implemented for complex decision-state and state-action spaces to explore the mission action design space of a minimally defined mission.

Research Conclusion 2.1*: The use of a neural network to approximate the immense decision-state space enables the use of the decision behavior algorithm for complex mission environments.*

5.4 Dimensionality of the Decision-State Space

The agent defined for the contested reconnaissance mission and implemented in experiment 2.1 has three state behaviors – reconnaissance, engagement, and evasion. However, as the number of state behaviors increases, the dimensionality of the problem increases. An increase in dimensionality is not desired for the already high dimensionality of technology evaluation studies. Each additional state behavior added to a technology evaluation study of n technologies adds 2^n additional RL algorithms that need to be trained. Any reduction in the number of state behaviors can provide significant decreases in dimensionality as long as the exclusion of the behavior does not limit the mission action space. The potential

Attributes		Alt 1	Alt 2	Alt 3	Alt 4
Configuration	Vehicle	Quad Copter	VTOL	Wing & Tail	Low RCS
	Planform	Low Speed	High Speed		
	Structure	Traditional Structure	Adv. Structures	Low Cost Structures	
Payload	Fuel Volume	<1000	1500	>2000	
	Sensors/Optics	Traditional Optics	Advanced Optics	COTS Optics	
	Munitions	Conventional Munitions	Adv. Munitions	Directed Energy	
Propulsion	Cruise Speed	High Supersonic	Supersonic	Subsonic	
	# Engines	6	4	2	1
	Engine Position	Forward Fuselage	Aft Fuselage	Under Wings	Over Wing
	Energy Source	Electric	Traditional Fuel		

Figure 5.37: Identification of technology alternatives from the morphological matrix.

dimensionality of the decision behavior approach is first quantified and then methods to reduce its size are investigated in the following sections.

5.4.1 Identification of Technology Alternatives

Determination of the dimensionality of technology evaluation starts with identifying the number of technologies of interest. This can be done by returning to the morphological matrix to identify technology alternatives. Each technology within the morphological matrix impacts that aircraft's performance or effectiveness, but does not significantly change the archetype of the vehicle. Figure 5.37 highlights the 9 identified technology alternatives in orange. Investigation of every configuration and technology within the matrix would result in 2^{21} alternatives if a 2-level full factorial Design of Experiments (DoE) is conducted and incompatibilities are ignored. The 9 identified technologies result in 2^9 alternatives. The reduction in alternatives from over 2 million to 512 by isolating specific technologies from the morphological matrix significantly reduces the dimensionality of the problem.

Implementation of the decision behavior approach requires a unique decision behavior algorithm for each alternative. In addition, each state behavior algorithm that is defined also requires the training of a NN for each alternative. The decision behavior approach implemented for the continuous contested reconnaissance mission would require 512 NNs for each state behavior algorithm and 512 NNs for the decision behavior algorithm. The total number of algorithms required for the investigation of 9 technologies requires the training

Table 5.4: Definition of technology bundles from the morphological matrix.

Technology Bundle	Description
Optic Technology	Advanced optics enable the agent to see further by increasing its field of view
Stealth Technology	Advanced materials reduce the agent's radar cross-section which reduces the defender's field of view
Structure Technology	Improved aircraft structure reduces the aircraft's weight while increasing its maximum turn rate
Endurance Technology	Electric propulsion and high aspect ratio configuration increases the aircraft's range and altitude, which impacts the defender's field of view
Munition Technology	Advanced munitions enable more precision in targeting, increasing the agent's probability of kill
High Speed Technology	Combination of a high speed configuration with advanced propulsion increase the agent's cruise speed and altitude

of $2,48$ NNs, assuming that all three state behaviors are defined by an algorithm. The training of $2,048$ NNs would require approximately $52,224$ hours or almost 6 years to train in sequence. Although training of each behavior can be done in parallel, the amount of training time required can be very significant if large computational resources are not available. This amount of NNs results in the dimensionality of the decision behavior approach to be prohibitive.

Any reduction in the number of technologies of interest would reduce the dimensionality of the problem. However, the choice of which technologies to keep or eliminate is not able to be determined a priori. Instead, the technologies can be grouped into technology bundles to reduce the dimensionality of the problem without eliminating any technologies. Six technology bundles, listed in Table 5.4, are defined from the set of 9 technologies highlighted in the morphological matrix.

The six technology bundles reduces the number of technology alternatives from 512 to 64 for a 2-level full factorial DoE. The reduction in alternatives results in 256 algorithms to be trained. The dimensionality of the problem now seems manageable. However, for every additional state behavior defined for the decision behavior approach, 2^n additional NNs are required to train. One method that can further reduce the dimensionality of the problem

Contested Reconnaissance UAV	Optic Technology	Stealth Technology	Structure Technology	Endurance Technology	Munition Technology	High Speed Technology
Technology	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**
Optic Technology		1	1	1	1	1
Stealth Technology			0	1	1	1
Structure Technology				1	1	1
Endurance Technology					1	0
Munition Technology						1
High Speed Technology						

Table 5.5: The technology compatibility matrix for the six identified technology bundles.

is by identifying technology incompatibilities with the Technology Compatibility Matrix (TCM). Two incompatibilities, illustrated in Table 5.5, are identified from the 6 technology bundles. The stealth and structure technology bundles are assumed to be incompatible due the flexible structure of advanced composite materials and radar absorbing materials. The other incompatibility is between the endurance and high speed technology bundles. The endurance technology bundle includes electric propulsion and high aspect ratio wings, which is directly counter to the low aspect ratio inherent in the high speed technology bundle.

The two incompatibilities reduce the technology alternative down from 64 to 36 which results in a total of 144 algorithm required for the decision behavior approach. The number of required algorithms is high, but acceptable for implementation of the decision behavior approach with three state behaviors; however, any increase in state behaviors can quickly cause the dimensionality to become prohibitive again. The number of state behaviors should not be limited before the decision behavior approach is implemented, but a method to reduce the number of state behavior algorithms required to train is needed to further address the dimensionality of the approach.

Table 5.6: State behavior definitions for a contested reconnaissance mission.

State Behavior	Description
Ingress	The initial state of the agent which defines its initial conditions
Recon	The agent is actively surveying the area while avoiding known defender locations
Engage	The agent is engaging the defender with predefined offensive capabilities
Evade	The agent maneuvers away from the defender in an optimal manner
Loiter	The agent loiters around a defender to improve its surveillance coverage
Retreat	The agent safely retreats from the contested environment
Egress	The final state of the agent which defines its route to exit the contested region

5.4.2 Identification of Suitable State Behaviors

The dimensionality of the decision behavior approach with N_{SBA} state behaviors can require up to $2^n(N_{SBA}+1)$. The bundling of technologies and identification of technology incompatibilities reduced the number of technology alternatives, but those options may not sufficiently reduce the dimensionality of the approach. In those cases, the suitability of each state behavior for each technology alternative can be addressed to further reduce dimensionality. Seven state behaviors are defined for the contested reconnaissance mission. Table 5.6 defines each state behavior including the original three behaviors used for the discrete contested reconnaissance mission. The new behaviors include ingress, loiter, retreat, and egress.

The addition of the four state behaviors doubles the dimensionality of the decision behavior approach by increasing the total number of NNs required to train from 144 NN to 288. The sequential training of each NN would require approximately 96 days to complete assuming each NN requires 8 hours to train. Further reduction in the dimensionality of the decision behavior approach is needed to ensure the computational requirements are manageable.

Contested Reconnaissance UAV	Optic Technology	Stealth Technology	Structure Technology	Endurance Technology	Munition Technology	High Supersonic Technology
Baseline: Reconnaissance	1	1	1	1	1	1
Ingress	1	1	1	1	1	1
Evade	1	1	1	1	1	1
Engage	1	1	1	0	1	0
Loiter	1	1	1	1	1	0
Retreat	1	1	1	1	1	1
Egress	1	1	1	1	1	1

Table 5.7: The behavior suitability matrix for seven state behaviors and six technology bundles.

In order to further reduce dimensionality, the suitability of each state behavior with each technology bundle should be evaluated. This process is similar to identifying technology incompatibilities, but is instead focused on state behavior and technology pairs. Since each state behavior is defined for a general task, most technologies are considered to be compatible with the behavior. In reality, some state behaviors may not be suitable to implement with a given technology. The Behavior Suitability Matrix (BSM), illustrated in Table 5.7, defines the suitability of each state behavior paired with each technology. Values of "1" signify that the behavior and technology are suitable while values of "0" signify an unsuitable behavior-technology pair.

The BSM identifies three unsuitable behavior-technology pairs. The engagement behavior is determined not to be a suitable behavior for an endurance technology bundle due to the weight and volume of munitions. In addition, the engagement behavior and the high speed technology bundle are not suitable due to the high volume efficiency desired for high speed aircraft. The final unsuitable behavior-technology pair is the loiter behavior and the

high speed technology bundle due to the large turn radius at high speeds. The identification of three unsuitable behavior-technology pairs reduces the dimensionality of the decision behavior approach from 288 to 252 NNs.

Overall, the introduction of the behavior suitability matrix proved successful in reducing the dimensionality of the decision behavior approach. However, the number of technology alternatives is found to have a much larger influence on the dimensionality of the decision behavior approach. The identification of 2 technology incompatibilities reduces the number of NNs from 512 to 288. In comparison, 3 unsuitable behavior-technology pairs with no incompatibilities reduces the number of NNs from 512 to 440. The BSM succeeds in reducing the number of NNs required to train with or without technology incompatibilities. These findings confirm the hypothesis and inform research conclusion 2.2.

Research Conclusion 2.2*: The use of a behavior suitability matrix succeeds in reducing the dimensionality of the decision behavior approach. However, the problem's dimensionality is found to be more sensitive to the number of technology alternatives than the number of state behaviors.*

5.5 Cooperative Agent Implementation

This section will extend knowledge learned from past experiments to multi-agent systems to aid in answering the hypothesis developed in Section 3.4.2. The extension of the decision behavior algorithm defined for the continuous contested reconnaissance mission is considered as the baseline alternative for each cooperative agent. Cooperative information is shared by having each agent update a single truth map of the surveyed area and known defender locations. Additional information is shared during the agent's reconnaissance

behavior to ensure the agents do not overlap or explore the same area. The mission environment assumes that each agent is flying at a unique altitude which allows agents to fly over each other without penalty.

The extensibility of the decision behavior approach was investigated to address the applicability of the approach to multi-agent systems. Specifically, the dimensionality of the decision behavior approach was investigated to address the exponential increase in alternatives for cooperative agents. The exploration of n technologies results in 2^n alternatives for a single agent mission. However, the addition of cooperative agents not only increases the complexity of the mission action design space, but also the dimensionality due to the introduction of mixing technology alternatives. The size of the mission action space of cooperative agents with different technology alternatives is substantial and should not be limited a priori. The dimensionality of mixing technology alternatives with a_c cooperative agents could increase the number of alternatives up to $2^{a_c n}$ for a 2-level full factorial design.

In order to address cooperative agent missions, the agent's are assumed to be homogeneous. This will enable cooperative missions to be addressed for the decision behavior approach while keeping the dimensionality at or below 2^n alternatives. However, if the number of cooperative agents is a design variable, than the number of decision and state behavior algorithms increases proportionally. This experiment will investigate the implementation of a decentralized agent framework to enable the identification of independent and dependent state behaviors, reducing the dimensionality of a cooperative agent decision behavior approach.

5.5.1 Identification of Cooperative Behaviors

The identification of cooperative behaviors requires that the decision behavior approach and RL algorithm be defined for cooperative agents. The decentralized agent framework is the appropriate mechanism to define agent decision and control for the decision behavior approach. This will enable agents to make localized decisions based on local observations

without an exponential growth in communication costs [135].

Decentralized agent decisions enables the identification of dependent and independent behaviors. Dependent state behaviors are those which require direct information from other cooperative agents such as their locations. The reconnaissance behavior is one example of a dependent state behavior. During reconnaissance, cooperative agent information is beneficial to ensure that the contested area is surveyed in an optimal manner with minimal agent overlap. Similar information sharing has been successful for the state-action design space of a wildfire surveillance mission [102]. Alternatively, independent state behaviors are those which do not depend on the interaction with other agents, but may still depend on global information, such as the global mission map. The evasion behavior is determined to be a state behavior that does not depend on cooperative information. This behavior is not concerned with the location of other agents due to its priority of escaping a defender. In some problems though, cooperative evasion may be desired. However, the continuous contested reconnaissance mission enables the use of an independent evasion behavior without reducing the mission action design space.

Along with the evasion behavior, the engagement behavior is determined to be independent. This behavior is not based on an algorithm and therefore does not require any global information. Overall, two independent state behaviors and one dependent state behavior are identified for a decentralized decision behavior approach. The evasion behavior algorithm from the single agent mission can be carried over without retraining the agent. Although the evasion behavior is independent of other agent's actions, it still uses the global defender map when choosing decisions in the state-action space. The shared global defender map does not impact the defender's evasion behavior due to its dependence on only defender information.

The reconnaissance behavior will require a new algorithm to be implemented for the cooperative contested reconnaissance mission. The decision behavior can be defined for local or global decisions. The implementation of the decision behavior approach for cooperative

agents and cooperative decisions are both complex problems. The implementation of this cooperative contested reconnaissance mission will assume only localized decisions need to be made in the decision-state space, while global decisions can be made by the dependent state behaviors in the state-action space. The decentralized agent decision framework for the defined mission requires only the reconnaissance behavior be retrained for cooperative decisions.

Cooperative Reconnaissance Behavior

The reconnaissance behavior is implemented in a cooperative state-action space using RL. The use of RL provides generality and robustness for learning cooperative behavior [118]. Typical multi-agent RL algorithms use off-policy methods such as Q-learning. Many off-policy methods have been successfully implemented for multi-agent problems [117, 118, 102, 119]. Their prolific use can be attributed to the belief that off-policy methods have a higher sampling efficiency [180]. However, some on-policy algorithms, such as Multi-Agent PPO, have proven that false. Implementation of the reconnaissance behavior will use the PPO algorithm defined in Section 5.3.2 with $N = 2$ actors.

Each reconnaissance agent utilizes the same behavior logic as the single agent reconnaissance state behavior to survey the contested area. The homogeneous agents also use the same baseline alternative specifications defined in Table 5.2. In order to capture cooperative information, the DNN architecture is redefined to incorporate the addition of more continuous state variables. The agents now have five continuous state variables and two observations maps that define its partially observable state. Using information from the other agent and the global map, the agent can choose to increase of decrease its bank angle.

The five continuous state variables are defined as the agent's own bank angle, ϕ_a, the agent's own heading, ψ_a, the agent's bearing relative to the cooperative agent's location, θ, the distance between the two agents, ρ, and the bank angle of the cooperative agent, ϕ_b. Both the survey map and defender map are also input into the agent's DNN to provide

global surveillance information along with known defender locations. The DNN architecture, illustrated in Figure 5.38, adds three additional fully connected dense layers with 100 neurons each to network defined for the single reconnaissance agent in Section 5.3.2. This DNN architecture is appropriate for this cooperative behavior due to its success in similar decentralized multi-agent systems [102].

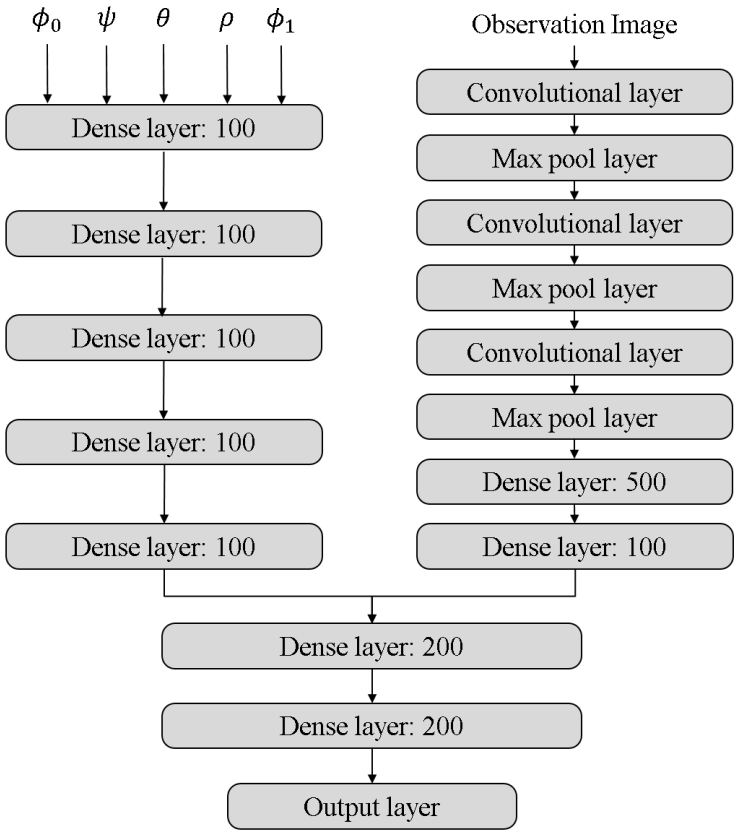

Figure 5.38: Neural network architecture for the cooperative reconnaissance state behavior modified from [102].

The two agents start in opposite corners of the map with (x, y) coordinates of $(50, 50)$ and $(950, 50)$. The DNN for the cooperative reconnaissance behavior is trained over $3,000$ with a maximum of $2,500$ time steps per episode. The limit in time steps provides sufficient time for the agent to reach the training goal of 70% map coverage. The minibatch size of 200 and $K = 5$ epochs is used for updating the algorithm's policy. The learning rate and discount factor are also set at values of $\alpha = 0.001$ and $\gamma = 0.99$, respectively. Convergence

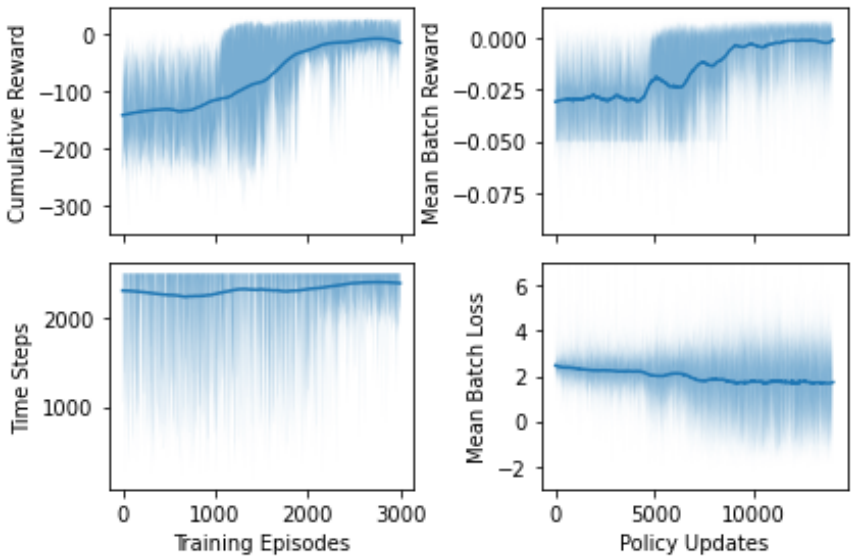

Figure 5.39: Convergence of the cooperative reconnaissance state behavior.

of the algorithm, illustrated in Figure 5.39, requires 17 hours of learning.

The representative reconnaissance mission is illustrated to ensure the agents conduct reconnaissance in an optimal manner while avoiding known defender locations. Each algorithm started from a unique random seed is tested on the same set of 200 missions. The algorithm that produces the highest mean reward from the test missions is used the reconnaissance behavior for the decision behavior algorithm. The baseline mission is illustrated in Figure 5.40 for the cooperative reconnaissance behavior algorithm with the highest mean reward.

5.5.2 Cooperative Decision Behavior

The use of decentralized agents enables the decision behavior to be independent from the number of agents. The decision behavior algorithm is provided the same inputs as the single agent reconnaissance mission. However, since the agent's are making localized decisions, the inputs are relative to the agent making the decision. In other words, the decision making agent is given its own bank angle, heading relative to the defender, available ammunition, and health for the continuous state variable inputs. Although the survey and defender maps

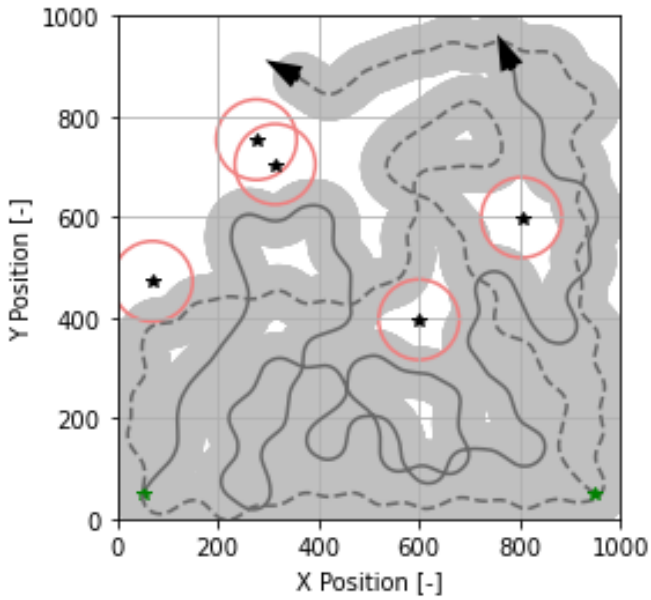

Figure 5.40: The cooperative reconnaissance behavior surveying the baseline mission with known defender locations.

are based on global observations from each agent, the inputs remain independent because they do not provide state information.

The same decision behavior algorithm from the single agent contested reconnaissance mission is used for the cooperative mission to define the agent's localized decision making algorithm. The cooperative mission is once again run using the same set of 200 missions that were generated stochastically for the single agent decision behavior algorithm. The MoEs, illustrated in Figure 5.41, are defined as the average health of the two agents and the amount of area surveyed for each mission solution. The mean MoEs for the cooperative agent decision behavior implementation provides a significant increase in the area survey while maintaining a similar health between the two agents as that of the single agent. These results are expected because the foundational behavior of both the decision behavior and state behaviors does not change as more agents are added, but the increase in agents enables the cooperation in achieving the mission goal of maximizing the surveyed area. The cooperative baseline mission as well as the frontier solution are highlighted. The frontier solution is defined as a point that lies on the Pareto frontier and is chosen by taking the

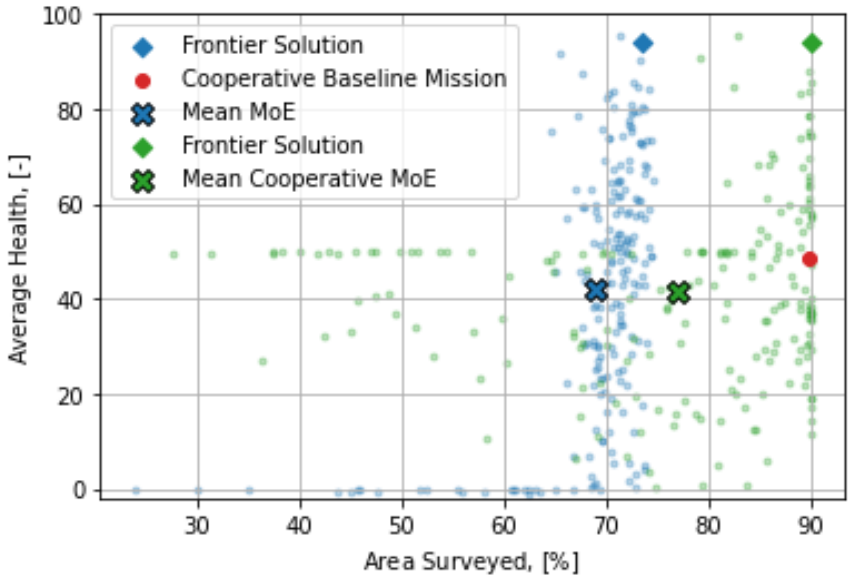

Figure 5.41: The MoEs for both the cooperative and the single agent contested reconnaissance mission with the same sample set of 200 missions.

maximum of the combined MoEs, equally weighted.

The corresponding MoEs for each mission along with the mean MoEs for the single and multi-agent mission sets are shown in Table 5.8. The cooperative agent missions resulted in an 8% increase in the area surveyed on average than the single agent contested reconnaissance missions. The frontier solutions of the cooperative and single agent missions are not an equal comparison due to the different defender locations for each mission. However, the shift in the overall Pareto frontier can be seen by the significant increase of 16.46% in area surveyed. Overall, the cooperative agent successfully completed 136 out of the set of 200 missions, resulting in a success rate of 68%. This success rate is significantly lower than the single agent which is believed to be due to the higher complexity of the cooperative reconnaissance state behavior. Further tuning of the algorithms rewards and hyperparameters will likely improve the overall performance of the cooperative agent's.

The baseline mission and frontier solution are illustrated in Figure 5.42. Since the same decision behavior algorithm is used from the single agent mission, similar decision behavior is found. The agents prefer to engage defenders early on in the mission and evade

Table 5.8: The MoEs of the cooperative and single agent decision behavior approach.

Mission (Agents)	**Average Health**	**Area Surveyed**
Mean MoE (1)	41.91 ± 26.96	$68.96\% \pm 7.19\%$
Baseline Mission (1)	0	62.43%
Frontier Solution (1)	94.14	73.61%
Mean MoE (2)	41.48 ± 19.26	$76.96\% \pm 15.14\%$
Baseline Mission (2)	48.675	89.99%
Frontier Solution (2)	94.12	90.07%

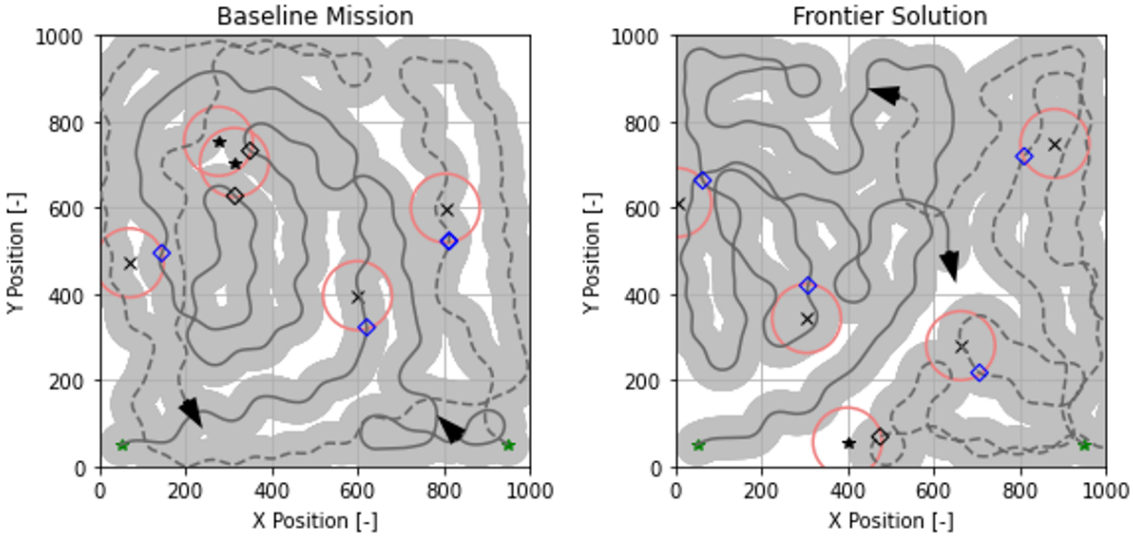

Figure 5.42: The baseline and frontier solutions of the cooperative contested reconnaissance mission.

defenders once their available ammunition is zero.

Overall, the cooperative mission successfully implemented the decision behavior approached for a decentralized ABM. The decentralized decision-making enabled the identification of independent behaviors, which reduced the total number of behavior algorithms that had to be retrained for the cooperative agent simulation. If every behavior algorithm were dependent on the number of agents in the model, than $2^n(N_{SBA}+1)$ algorithms would be required for every unique number of cooperative agents investigated. Implementation of the decision behavior approach in a decentralized ABM results in a reduction in dimensionality by only requiring an additional $2^n(N_{dep})$ algorithms, where N_{dep} is the number of agent dependent behavior algorithms. The investigation of $n = 6$ technologies with

$N_{SBA} = 3$ would result in 512 NNs required to train while the identification of 2 independent state behaviors and an independent decision behavior results in 320 required NNs for the investigation of one and two agent mission.

The use of decentralized agents for the decision behavior approach expands the mission action design space by including cooperative agent missions. The investigation of decentralized cooperative agents found that the decision behavior approach can be applied to multi-agent systems without the linear increase in dimensionality. This finding expands the possible applications the decision behavior approach could have for technology evaluation.

Research Conclusion 2.3*: The use of decentralized agents enables the identification of independent behavior algorithms which reduces the dimensionality of the decision behavior approach when implemented across multi-agent systems.*

5.6 The Exploration of Tactical Alternatives

The RL algorithm selected for the continuous reconnaissance mission utilizes a stochastic policy. This provides action value distributions instead of deterministic values. The ϵ-greedy policy, discussed in Section 5.2.2, typically chooses the action that returns the maximum action value unless the random number drawn from a uniform distribution passes the threshold, $1 - \epsilon$, in which case would result in the selection of a non-optimal action value. The problem with this approach is the possibility of randomly choosing an action with a very low action value, which could result in a significant penalty for the agent. The use of a stochastic policy provides probability distributions for each possible action where every action has a varying degree of probability that it will be selected. This enables the NN to provide appropriate action probabilities for perceptually identical states instead of almost

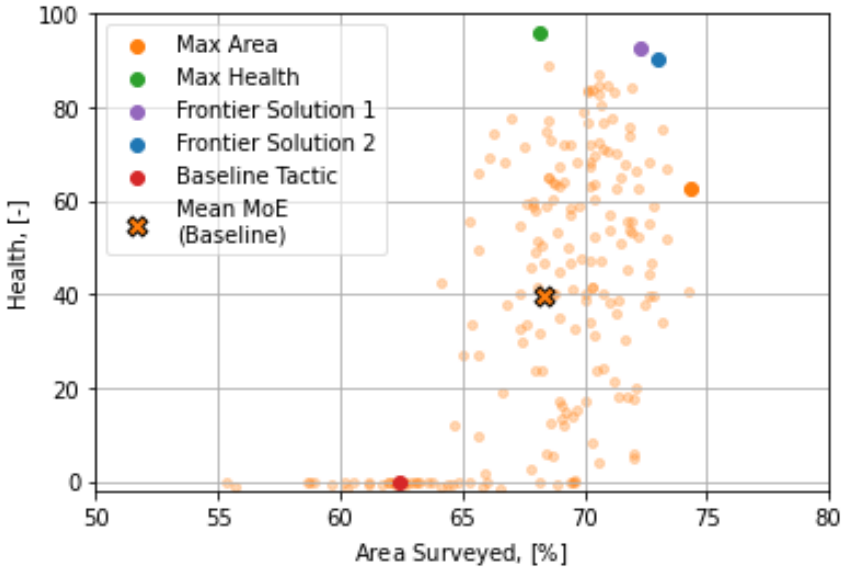

Figure 5.43: The MoEs of different tactical alternatives for the baseline mission.

always choosing the maximum action value in a deterministic policy.

Application of a stochastic policy with the decision behavior approach provides stochasticity in the agent's decision behavior algorithm. The use of a stochastic policy in particular enables the agent to explore tactics that result in similar MoEs, but use different tactics to achieve them. The continuous contested reconnaissance mission was investigated to evaluate the use of a stochastic policy with the decision behavior algorithm. The decision and state behavior algorithms developed in Section 5.3.2 are implemented for the baseline mission.

5.6.1 The Stochastic Decision Policy

The baseline mission is sampled 200 times with the stochastic decision behavior policy. Figure 5.43 illustrates the variation in the MoEs for the baseline mission samples. Four solutions were chosen to investigate the tactical alternatives chosen for the baseline mission. The four solutions are considered to be in the set of optimal solutions on the Pareto frontier. The response of the baseline tactical alternative, illustrated in Figure 5.35, is highlighted again in red for comparison.

Table 5.9: The MoEs of the variation of tactical alternatives.

Mission	**Average Health**	**Area Surveyed**
Mean MoE	39.68 ± 28.84	$68.35\% \pm 3.84\%$
Maximum Health	95.93	68.19%
Maximum Area Surveyed	62.62	74.37%
Frontier Solution 1	92.82	72.26%
Frontier Solution 2	90.53	73.01%
Baseline Tactical Alt.	0	62.43%

The stochastic policy resulted in 160 successful tactical alternatives out of 200 samples. The mean response is significantly higher than the baseline alternative for the baseline mission. Their MoEs along with those of the highlighted set of optimal tactical alternatives are compared in Table 5.9. The stochastic policy finds that the exploration of tactical alternatives enables the improvement in mission effectiveness by identifying alternative *ways* to use the baseline vehicle alternative. Compared to the baseline tactical alternative, the stochastic policy enabled the agent to not only successfully complete the mission, but also achieve very high MoEs.

Each tactical alternative used by the decision behavior algorithm can be defined by the set of decisions made by the decision behavior algorithm. These decisions are illustrated in Figure 5.44 for the baseline mission. All four tactical alternatives chose to engage the first two defenders that were encountered. It also is worth noting that the agent is flying directly towards the first two defenders when they are discovered, which is one of the more consequential states the agent could find itself in. Once the agent's reach zero ammunition, they favor the evasion state behavior while occasionally choosing reconnaissance when their heading is more tangential to the defender's field of view. Some sub-optimal decision are chosen without too high of a consequence such as choosing to engage a defender when no ammunition is available, resulting in a miss (M).

The maximum area surveyed solution resulted in the lowest health value out of the four highlighted solutions. This result is expected when maximizing the area surveyed due to the increased chance of discovering more defenders. This solution was the only one to discover

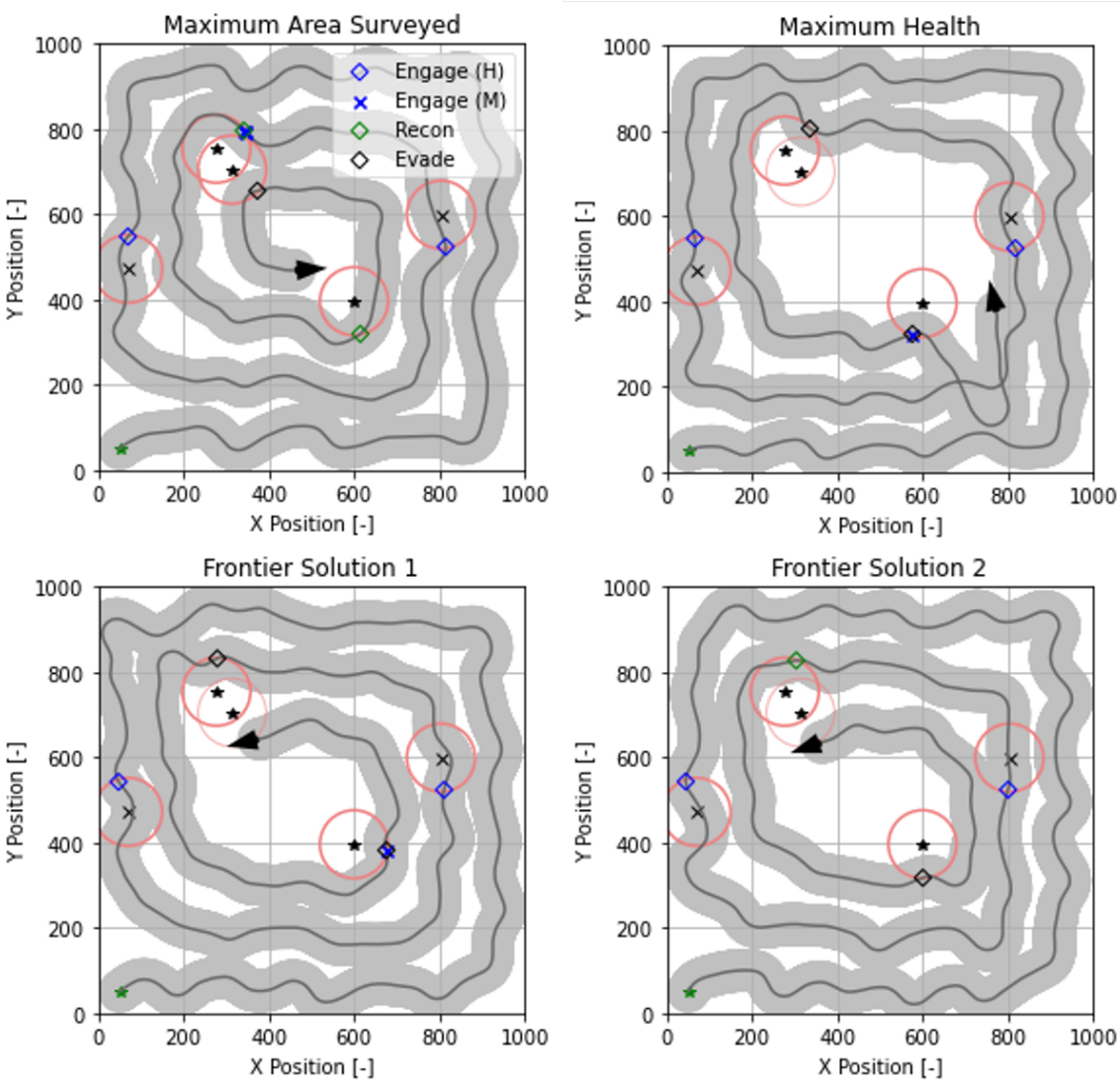

Figure 5.44: Tactical alternatives of the baseline mission.

all five defenders in the contested area, which is partly why the solution had a final health of 62.62, almost 30 points lower than the other three solutions. The algorithm's choice of reconnaissance after discovering two of the five defenders prioritized the exploration of the map over evading the defenders.

The baseline mission provides a fixed environment to explore tactical alternatives with a stochastic policy. The results found that variation in tactical alternatives can produce a wide range of responses for the baseline technology alternative. The stochastic policy enables the variation of the *ways* a technology alternative can be used to complete a mission without

requiring SMEs to craft mission tactics a priori. These findings confirm hypothesis 3 and inform the following corresponding research conclusion.

Research Conclusion 3*: The use of a decision behavior algorithm with a stochastic policy enables the exploration of tactical alternatives.*

CHAPTER 6
THE TECHNOLOGY-TACTIC SYNTHESIS

Confirmation of the proposed Tech-DEBATE methodology implemented for the baseline technology alternative gives confidence to its applicability for a full technology evaluation study. Exploration of the *ways* each technology alternative is used augments the quantification of mission effectiveness thereby providing more information early on in the design process. The discovery of novel tactical alternatives could enable more informed technology investment decisions during the development and acquisition of evolutionary or revolutionary technologies.

The six technology bundles identified in Section 5.4.1 were implemented in the contested reconnaissance mission. A two-level full factorial design was conducted with two technology incompatibilities resulting in 36 technology alternatives including the baseline alternative. Their responses were investigated to quantify the impact the decision behavior approach has on their overall mission effectiveness. The impact of each technology was captured through "k"–factors that operate on disciplinary metrics. Seven metrics were identified that can fully capture the impacts of each technology. The "k"–factors operate on the agent's baseline cruise speed, field of view, turn rate, probability of kill, and range as well as the defender's baseline field of view. The agent's ammunition "k"-factor, k_A, operates on the maximum ammunition value of 4 as opposed to the baseline value of 2. The "k"-factor vectors for each technology bundle are defined in Table 6.1

In practice, these technology "k"-factors are determined by identifying the theoretical limit of each technology through research or literature review [25]. However, this proof-of-concept study chose arbitrarily conservative values to provide hypothetical limits for each technology that enable variation of the Measures of Effectiveness (MoEs). The non-dimensionalized impact was included to quantify the total improvement or consequence all

Table 6.1: The technology impact matrix for the contested reconnaissance UAV.

Contested Reconnaissance UAV		Technology Bundles						Non-Dimensionalized Impact	
		Optic Technology	Stealth Technology	Structure Technology	Endurance Technology	Munition Technology	High Speed Technology		
k-factor Name		T1	T2	T3	T4	T5	T6	Minimum	Maximum
1	Cruise Speed, kv	0%	0%	0%	-10%	0%	20%	-10%	20%
2	Field of View, kfov	20%	0%	0%	10%	0%	0%	0%	30%
3	Stealth, kfovdef	0%	-20%	0%	-10%	0%	-10%	-40%	0%
4	Maneuverability, ke	0%	0%	10%	-10%	0%	-10%	-20%	10%
5	Probability of Kill, kPk	5%	10%	0%	0%	15%	0%	0%	30%
6	Munitions, kammo	-25%	-50%	0%	-50%	0%	-25%	-100%	0%
7	Range, krange	0%	-5%	10%	15%	-10%	-15%	-30%	25%

technologies could have on the agent's performance metrics, assuming no incompatibilities.

The 36 technology alternatives are defined in Table A.1 in Appendix A. Every technology alternative requires the training of a reconnaissance, evasion, and decision behavior algorithm. Those technology alternatives not compatible with the engagement behavior are also noted in Table A.2 by an available ammunition value of 0. The mean batch reward for each technology alternative, illustrated in Figures B.1, B.2, B.3, in Appendix B, is used to determine convergence of the behavior algorithms. All 108 algorithms are trained from scratch 5 separate times, each with a unique random seed. The random seed that results in the highest reward for each state behavior was used during the training of the decision behavior algorithm for each technology alternative. The decision behavior algorithms for each technology alternative that result in the highest reward were used in the comparison of the MoEs.

The MoEs of each technology alternative were compared to quantify the impact both the *means* and *ways* have on an alternative's mission effectiveness. Tactical alternatives of the baseline technology are first investigated using the results from Section 5.6.1 to identify the *ways* frontier. How each technology impacts the alternative's mission effectiveness was then investigated and compared to variation of the tactical alternatives of the baseline

technology alternative. Once the appropriate *means* frontier and the *ways* frontier were identified, the results from decision behavior approach were utilized to quantify the impact the *means* and *ways* have on the mission effectiveness. This investigation quantified the impact the Tech-DEBATE methodology has on technology investment by providing critical decision-making data derived from each alternative's mission effectiveness.

6.1 Variation of the Ways

The use of a stochastic policy for the decision behavior algorithm enabled variation of tactical alternatives with the baseline mission. The *ways* the baseline technology is used in a stochastic mission environment is desired to enable further exploration of the mission action design space. However, Section 5.6.1 found that sampling the same environment mission multiple times with a stochastic policy resulted in the improved quantification of MoEs for the baseline mission. Those results are compared with the sample set of 200 stochastic missions, each sampled once, using the baseline technology alternative. Figure 6.1 illustrates the comparison of response from the baseline mission sampled 200 times with a stochastic policy and 200 stochastic missions sampled once with a stochastic policy. One and two standard deviations of the two data sets are illustrated as dashed and solid lined ellipses, respectively. The mean MoEs of each data set corresponds to the center of the ellipses. A standard deviation of 1σ represents the region that 68% of the response is expected to fall within while 2σ standard deviations represents the region where 95% will fall within [181].

The comparison of results using the stochastic policy for the baseline mission and a stochastic set of missions find that the baseline mission resulted in a much smaller standard deviation of area surveyed. The sampling of the baseline mission resulted in mean MoEs of 39.68 ± 28.84 health and $68.35\% \pm 3.84\%$ area surveyed while the MoEs of the stochastic mission set resulted in 41.91 ± 26.96 health and $68.96\% \pm 7.19\%$ area surveyed. The mean response for both sets of results are very similar, which is expected due

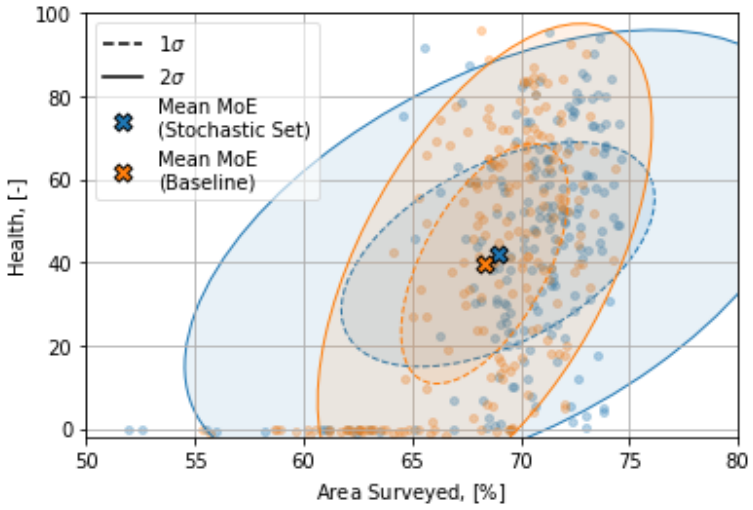

Figure 6.1: Variation of tactical alternatives for the baseline and a stochastic set of missions.

to the same decision behavior algorithm driving the baseline agent. The higher standard deviation in area surveyed for the set of stochastic missions is initially believed to be due to an increase in failed missions. However, sampling the baseline mission resulted in 40 failed missions while the stochastic mission set only resulted in 24 failed missions. The cause of the stochastic mission set's higher standard deviation is due to the occurrence of advantageous defender placements.

One solution not shown in Figure 6.1, but included in the calculation of the mean MoEs, is a failed mission from the set of stochastic missions that is only able to achieve 23% area coverage of the contested area before incurring over 100 damage. This mission, illustrated in Figure 6.2, makes valid decisions using the decision behavior algorithm, but its reconnaissance state behavior does not. The results show that the reconnaissance state behavior prioritized surveillance of the map over avoiding a known defender. The close placement of defender may also have had a part in the decisions that the reconnaissance behavior made. These findings raise the possibility of outliers that can be removed from the data.

The identification of outliers is conducted using boxplots to identify the points that fall outside $1.5\times$inner-quartile range [182]. This method is acceptable since the range of health does not have any visible outliers for either data set, which enables the isolation of

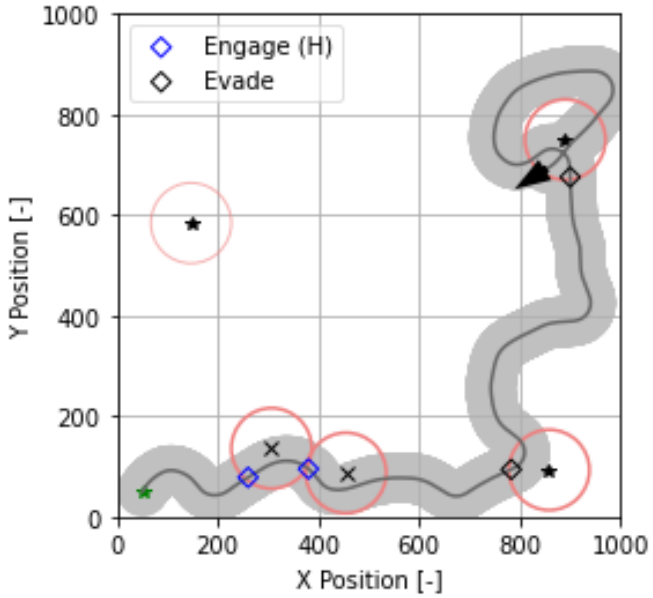

Figure 6.2: The lowest performing solution from the stochastic mission set.

the survey area to identify outliers. If this were not the case, the use of multi-variate outlier determination, such as an adaptive χ^2 method is preferred [183]. Figure 6.3 illustrates a boxplot of each MoE to visualize the presence of outliers in each data set. The area surveyed data is much more tightly clustered, resulting in outliers for missions which incurred 100 damage early on in the scenario.

The identified outliers are removed from the data set and new standard deviation ellipses are drawn. Figure 6.4 illustrates the shift in ellipse shape enabling the identification of a

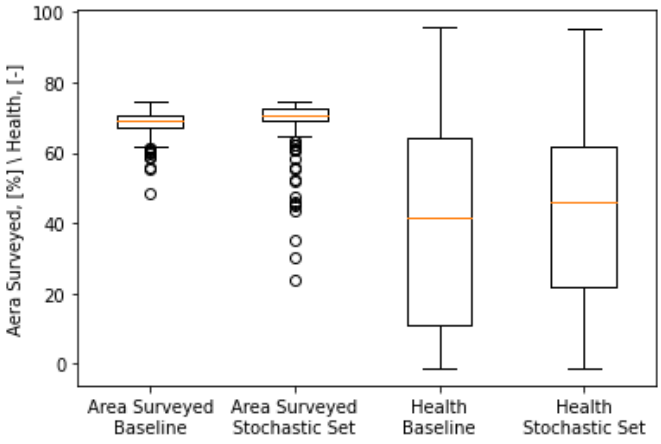

Figure 6.3: Boxplot visualization for each MoE from the two data sets.

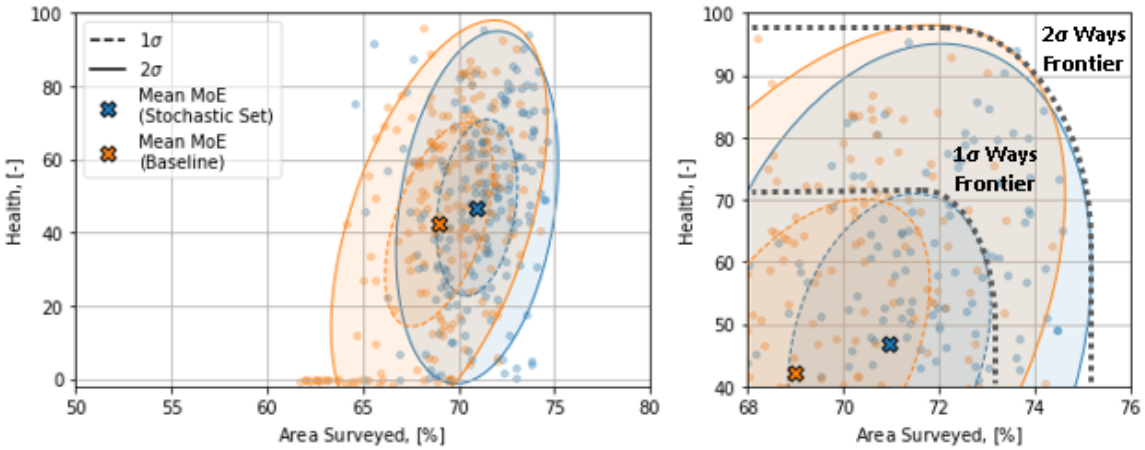

Figure 6.4: Visualization of Pareto frontiers generated by a stochastic policy.

Pareto Frontier. Both the baseline mission data set and the stochastic mission set achieved similar 2σ frontiers which are based on the ellipses that correspond to a standard deviation of 2σ. However, the mean response along with the ellipse corresponding to 1σ standard deviation for both data sets tells a different story. The stochastic mission set is able to achieve higher mean health and area surveyed than the baseline mission set. This result is expected due to the benefit of stochastic variation of the defender locations. Based on these findings, the 1σ frontier is determined to be a more appropriate measure of the variation of the *ways*.

The isolation of the results from the stochastic mission set can be expanded to highlight the additional information that the variation of the ways provides. The standard deviation ellipses not only highlight the optimal expected MoEs, but can also be used to identify the worst performance one would expect for the technology alternative. Additional Pareto frontiers can be drawn to highlight the worst performance one would expect for a technology alternative. These frontiers, illustrated in Figure 6.5, can be defined as the -1σ frontier and -2σ frontier using their corresponding standard deviation ellipse.

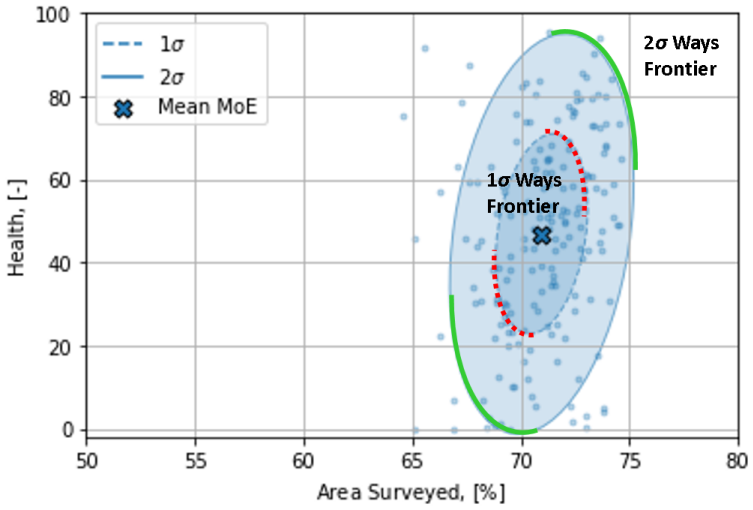

Figure 6.5: Variation of the ways enables the identification of the best and worst expected performance frontiers.

6.2 Variation of the Means

The *means* or technology alternatives with the baseline tactical alternative were investigated to quantify their MoEs for the contested reconnaissance mission. The baseline tactical alternative is defined as the reconnaissance state behavior. This is equivalent to a decision behavior algorithm that can only choose one state behavior – reconnaissance. Even though most technology alternatives have ammunition for engagement, the baseline tactic assumes that the agent has no engagement protocol. The MoEs for each technology alternative are quantified using the stochastic mission data set. The agent is given the same initial conditions defined in Section 5.3.2; however no defender locations are known initially. The variation of the *means* is illustrated in Figure 6.6. The baseline technology alternative using the baseline tactical alternative is listed as the baseline point in the figure.

The investigation of each technology alternative with the baseline tactical alternative results in an increase in mean health for almost all alternatives compared to the baseline alternative. Several technology alternatives also produce a higher mean area surveyed than the baseline. A Pareto frontier can be drawn based on the 1σ ellipses of the technology alternatives. Figure 6.7 illustrates the 1σ *means* frontier. Variation of the *means* enables

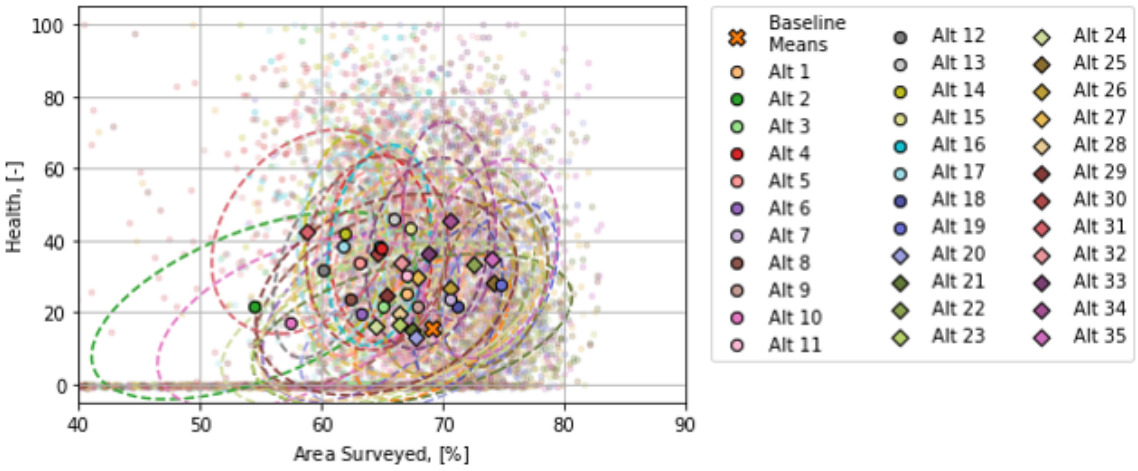

Figure 6.6: Variation of MoEs for each technology alternative with the baseline tactical alternative.

the identification of technologies that are in the set of optimal technologies defined by the Pareto frontier. Both technology alternative 34 and 21 lie on the 1σ ways frontier. Technology 34 uses the optics, stealth, and endurance technology bundles to achieve its high health while technology 21 uses the optics, munitions, and high speed technology bundles.

The identification of optimal solutions that lie on the 1σ ways frontier seem like the appropriate solutions to consider for technology investment. In reality, more information should be investigated to ensure the highlighted solutions can actually achieve the promised optimal MoE responses. A solution that has the highest worst performing responses within the 1σ ellipse may actually provide a better solution even if it does not lie on the 1σ ways frontier. To identify those solutions, the -1σ ways frontier is identified using the best performing solutions that lie opposite to their optimal responses. Figure 6.8 illustrates the 1σ and -1σ ways frontiers. Technology alternative 21 is found to provide a much larger variation in response than other technology alternatives, which causes it to be removed from the set of optimal solutions. Technologies 13 and 19 are discovered to provide the best MoEs for the worst performing optimal solutions even though their best performing solutions did not lie on the 1σ ways frontier. Alternatives 13 and 19 used the stealth and high speed tech-

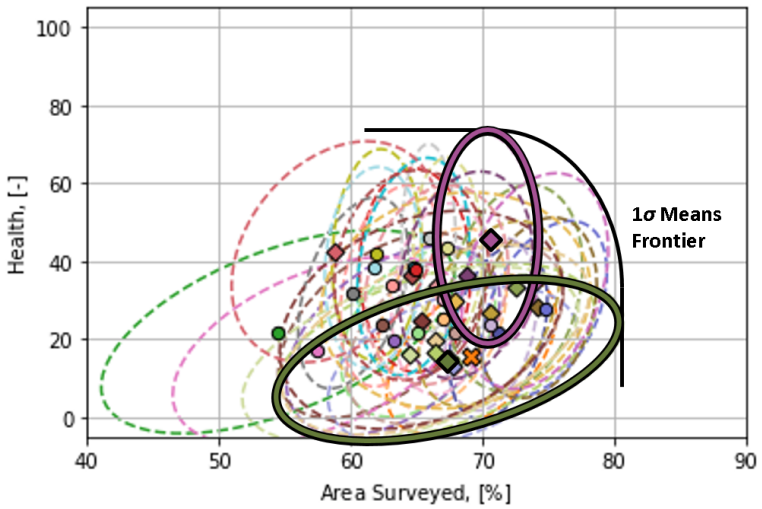

Figure 6.7: The Pareto frontier of 1σ standard deviation away from the mean response of each technology alternative. Legend illustrated in Figure 6.6.

nology bundles and the optics and high speed technology bundles, respectively. Comparing alternative 13 and 34, alternative 13 is considered to be a more robust solution because it provides a smaller standard deviation of the MoEs. This additional insight into the data allows the decision-maker to identify a set of optimal solutions, a set of robust solutions, and a set of solutions which have the highest expected worst MoEs.

The variation of technology alternatives was compared to the variation in tactical alternatives to evaluate the decision behavior algorithm's impact on the MoEs. Figure 6.9 illustrates the baseline technology with and without the variation of tactical alternatives. The variation of tactical alternatives using the decision behavior algorithm enabled a sharp increase in the agent's mean health. This result is expected due to the agent's new ability to engage or evade a defender. However, the increase in the baseline *ways* alternative surprisingly out performs most technology alternatives. These results suggest that the *ways* the baseline technology is used could provide a feasible solution where none existed before. Such a solution would also inform technology investment by providing an alternative to achieving a desired mission effectiveness without the need to invest in a new technology.

The comparison of the *means* versus the *ways* finds that the decision behavior approach

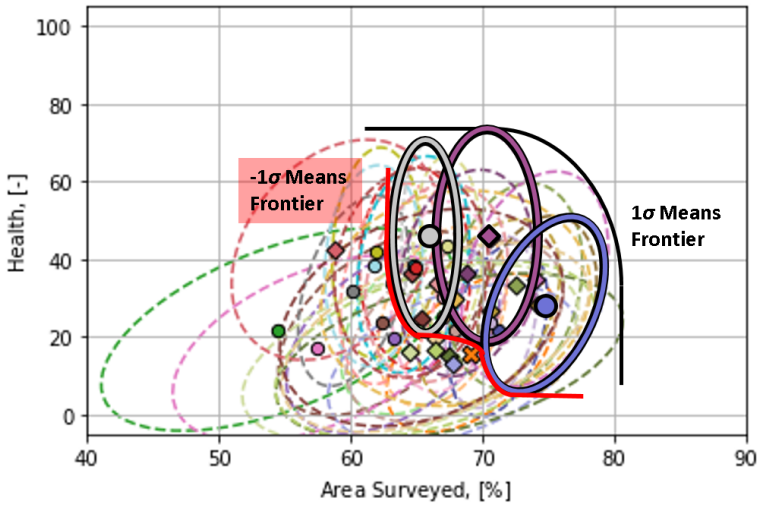

Figure 6.8: The Pareto frontier of 1σ and -1σ standard deviation away from the mean response of each technology alternative. Legend illustrated in Figure 6.6.

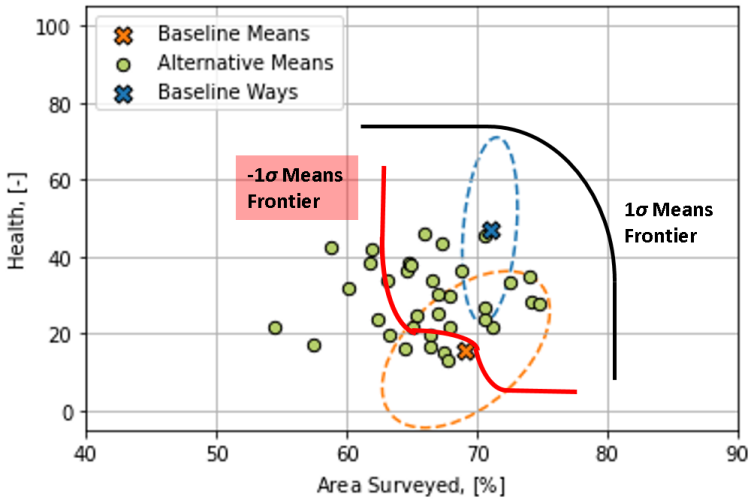

Figure 6.9: The baseline *means* versus the baseline *ways*.

applied to a baseline technology does provide better quantification of the alternative's mission effectiveness than a fixed baseline tactic. Although this finding is expected, the decision behavior approach also provided a competitive solution when compared to other technology alternatives. Identification of new tactics through the decision behavior approach could provide immediate improvements to existing technologies. However, the variation of the *means* and *ways* through the Tech-DEBATE methodology could further augment the quantification of mission effectiveness.

6.3 Variation of the Means and Ways

The *means* and *ways* frontier plot provides a way to visualize the multi-objective design space by quantifying the MoEs for each technology-tactic alternative. The three frontiers, shown in Figure 6.10, illustrate three different methods to improve the MoEs. The *ways* frontier is generated by the tactical solutions which improve the baseline technology. The *means* frontier is formed by exploring new technologies using a baseline tactical alternative. The final frontier illustrates the combination of *means* and *ways* or the technology-tactic synthesis. This frontier further opens up the design space during technology evaluation to provide useful decision-making data that can better quantify the mission effectiveness of a given technology.

Implementation of the Tech-DEBATE methodology enables the quantification of the three frontiers. The novel approach to augment the quantification of mission effectiveness provides the foundation to explore the *means* and *ways*. The MoEs for each technology alternative is investigated with the decision behavior approach to identify the *means* and *ways* frontier for the contested reconnaissance mission. Additional analysis of the results was then conducted to provide further insight on the mission action design space.

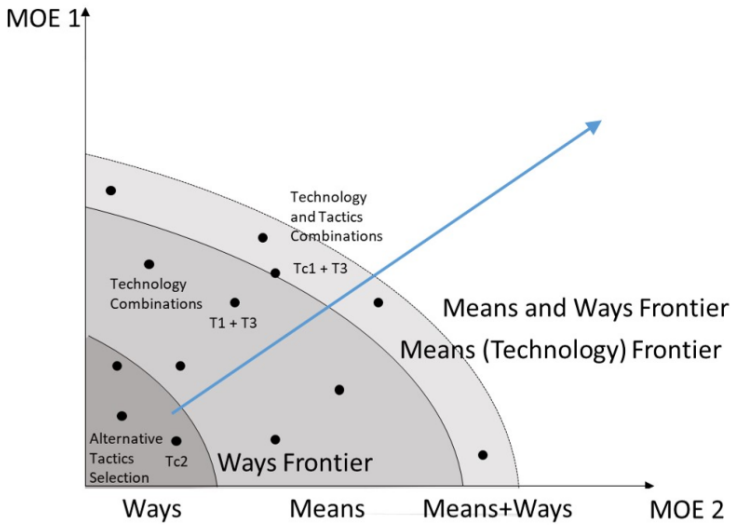

Figure 6.10: The notional *means* and *ways* frontier plot [39].

6.3.1 The Technology-Tuned Decision Behavior Algorithm

The defined decision behavior approach for the contested reconnaissance mission trained 108 algorithms for the 36 technology alternatives. The convergence of each algorithm is illustrated in Appendix B. The decision behavior algorithm that achieved the highest average reward out of the five random initial seeds is utilized for each respective technology. The MoEs for each technology-tuned decision behavior algorithm were determined using the same stochastic mission set used to quantify the *ways* frontier and *means* frontier. The results of the decision behavior approach for each technology alternative are illustrated in Figure 6.11.

The variation of both technology alternatives and tactical alternatives results in significantly improved agent health while also increasing the total area surveyed compared to the baseline technology and tactical alternative. The results enable the identification of the 1σ and -1σ *means* and *ways* frontier using the 1σ standard deviation ellipses for each alternative. Figure 6.12 illustrates the frontier for all technology and tactical alternatives that fall within 1σ of their mean MoEs as well as the set of alternatives that produce the -1σ frontier. Alternatvies 8, 20, and 24 were identified as one possible set of optimal solutions

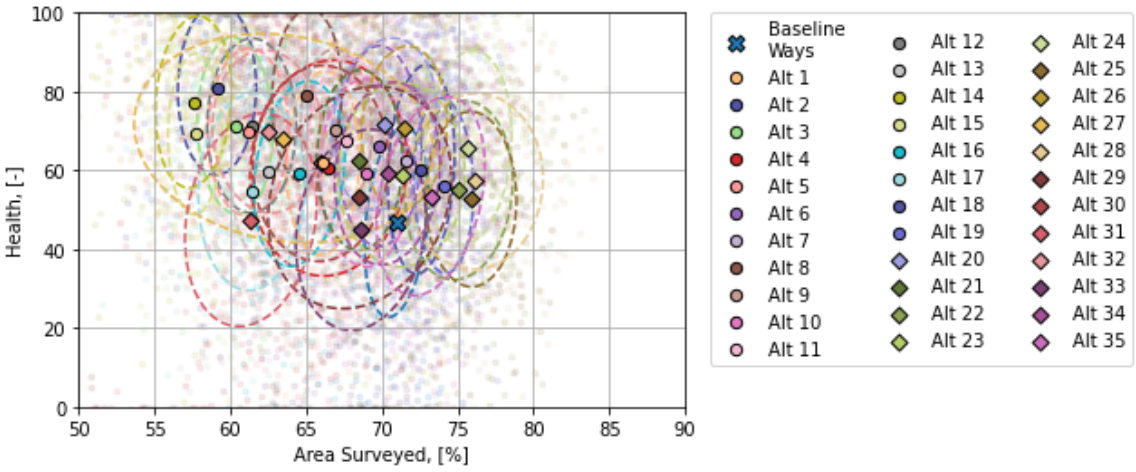

Figure 6.11: The technology-tactic synthesis for the contested reconnaissance mission.

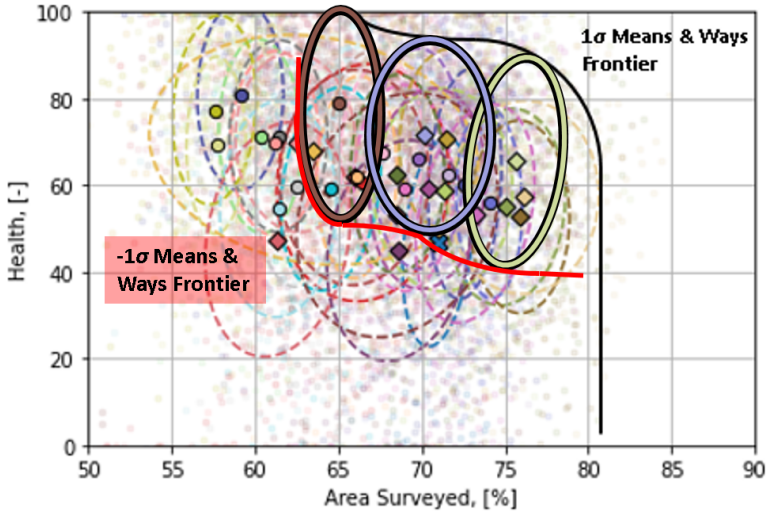

Figure 6.12: The 1σ *means* and *ways* frontier. Legend illustrated in Figure 6.11

that provide both the best performing MoEs and the highest worst performing responses. Note that none of the three identified alternatives are the same optimal solutions identified during variation of only the means. This suggests that variation of the means and ways enables the identification of each technology's optimal tactic that quantifies its true mission effectiveness.

The comparison of each frontier was made to evaluate the impact that the variation of technology and tactical alternatives have. The *ways* frontier illustrates the 1σ frontier for the baseline tactic varying its tactical alternatives. The *means* frontier illustrates the

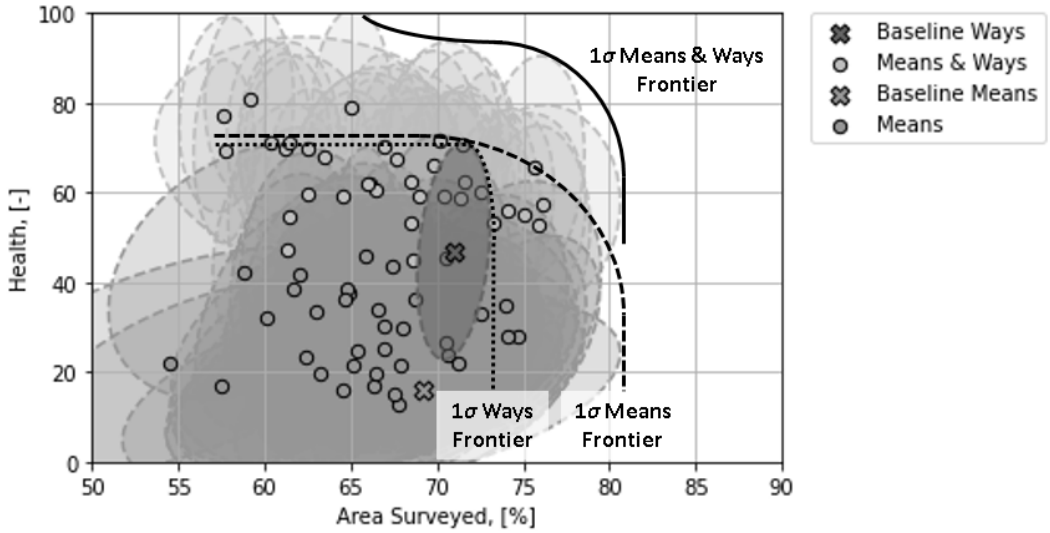

Figure 6.13: The *means* and *ways* frontier for the contested reconnaissance mission.

1σ frontier for the baseline tactical alternative – the reconnaissance state behavior – with 36 technology alternatives. The combination of the tactical alternatives and technology alternatives through the decision behavior approach enables the identification of the *means* and *ways* frontier. All three frontiers are illustrated in Figure 6.13.

The variation of both the technology and tactical alternatives provided significant improvement of the agent's health and provided the same optimality of area surveyed than the variation of just the technology alternatives. The results of the 36 technology alternatives are defined in Table A.2 in Appendix A. The decision behavior approach enabled the exploration of the *means* and *ways* by developing a decision and state behavior agent framework for a minimally defined mission. These results also enabled the analysis of specific solutions on the frontiers to understand the decisions each agent made to achieve their respective MoE.

6.3.2 Analysis of Results

The technology evaluation study produced 36 technology-tuned decision behavior algorithms. Two alternatives are identified from the 1σ *means* and *ways* frontier and compared to the baseline technology. Alternative 8 and 24 both lie on the 1σ Pareto frontier; how-

Table 6.2: The MoEs of the baseline technology alternative along with alternative 8 and 24 from the Pareto frontier.

Technology Alternative	**Health**	**Area Surveyed**
Baseline Alternative	41.91 ± 26.96	$68.96\% \pm 7.19\%$
Alternative 8	79.22 ± 21.99	$64.94\% \pm 2.45\%$
Alternative 24	63.37 ± 26.92	$74.79\% \pm 4.84\%$

ever alternative 8 provides a much higher mean health MoE while alternative 24 provides a higher mean area surveyed. Both technology alternatives had two technologies that included the structure technology bundle. The other technology bundles for the two alternatives were optics and munitions for alternative 8 and 24 respectively. Their MoEs are listed in Table 6.2 along with the baseline technology alternative.

Technology alternative 8 has a smaller standard deviation for both MoEs which implies that its mission effectiveness is more consistent as the stochastic mission scenario changes. This may be desirable to the decision maker due to the higher confidence in the alternative's mission effectiveness. Higher confidence in the data helps ensure the right technology is chosen for further investment. The 'right' technology is one that actually delivers the desired gain in mission effectiveness.

Two tactical alternatives were investigated for each technology alternative: 1) the solutions that resulted in the highest maximum area surveyed and 2) a frontier solution defined as the maximum combined MoE. Table 6.3 summarizes the MoEs for each tactical alternative. Both technology alternatives had the structure technology bundle, which improved the agent's range and maneuverability. However, the second technology bundles seemed to each improve the opposite MoEs. The optics technology bundle improves the agent's field of view by 20% and probability of kill by 5% resulting in higher area surveyed on average for alternative 24. The munitions technology significantly improved the agent's probability of kill by 15% while penalizing the agent's range by -10%. The combination of the structure and munitions technology bundle for alternative 8 results in a much higher health on average for the agent.

Table 6.3: The MoEs for two tactical alternatives for technology alternative 8 and 24.

Technology Alternative	Health	Area Surveyed
8: Maximum Area	78.22	69.95
8: Frontier Solution	99.93	69.01
24: Maximum Area	56.71	82.48
24: Frontier Solution	99.95	79.02

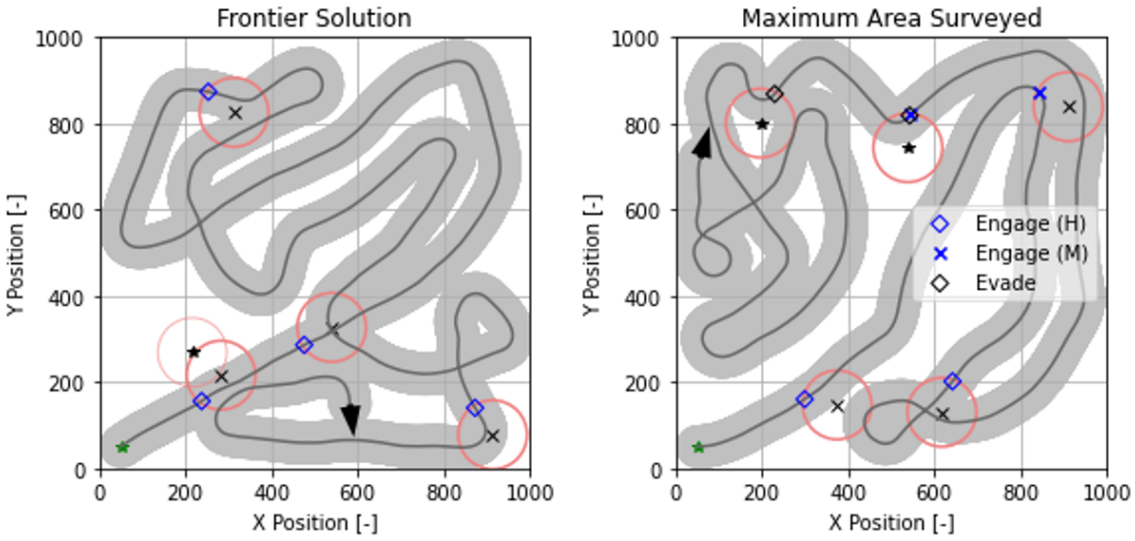

Figure 6.14: The maximum area surveyed and frontier point solutions of alternative 8.

Each tactical alternative is investigated by comparing the decisions made throughout each mission. Figure 6.14 illustrates the decisions that alternative 8 made throughout the mission for both the frontier and the maximum area surveyed point solutions. Both solutions chose to engage the first four defenders it encountered. Once the agent's ammunition of is empty, evasion is chosen. The tactical alternative for the maximum area surveyed point solution evaded the fourth and fifth defenders encountered after unsuccessfully engaging the fourth defender with its last munition.

The tactical alternatives for technology alternative 24 are not too dissimilar to those for alternative 8. Both point solutions, illustrated in Figure 6.15, chose to engage the first three defenders, which corresponds to the agent's initial ammunition of 3. Once its ammunition was empty, the agent chooses to conduct reconnaissance and evasion for the two additional defenders found for the maximum area surveyed point solution.

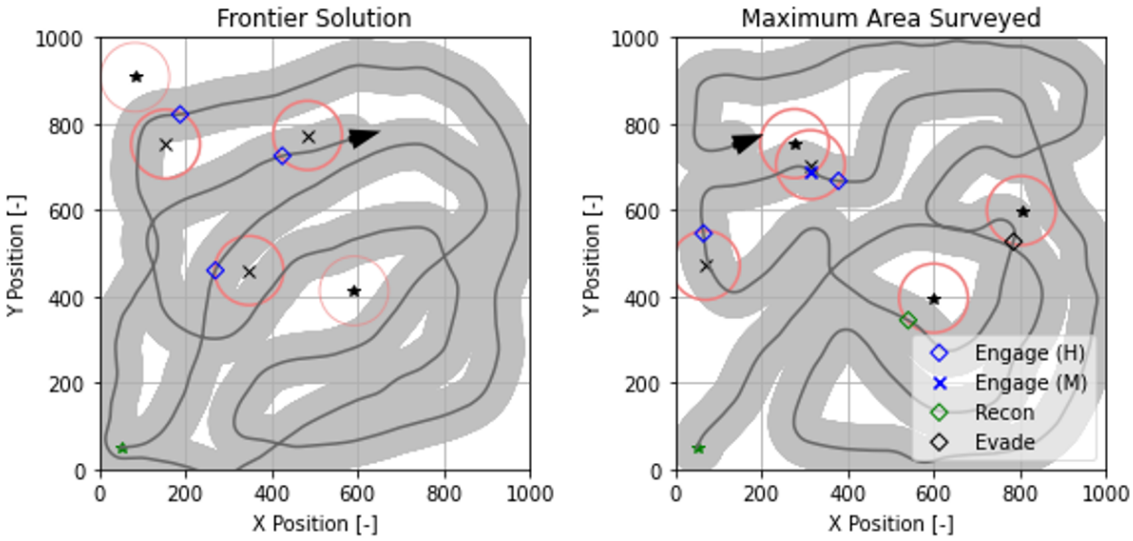

Figure 6.15: The maximum area surveyed and frontier point solutions of alternative 24.

Additional analysis was conducted to verify the results and compare the selected alternatives to the set of all technology alternatives. The agent's health was averaged over all the sample missions as a function of time to understand the rate each technology alternative incurs damage. Figure 6.16 illustrates the agent's average health as a function of time for each technology alternative. Alternative 8 results in the lowest rate of incurred damage as the mission progresses. This result is expected due to the agent's preference to always engage the first four defenders, which corresponds to a maximum initial ammunition of 4. Alternative 24 lands in the middle of the data for the average rate of damage incurred. However, both technology alternatives have a much lower average rate of damage incurred than the baseline technology alternative. The higher ammunition, probability of kill, and maneuverability for the two alternatives contribute to their reduced average rate of damage incurred when compared to the baseline.

The frequency of action selection as the mission progresses was investigated for the baseline alternative and compared to both alternative 8 and 24. Figure 6.17 illustrates a histogram of the first 9 actions selected by the agent through the 200 missions from the stochastic mission set for each technology alternative. Decision numbers over 5 indicate that the agent failed an engagement behavior by either missing the defender or choosing to

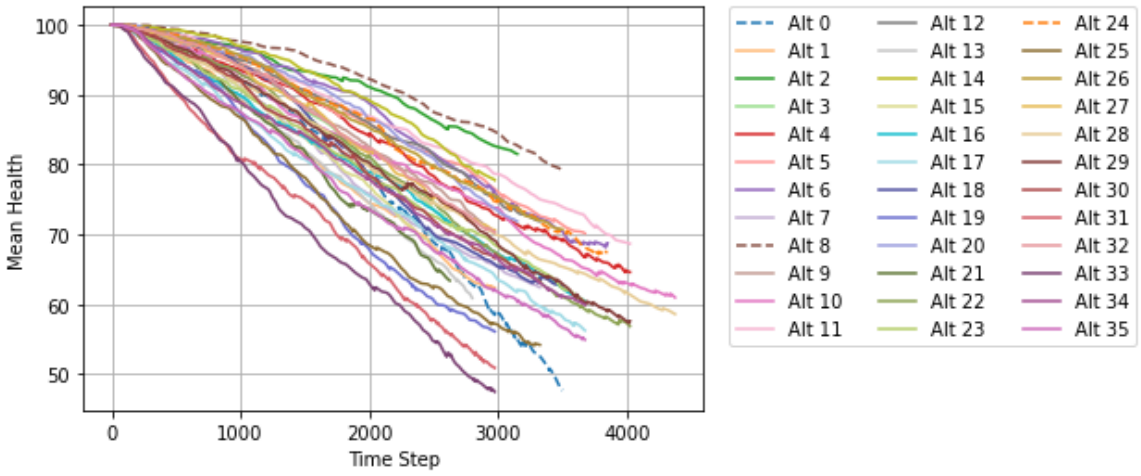

Figure 6.16: The progression of the agent's average health as a function of time from the stochastic mission set.

engage with no ammunition available. The frequency of choosing engagement with zero ammunition is determined to be negligible based on the low frequency of actions past a decision number of 6. The engagement state behavior is found to be heavily favored early on in the mission, but quickly transitions to the evasion state behavior once its ammunition runs out. Although alternatives 8 and 24 have increased maneuverability from the structure technology bundle, alternative 8 is found to favor engagement more than alternative 24.

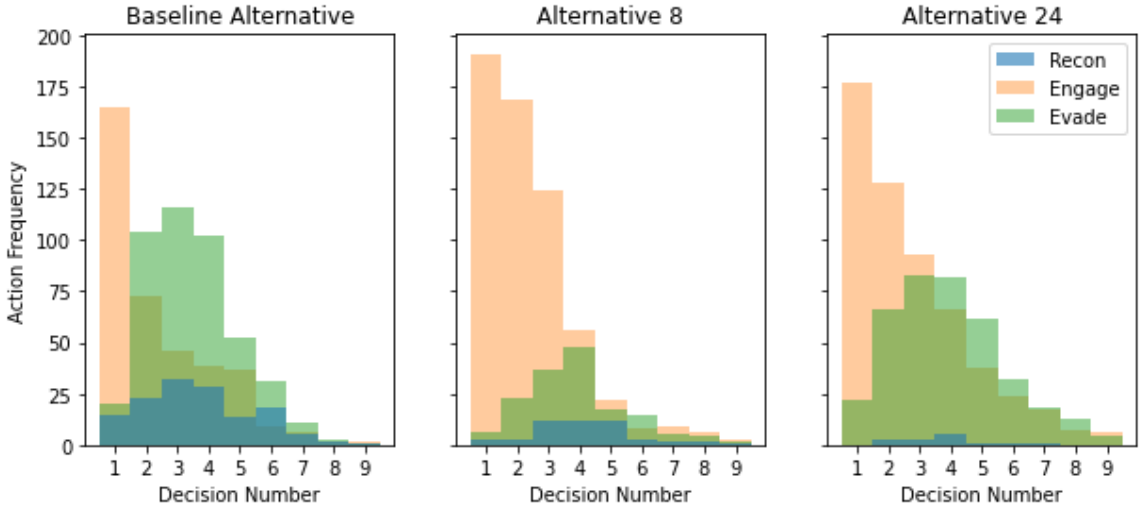

Figure 6.17: The order of decisions the decision behavior algorithm makes as new defenders are encountered.

The identified technology alternatives provide two unique solutions that are both in the set of optimum solutions defined by the frontier. Each solution provides different tactical alternatives for the same contested reconnaissance mission. The point solution investigation provided traceability of the tactical alternatives each chose to achieve their respective MoEs. The optimal solution is one that is identified from a multi-attribute decision making technique that can effectively rank the agent's performance and mission effectiveness based on the decision maker's requirements. The performance and effectiveness of each technology-tactic alternative is used to define their overall system effectiveness.

The MoPs for each technology alternative are evaluated based on the results of the TIM. Alternative 8 has a more effective engagement behavior due to its higher probability of kill of 0.86 compared to alternative 24 which has a value of 0.79. Alternatively, the field of view and range for alternative 24 are larger with a radius of 6 and range of 3850 compared to a radius of 5 and range of 3500 for alternative 8. The system effectiveness of each alternative should then be quantified as the combination of each alternative's MoEs and MoPs. The decision-maker can use this information to make a more informed decision about the appropriate technology to invest in. Additional missions could be investigated to better inform each alternative's capability which would furhter improve the quantification of their system effectiveness. However, additional missions other than the contested reconnaissance mission were not addressed in this thesis.

6.4 Conclusion of the Tech-DEBATE Methodology

The results from the Tech-DEBATE methodology identified a *means* and *ways* frontier based on a 1σ standard deviation from the mean MoE for each technology alternative. Investigation of two points of the frontier found significant improvements in the agents health or mission surveillance by utilizing two unique sets of technology bundles. The overall methodology augments the quantification of mission effectiveness by incorporating stochasticity into the agent's tactical alternatives. This enables the learned algorithms to

explore a minimally defined mission using a non-deterministic set of actions based on the algorithm's policy.

The decision behavior approach introduced the discretization of the agent's framework into decision and state behaviors, enabling the simplification of complex mission behaviors while providing traceability of the agent's decisions. Previous technology evaluation methodologies utilized a fixed mission environment or no mission at all. This lack of emphasis on each alternatives mission effectiveness results in a loss of critical information early on in the design process. Increasing the information about each technology's mission effectiveness provides more information to the decision maker during technology investment decisions. More informed technology investment decisions reduces time and risk associated with the development and acquisition of new technologies.

Alternatively, methodologies that emphasized a minimally defined mission failed to address both the complexity of the mission action design space and the dimensionality of the overall technology evaluation study. The Tech-DEBATE methodology addresses the inherent complexity of the mission action design space by providing steps to capture the decision-state and state-action spaces. Dimensionality of the methodology is also addressed through the introduction of the behavior suitability matrix and decentralized implementation for cooperative mission environments.

The culmination of the Tech-DEBATE methodology is illustrated through the identification of the *means* and *ways* frontier. Implementation of the methodology for the contested reconnaissance mission found an increase in the baseline alternative's MoEs that is competitive with those from the investigation of the *means*. However, when both the technology and tactical alternatives are varied, the design space opens up further, enabling the identification of technology and tactical alternatives. The finding from the implementation of the Tech-DEBATE methodology for th contested reconnaissance mission inform the overall research conclusion.

Overall Research Conclusion: *The proposed Tech-DEBATE methodology augments current technology evaluation methods through the infusion of tactics formulation and exploration which enhances the quantification of mission effectiveness to inform technology selection.*

CHAPTER 7
CONCLUSION

This thesis proposes the Technology-tuned Decision Behavior Algorithms for Tactics Exploration (Tech-DEBATE) Methodology. This methodology is an Agent-Based Modeling & Simulation (ABMS) approach to augment the quantification of mission effectiveness in contested environments. The motivation behind such a methodology resides in improving the development and acquisition processes of the military, specifically technology evaluation methodologies used to select suitable technologies for investment. The novel identification of the *means* and *ways* technology tradespace was presented and demonstrates how it can better inform technology investment.

The development of the novel Tech-DEBATE methodology, illustrated in Figure 7.1, identified five gaps that required further research before it could be implemented. The first two gaps were defined through literature by identifying agent-based modeling with a state machine agent framework. These observations lead to the development of the novel decision behavior approach which enabled the exploration of a minimally defined mission through the discretization of the mission action space into decision and state behaviors. The final three gaps were addressed though the identification of five research questions an hypotheses.

Five experiments were developed and executed which confirmed the stated hypotheses for each research question. The logical diagram, illustrated in Figure 7.2, outlines the mission utilized for the experiments. These experiments further defined the decision behavior approach by identifying deep reinforcement learning algorithms for the agent's decision and state behaviors, a decentralized agent framework to enable the incorporation of cooperative multi-agent systems, and the use of a stochastic agent policy to enable exploration of tactical alternatives. The experiments also informed the procedures of the

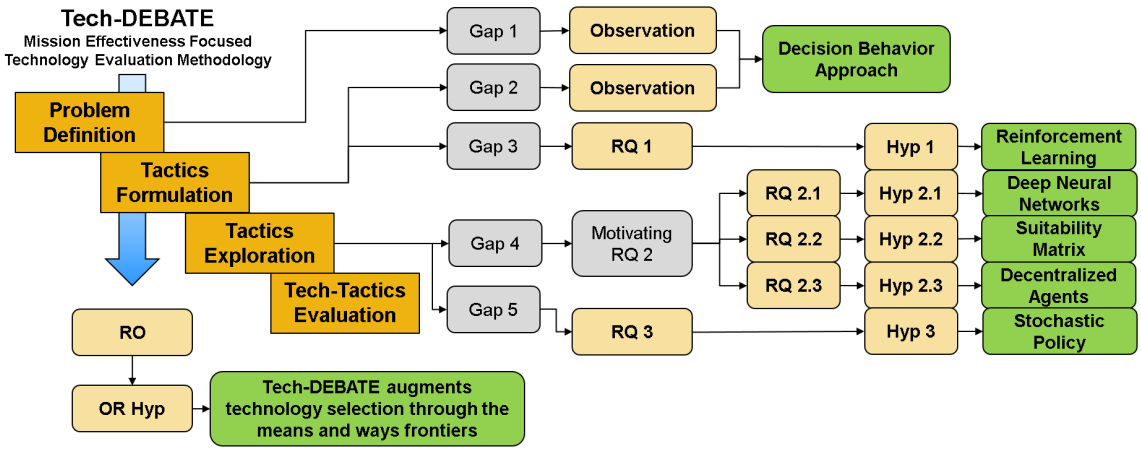

Figure 7.1: Outline of the research approach and development of the Tech-DEBATE methodology.

Tech-DEBATE methodology by introducing the novel behavior suitability matrix to reduce the dimensionality of the technology-tactic evaluation problem and enabling the identification of the *means* and *ways* frontier. The overall research hypothesis was confirmed by conducting a final experiment that evaluated 36 technology alternatives and their optimal tactical alternatives.

Implementation of the methodology utilized a contested reconnaissance mission to conduct a technology and tactics evaluation study. A total of 6 technology bundles were investigated with two technology incompatibilities and two unsuitable technology-behavior pairs which resulted in 36 technology alternatives for a full factorial 2-level design of experiment. Each alternative was evaluated within the specified mission environment and their optimal tactics were determined through the exploration of tactical alternatives. The final experiment confirmed the overall research hypothesis by enabling the novel comparison of technology-tactic alternatives through the *means* and *ways* frontiers.

7.1 Summary of the Tech-DEBATE Methodology

The Tech-DEBATE methodology is comprised of four steps that provide a framework for the formulation and exploration of tactics. This methodology seeks to quantify the mission

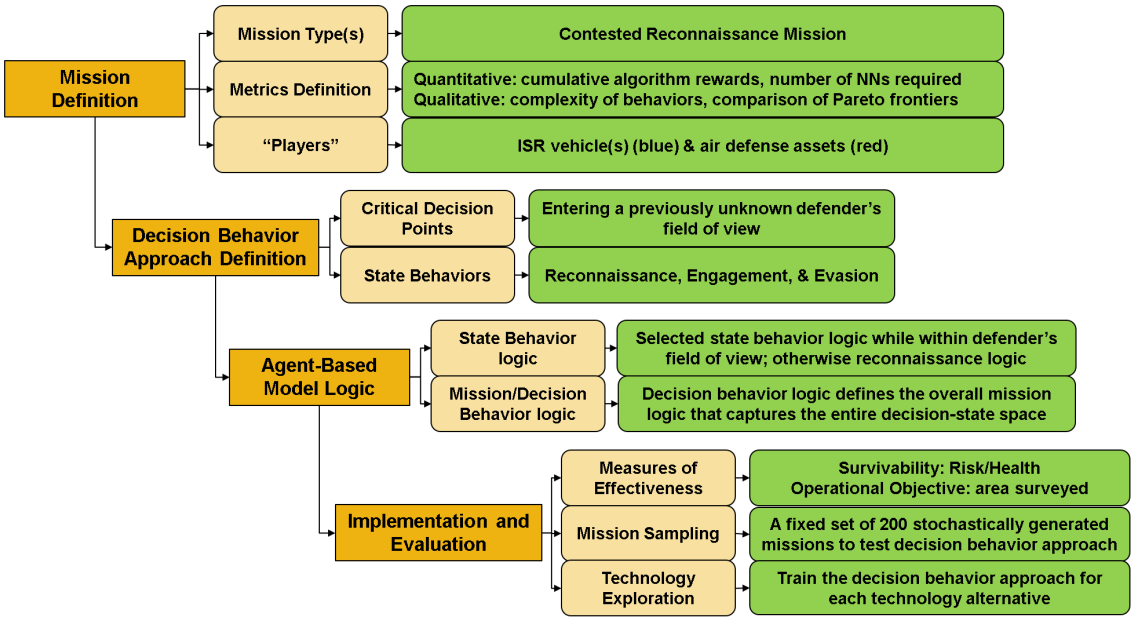

Figure 7.2: Outline of the contested reconnaissance mission experiments.

effectiveness of a technology by exploring its mission action design space. The mission action design space has an immense action space that can quickly become prohibitive. Instead, the mission action space is investigated through the classification of the decision-state and state-action spaces. Exploring the decision points throughout a mission produces a manageable design space without limiting the mission action space. Discretization of the mission action space into discrete decision points provides a traceable way to record the tactics that intelligent agents choose.

Agents within this framework gain their 'intelligence' from a reinforcement learning algorithm. When this algorithm is developed for the decision-making behavior of an agent, it is called a decision behavior algorithm. Such an algorithm can reduce the modeler's dependence on SMEs by removing them from tactics development. SMEs can provide critical information when developing the agent framework, but may introduce bias in the development of tactics. Once the decision behavior function is developed for each technology, the design space can be explored. The novel application of reinforcement learning for technology evaluation enables tactics exploration of evolutionary and revolutionary technologies

whose optimal tactics may be unknown or complex.

The results gained from the synthesis of technologies and tactics provides crucial information to the greater Multi-Attribute Decision-Making (MADM) problem. Technology selection methods of choice can utilize the Measures of Effectiveness (MoEs) from the Tech-DEBATE methodology to make an informed choice for technology investment purposes. This methodology improves current development and acquisition processes by infusing decision-making data with enhanced mission effectiveness to better inform decision-making early in the development and acquisition processes. The proposed Tech-DEBATE Methodology is summarized below and illustrated in Figure 7.3.

Technology-tuned Decision Behavior Algorithm for Tactics Exploration (Tech-DEBATE) Methodology

Step 1: Problem Definition

i. Define the mission, measures of effectiveness, and metrics of success

ii. Identify technologies and mission variables using a morphological matrix

iii. Develop an agent-based model using the decision behavior approach

Step 2: Tactics Formulation

i. Develop agent logic for minimally defined mission

ii. Define a stochastic policy reinforcement learning algorithm for the agent's decision and state behaviors

iii. Identify agent-independent behaviors for cooperative multi-agent systems

Step 3: Tactics Exploration

i. Train the behavior algorithms using deep neural networks

ii. Perturb the ways by sampling the stochastic mission space with the trained stochastic policy from each technology-tuned agent

Step 4: Technology-Tactics Evaluation

i. Compare technology-tactic alternatives through the identification of *means* and *ways* frontiers

ii. Evaluate system effectiveness

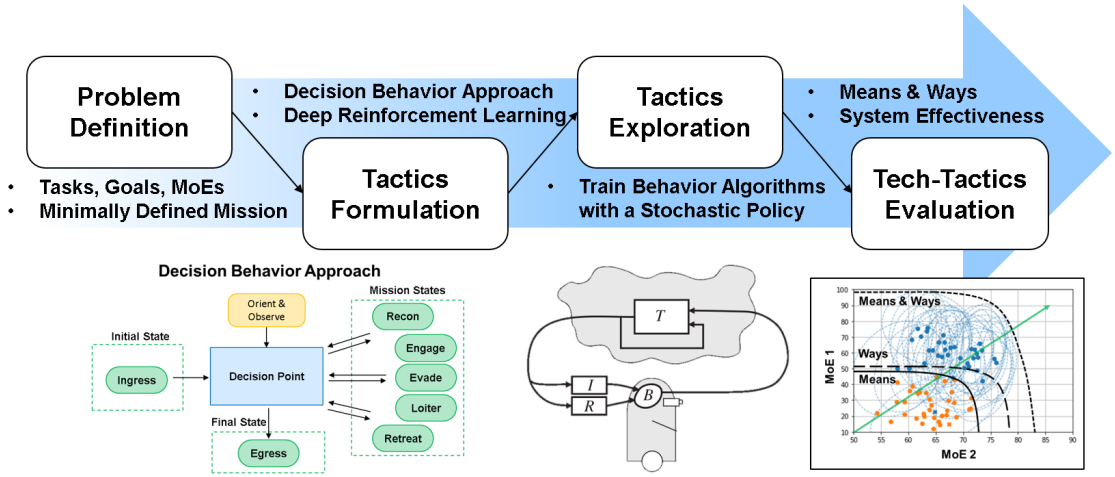

Figure 7.3: The Tech-DEBATE Methodology.

7.2 Contributions

7.2.1 The Decision Behavior Approach

The main technical contribution of this thesis is the decision behavior approach. This agent-based modeling framework discretizes the mission action design space into the decision-state and state action spaces. The proposed agent framework uses a partially observable Markov decision process to define both the agent's decision and state behavior frameworks. Perfect information along with imperfect observations by the agent result in further exploration of the mission action design space by enabling the use of minimally defined missions. The use of deep reinforcement learning with a stochastic policy enables the exploration of tactical alternatives for a technology alternative. The decision behavior approach improves the quantification of mission effectiveness by providing a traceable approach for tactics

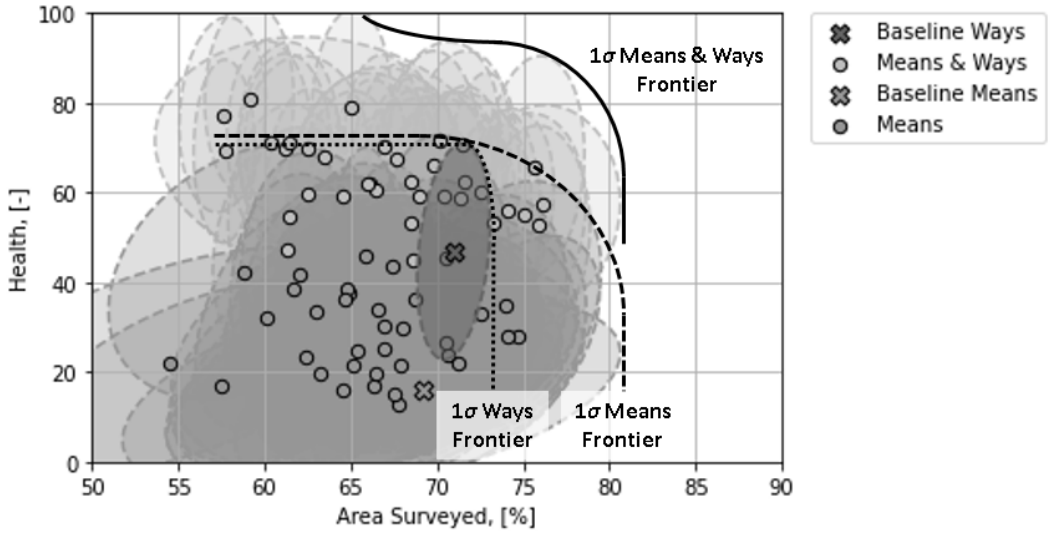

Figure 7.4: Identification of the *means* and *ways* frontier.

exploration. The decision behavior approach provides a novel way to explore tactical alternatives with deep reinforcement learning.

7.2.2 The Tech-DEBATE Methodology

This thesis addresses the gaps in current technology evaluation methodologies to enable the trade-off of the *means* and *ways*. As far as the author is aware, the proposed Tech-DEBATE methodology is the first proposed method to quantify the *means* and *ways* for technology evaluation methodologies. This novel methodology incorporates the decision behavior approach to enable the exploration of both technology alternatives and tactical alternatives in a traceable manner. The methodology enables further exploration of the technology design space by improving the quantification of mission effectiveness through deep reinforcement learning. The data's foundation is based on traceable tactical alternatives that increase the confidence in the measures of effectiveness for each technology-tactic alternative. Figure 7.4 illustrates the progression in the quantification of mission effectiveness by only exploring technology alternatives (*means*), only exploring tactical alternatives (*ways*), and exploring both technology and tactical alternatives (*means* and *ways*).

The methodology enables better informed decisions for technology investment, thereby

reducing risks in the development and acquisition of new technologies. The reduction in risk inherently reduces the costs and development time associated with investment in new technologies. The Tech-DEBATE methodology provides a new methodology for technology evaluation through its emphasis on quantifying mission effectiveness in a minimally defined mission to inform technology investment decisions.

7.3 Implications of the Tech-DEBATE Methodology

The Tech-DEBATE methodology provides critical technology performance and effectiveness information to the decision-maker to better inform technology selection. This methodology emphasizes the quantification of mission effectiveness to ensure the right technology investments are made that can reduce the risks, costs, and life cycle of the US development and acquisition process. Investing in the right technologies early on helps to enable technological superiority in the the expected highly contested environment of 2030 and beyond.

The novel ability to investigate both technology and tactical alternatives in a minimally defined mission simulation provides insight into the larger operational tradespace. The Tech-DEBATE methodology provides a way to quantify the *means*, *ways*, and *ends* for the conceptual design of technology alternatives. The technologies used, tactics employed, and mission objectives met provide one piece to the larger problem of determining strategic and operational concepts that lead to the *ends*. The implications of the methodology not only inform technology selection for military planners, but could also be leveraged to explore the effectiveness of technology-tactic alternatives against adversarial strategies such as the A2/AD strategy. Proper definition of the mission environment could identify or address gaps in current military forces against adversarial strategies. The identification of operational or technological gaps could inform policy makers in addressing adversarial strategies or proper use of revolutionary technology alternatives.

7.4 Future Work

This thesis presents the Tech-DEBATE methodology implemented for a contested reconnaissance mission. The outcome of the methodology aims to quantify the system effectiveness of each technology-tactic alternative. Future work could investigate the implementation of the decision behavior approach across several mission types to quantify the capability of each alternative. The quantification of mission effectiveness across multiple missions would enable the quantification of a technology alternative's capability which would better inform its system effectiveness. Such an implementation could investigate the use of the same set of state behaviors from the decision behavior approach used for several mission types.

The quantification of mission effectiveness for each technology alternative could be further investigated from a defense planner's point of view. Defense planning is defined as "the employment of analytical, planning, and programming efforts to determine what sort of armed forces a state needs" [184]. Investigation of the Tech-DEBATE methodology could be implemented from the defense planner's point of view to quantify the number and type of assets required to defeat adversarial strategies and support the larger US defense strategy. Incorporation of competitive agents could also be investigated to enable the investigation of more adversarial strategies like the A2/AD strategy.

The exploration of tactics in a minimally defined mission could also enable the identification of novel tactics. Such information could inform policy makers on the proper or expected employment of adversarial forces against evolutionary and revolutionary technologies. Insight into the employment of technologies and exploration of tactical alternatives could inform policy that can address the expected highly contested air space in 2030 and beyond. Each area outlined above could expand the Tech-DEBATE methodology to further reduce the risks, costs, and life cycle of the development and acquisition process.

Appendices

APPENDIX A

RESULTS OF THE TECHNOLOGY-TACTIC SYNTHESIS

Table A.1: The agent's metrics for each technology alternative.

Alternative	T1	T2	T3	T4	T5	T6	v	$R_{A/C}$	R_{def}	$\Delta\varphi_{evasion}$	P_k	A	d_{range}
Baseline	0	0	0	0	0	0	20	5.0	8.0	1.20	0.75	2	3500
1	0	0	0	0	0	1	24	5.0	7.2	1.08	0.75	3	2975
2	0	0	0	0	1	0	20	5.0	8.0	1.20	0.86	4	3150
3	0	0	0	0	1	1	24	5.0	7.2	1.08	0.86	3	2625
4	0	0	0	1	0	0	18	5.5	7.2	1.08	0.75	2	4025
5	0	0	0	1	1	0	18	5.5	7.2	1.08	0.86	2	3675
6	0	0	1	0	0	0	20	5.0	8.0	1.32	0.75	4	3850
7	0	0	1	0	0	1	24	5.0	7.2	1.20	0.75	3	3325
8	0	0	1	0	1	0	20	5.0	8.0	1.32	0.86	4	3500
9	0	0	1	0	1	1	24	5.0	7.2	1.20	0.86	3	2975
10	0	0	1	1	0	0	18	5.5	7.2	1.20	0.75	2	4375
11	0	0	1	1	1	0	18	5.5	7.2	1.20	0.86	2	4025
12	0	1	0	0	0	0	20	5.0	6.4	1.20	0.83	2	3325
13	0	1	0	0	0	1	24	5.0	5.6	1.08	0.83	1	2800
14	0	1	0	0	1	0	20	5.0	6.4	1.20	0.94	2	2975
15	0	1	0	0	1	1	24	5.0	5.6	1.08	0.94	1	2450
16	0	1	0	1	0	0	18	5.5	5.6	1.08	0.83	0	3850
17	0	1	0	1	1	0	18	5.5	5.6	1.08	0.94	0	3500
18	1	0	0	0	0	0	20	6.0	8.0	1.20	0.79	3	3500
19	1	0	0	0	0	1	24	6.0	7.2	1.08	0.79	2	2975
20	1	0	0	0	1	0	20	6.0	8.0	1.20	0.90	3	3150
21	1	0	0	0	1	1	24	6.0	7.2	1.08	0.90	2	2625
22	1	0	0	1	0	0	18	6.5	7.2	1.08	0.79	1	4025
23	1	0	0	1	1	0	18	6.5	7.2	1.08	0.90	1	3675
24	1	0	1	0	0	0	20	6.0	8.0	1.32	0.79	3	3850
25	1	0	1	0	0	1	24	6.0	7.2	1.20	0.79	2	3325
26	1	0	1	0	1	0	20	6.0	8.0	1.32	0.90	3	3500
27	1	0	1	0	1	1	24	6.0	7.2	1.20	0.90	2	2975
28	1	0	1	1	0	0	18	6.5	7.2	1.20	0.79	1	4375
29	1	0	1	1	1	0	18	6.5	7.2	1.20	0.90	1	4025
30	1	1	0	0	0	0	20	6.0	6.4	1.20	0.86	1	3325
31	1	1	0	0	0	1	24	6.0	5.6	1.08	0.86	0	2800
32	1	1	0	0	1	0	20	6.0	6.4	1.20	0.98	1	2975
33	1	1	0	0	1	1	24	6.0	5.6	1.08	0.98	0	2450
34	1	1	0	1	0	0	18	6.5	5.6	1.08	0.86	-1	3850
35	1	1	0	1	1	0	18	6.5	5.6	1.08	0.98	-1	3500

Table A.2: The results of the technology-tactic synthesis for the contested reconnaissance mission.

Alternative	T1	T2	T3	T4	T5	T6	Health	Area	Missions Success
Baseline	0	0	0	0	0	0	41.91 ± 26.96	68.96% ± 7.19%	88%
1	0	0	0	0	0	1	61.41 ± 23.70	66.00% ± 2.64%	99%
2	0	0	0	0	1	0	80.81 ± 22.34	58.62% ± 3.74%	99%
3	0	0	0	0	1	1	71.27 ± 22.77	59.91% ± 3.39%	100%
4	0	0	0	1	0	0	58.77 ± 29.33	64.47% ± 9.35%	91%
5	0	0	0	1	1	0	68.41 ± 22.99	60.42% ± 4.69%	98%
6	0	0	1	0	0	0	64.92 ± 27.60	68.67% ± 5.20%	95%
7	0	0	1	0	0	1	61.80 ± 22.06	71.25% ± 3.93%	99%
8	0	0	1	0	1	0	79.22 ± 21.99	64.94% ± 2.45%	100%
9	0	0	1	0	1	1	69.36 ± 21.55	66.43% ± 4.71%	99%
10	0	0	1	1	0	0	56.68 ± 26.20	67.52% ± 7.28%	93%
11	0	0	1	1	1	0	68.33 ± 20.84	67.12% ± 3.42%	100%
12	0	1	0	0	0	0	71.55 ± 21.86	61.17% ± 3.24%	100%
13	0	1	0	0	0	1	60.59 ± 23.14	62.21% ± 2.89%	100%
14	0	1	0	0	1	0	77.39 ± 21.59	57.33% ± 2.91%	100%
15	0	1	0	0	1	1	69.28 ± 20.32	57.64% ± 1.82%	100%
16	0	1	0	1	0	0	58.03 ± 24.80	63.44% ± 6.17%	96%
17	0	1	0	1	1	0	53.70 ± 25.78	60.53% ± 6.01%	96%
18	1	0	0	0	0	0	58.54 ± 27.99	71.61% ± 5.42%	94%
19	1	0	0	0	0	1	54.70 ± 23.59	73.78% ± 2.69%	98%
20	1	0	0	0	1	0	71.51 ± 22.28	70.04% ± 3.61%	100%
21	1	0	0	0	1	1	61.48 ± 24.59	67.61% ± 4.85%	97%
22	1	0	0	1	0	0	53.80 ± 23.77	73.70% ± 7.09%	96%
23	1	0	0	1	1	0	58.25 ± 24.66	70.30% ± 6.33%	97%
24	1	0	1	0	0	0	63.37 ± 26.92	74.79% ± 4.84%	94%
25	1	0	1	0	0	1	52.02 ± 23.04	75.10% ± 4.69%	97%
26	1	0	1	0	1	0	70.73 ± 20.71	71.24% ± 3.12%	100%
27	1	0	1	0	1	1	67.80 ± 26.85	63.37% ± 10.01%	96%
28	1	0	1	1	0	0	55.94 ± 23.89	74.79% ± 6.87%	96%
29	1	0	1	1	1	0	51.84 ± 29.22	67.52% ± 7.77%	91%
30	1	1	0	0	0	0	61.37 ± 25.78	64.31% ± 7.44%	97%
31	1	1	0	0	0	1	44.74 ± 28.50	59.01% ± 8.45%	88%
32	1	1	0	0	1	0	69.62 ± 21.04	61.95% ± 4.64%	99%
33	1	1	0	0	1	1	44.11 ± 25.64	67.85% ± 5.72%	93%
34	1	1	0	1	0	0	56.61 ± 25.62	68.85% ± 7.32%	95%
35	1	1	0	1	1	0	50.79 ± 26.97	71.40% ± 8.02%	93%

APPENDIX B

DECISION & STATE BEHAVIOR CONVERGENCE

The convergence of the reconnaissance and evasion state behavior algorithms for each technology alternative are illustrated in Figures B.1 and B.2. Each alternative is trained from scratch using five unique random seeds. The results from all five seeds are averaged and smoothed using Savitzky-Golay filtering with various windows sizes.

The decision behavior algorithm's convergence for each technology alternative is illustrated in Figure B.3. Each technology alternative is also sampled from five random seeds to produce five separate converged algorithms. Alternatives 16, 17, 31, 33, 34, and 35 resulted in poor convergence due to their lack of the engagement state behavior which adversely impacted the algorithm's decision-state space.

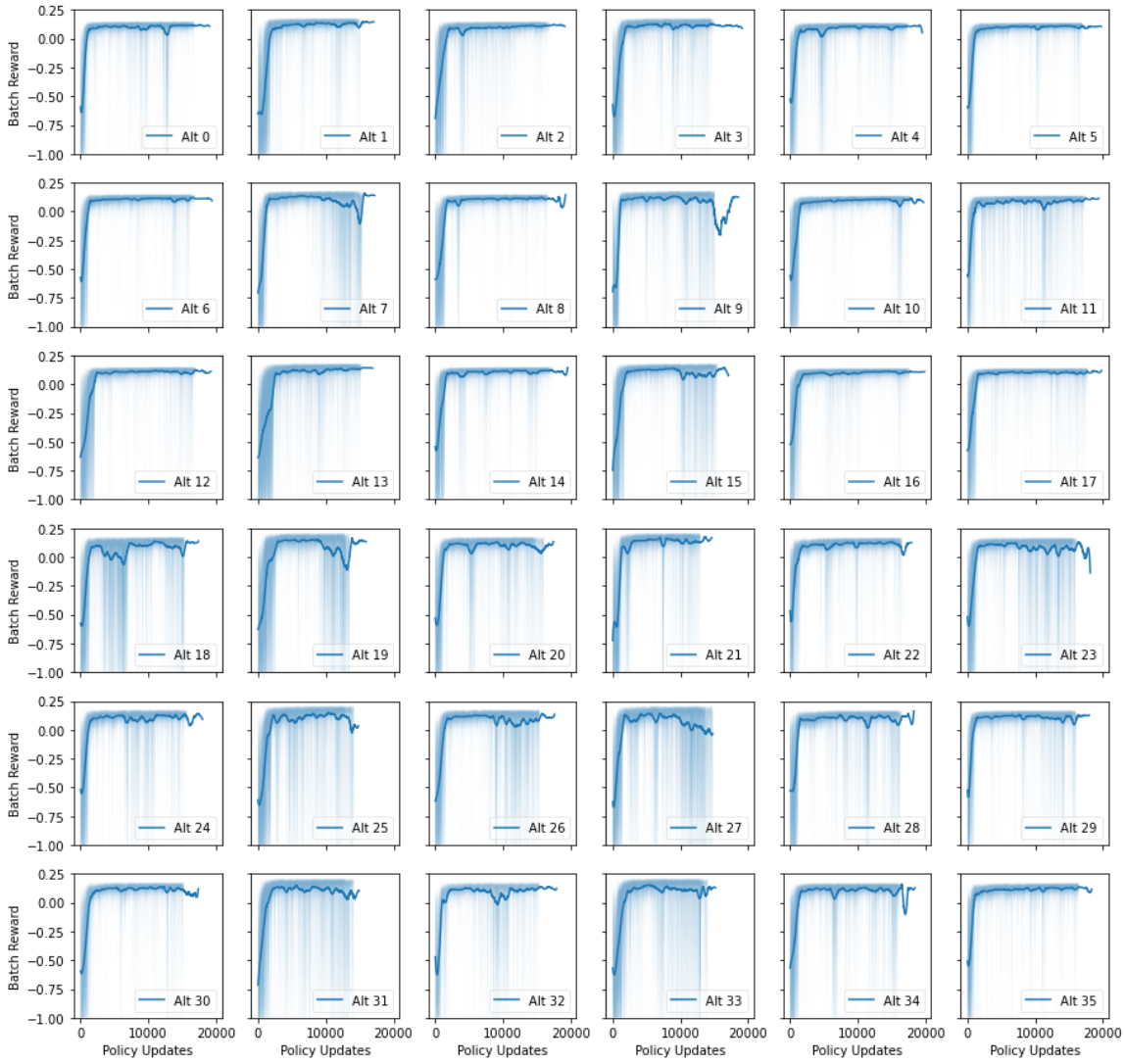

Figure B.1: Batch reward convergence of the reconnaissance behavior algorithm for each technology alternative.

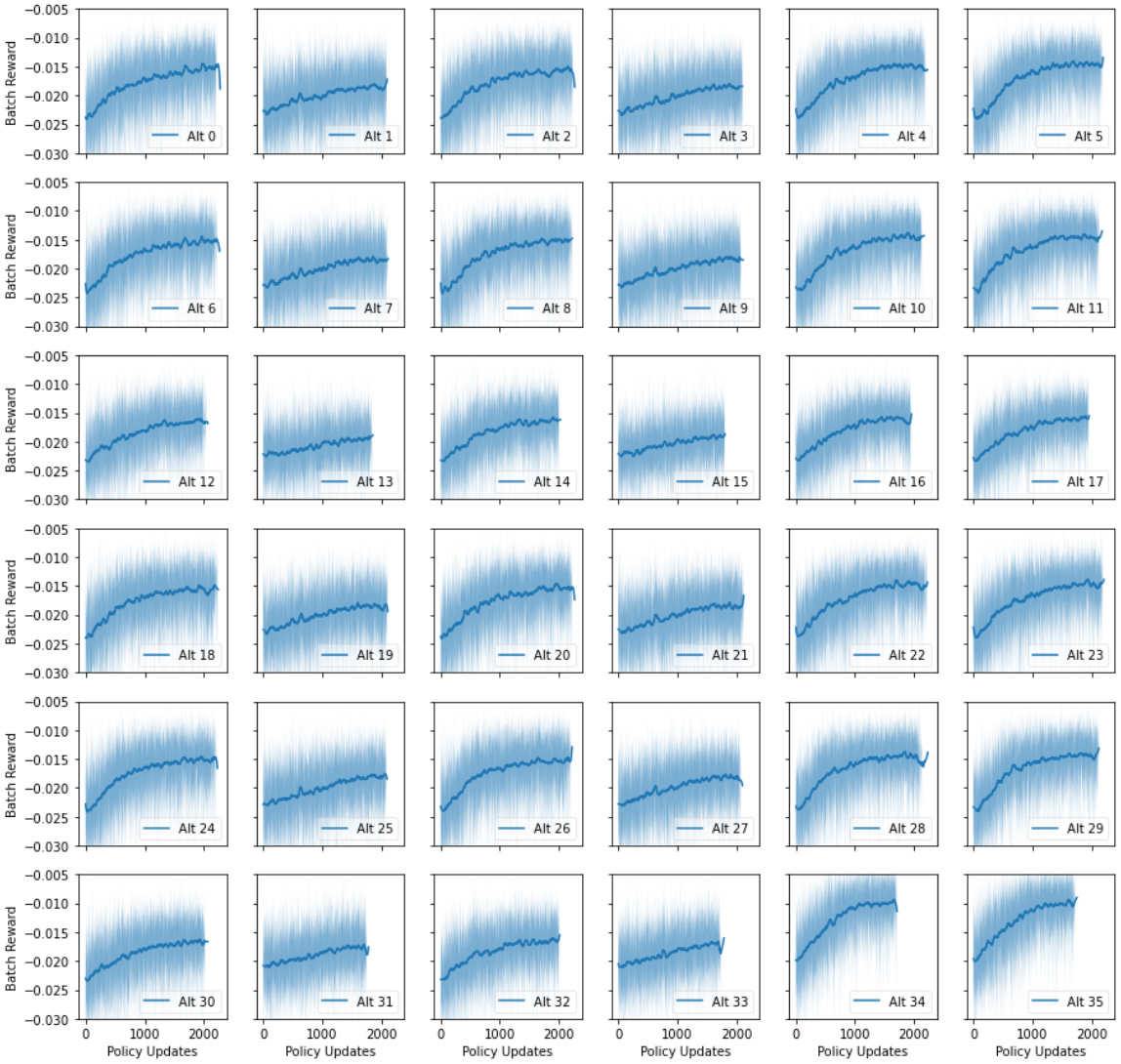

Figure B.2: Batch reward convergence of the evasion behavior algorithm for each technology alternative.

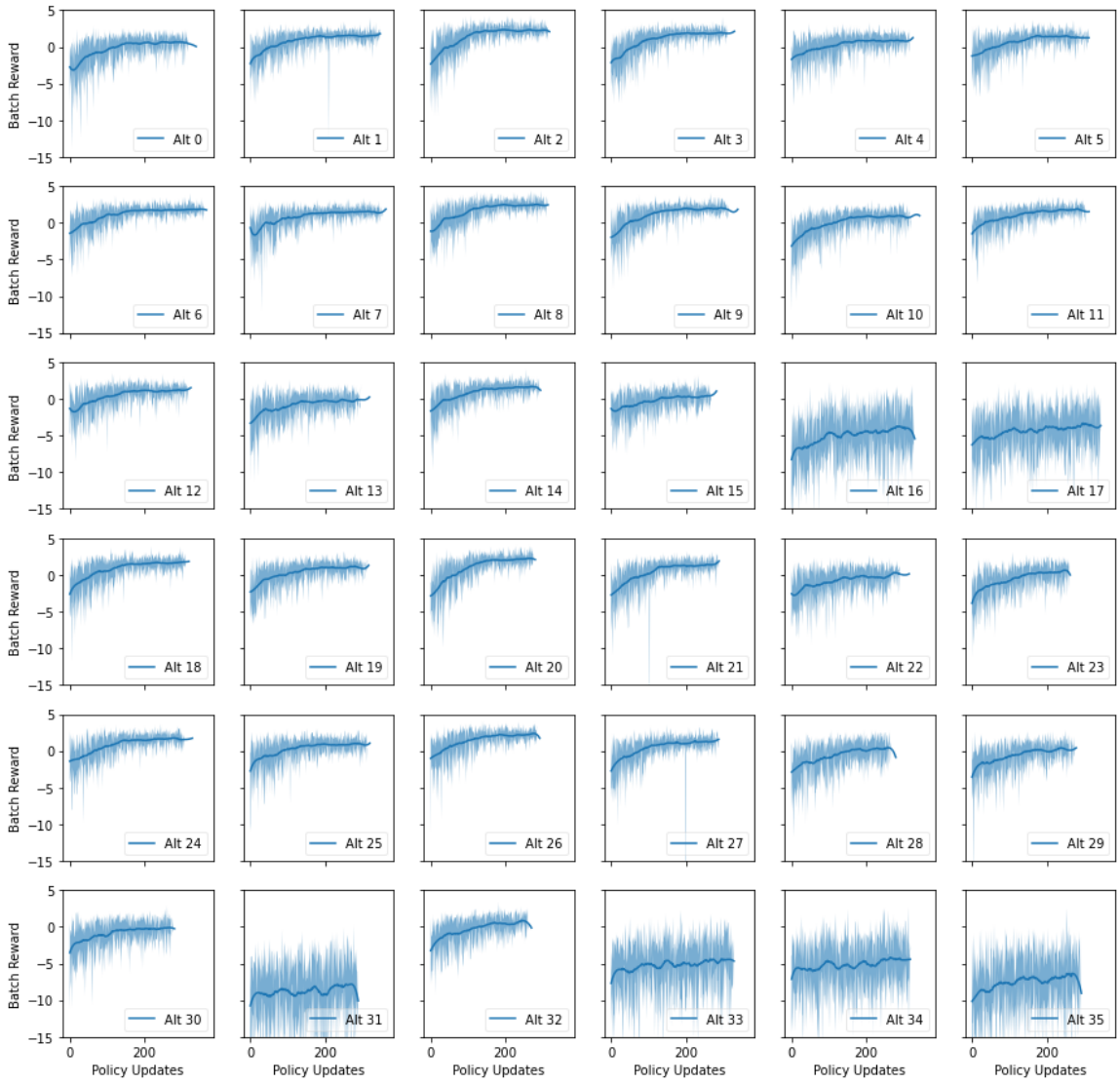

Figure B.3: Batch reward convergence of the decision behavior algorithm for each technology alternative.

REFERENCES

[1] D. D. Chipman, "Airpower A New Way of Warfare (Sea Control)," *Airpower Journal*, 1997.

[2] P. G. Gillespie, *Weapons of Choice: The Development of Precision Guided Munitions*. Tuscaloosa: University of Alabama Press, 2006, ISBN: 0-8173-8189-9.

[3] J. A. I. Warden, "Air Theory for the Twenty-First Century," in *Battlefield of the Future: 21st Century Warfare Issues*, ser. 3, B. R. Schneider and L. E. Grinter, Eds., Air University Press, Maxwell Air Force Base, AL, 1998, ch. 4, pp. 109–124.

[4] United States Air Force, "Air Superiority 2030 Flight Plan," Enterprise Capability Collaboration Team, Tech. Rep., May 2016.

[5] T. G. Mahnken, "Weapons: The Growth & Spread of the Precision-Strike Regime," *Daedalus*, vol. 140, no. 3, pp. 45–57, 2011.

[6] E. A. Cohen, "The Mystique of U.S. Air Power," *Foriegn Affairs*, vol. 73, no. 1, pp. 109–124, Jan. 1994.

[7] P. T. Biltgen, "A Methodology for Capability-Based Technology Evaluation for Systems-of-Systems," PhD thesis, Georgia Institute of Technology, May 2007.

[8] D. H. Rumsfeld, "Transformation Planning Guidance," *Washington, DC: Department of Defense*, pp. 6–7, 2003.

[9] United States Air Force, "Transformation Flight Plan," HQ USAF/XPXC Future Concepts and Transformation Division, Tech. Rep., 2004.

[10] ——, "AF releases Air Superiority 2030 Flight Plan," *Secretary of the Air Force Public Affairs Command Information*, May 2016.

[11] B. G. A. Grynkewich, "The Future of Air Superiority Part III: Defeating A2/AD," *War on the Rocks*, Jan. 2017.

[12] J. M. Acton, "Escalation through entanglement: How the vulnerability of command-and-control systems raises the risks of an inadvertent nuclear war," *International security*, vol. 43, no. 1, pp. 56–99, 2018.

[13] S. Joshi. (Apr. 2019). Demystifying the Anti-Access/Area Denial (A2/AD) Threat, Medium.com.

[14] F. E. Morgan, *Crisis Stability and Long-Range Strike: A Comparative Analysis of Fighters, Bombers, and Missiles*. Santa Monica, CA: RAND Corporation, 2013, ISBN: 9780833078452.

[15] D. Deptula, "Long-Range Strike: 'More Potent,' More Survivable & Cheaper," *Breaking Defense*, Jan. 2017.

[16] K. Button, "Hypersonic Weapons Race," *AIAA: Aerospace America*, Jun. 2018.

[17] D. Gettinger, "Drones operating in syria and iraq," *Center for the Study of the Drone at Bard College*, 2016.

[18] R. P. Haffa Jr and A. Datla, "Joint intelligence, surveillance, and reconnaissance in contested airspace," *Air & Space Power Journal*, vol. 28, no. 3, p. 29, 2014.

[19] A. Gregg, "Military-Industrial Complex Finds a Growth Market in Hypersonic Weaponry," *The Washington Post*, Dec. 2018.

[20] B. G. A. Grynkewich, "The Future of Air Superiority Part II: The 2030 Problem," *War on the Rocks*, Jan. 2017.

[21] D. N. Mavris, D. DeLaurentis, O. Bandte, and M. Hale, "A stochastic approach to multi-disciplinary aircraft analysis and design," in *36th AIAA Aerospace Sciences Meeting and Exhibit*, 1998, p. 912.

[22] D. N. Mavris and D. A. DeLaurentis, "A stochastic design approach for aircraft affordability," *Georgia Institute of Technology*, 1998.

[23] J. W. Dykes, "Tempus: A methodology for model-based robust-optimal design of time-dynamic system identification experiments using variational asymptotic expansions," PhD thesis, Georgia Institute of Technology, 2016.

[24] D. N. Mavris and M. R. Kirby, "Technology identification, evaluation, and selection for commercial transport aircraft," *58th Annual Conference of Society of Allied Weight Engineers, Inc.*, 1999.

[25] M. R. Kirby, "A methodology for technology identification, evaluation, and selection in conceptual and preliminary aircraft design," PhD thesis, Georgia Institute of Technology, 2001.

[26] D. Tejtel, C. Zeune, A. Revels, T. Held, and W. Braisted, "Breathing new life into old processes: An updated approach to vehicle analysis and technology assessment," in *AIAA 5th ATIO and16th Lighter-Than-Air Sys Tech. and Balloon Systems Conferences*, 2005, p. 7304.

[27] D. Caudill, J. Zeh, C. Buell, B. Givens, and D. O'Quinn, "Aerospace vehicle technology assessment & simulation (avtas) mission level simulation system (mls2)," in *AIAA Modeling and Simulation Technologies Conference and exhibit*, 2004, p. 4933.

[28] P. T. Biltgen, "Uncertainty Quantification for Capability-Based Systems-of-Systems Design," in *26th International Congress of the Aeronautical Sciences, Anchorage USA, ICAS2008-1.3*, vol. 3, 2008.

[29] P. D. Collopy and P. M. Hollingsworth, "Value-driven design," *Journal of aircraft*, vol. 48, no. 3, pp. 749–759, 2011.

[30] D. Gorissen, E. Quaranta, M. Ferraro, B. Schumann, J. v. Schaik, M. B. I. Gisbert, A. Keane, and J. Scanlan, "Value-based decision environment: Vision and application," *Journal of Aircraft*, vol. 51, no. 5, pp. 1360–1372, 2014.

[31] C. Davies, M. Stelmack, P. S. Zink, A. De La Garza, and P. Flick, "High fidelity mdo process development and application to fighter strike conceptual design," in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, p. 5490.

[32] W. Hurwitz, S. Donovan, J. Camberos, and B. German, "A Systems Engineering Approach to the Application of Multidisciplinary Design, Analysis and Optimization (MDAO) for Efficient Supersonic Air-Vehicle Exploration (ESAVE)," in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, p. 5491.

[33] R. A. Reuter, S. Iden, R. D. Snyder, and D. L. Allison, "An Overview of the Optimized Integrated Multidisciplinary Systems Program," in *57th AIAA / ASCE / AHS / ASC Structures, Structural Dynamics, and Materials Conference*, 2016, p. 0674.

[34] D. L. Allison and R. M. Kolonay, "Expanded MDO for Effectiveness Based Design Technologies: EXPEDITE Program Introduction," in *2018 Multidisciplinary Analysis and Optimization Conference*, 2018, p. 3419.

[35] C. C. Davies, "Lockheed Martin Overview of the AFRL EXPEDITE Program," in *AIAA Scitech 2019 Forum*, 2019, p. 0173.

[36] A. Papageorgiou, M. Tarkian, K. Amadori, and J. Ölvander, "Multidisciplinary Design Optimization of Aerial Vehicles: A Review of Recent Advancements," *International Journal of Aerospace Engineering*, vol. 2018, 2018.

[37] D. L. Clark, D. L. Allison, H. Bae, and E. Forster, "Metamodeling for effectiveness based aircraft design under uncertainty," in *2018 Multidisciplinary Analysis and Optimization Conference*, 2018, p. 3743.

[38] D. L. Clark Jr, D. L. Allison, H. Bae, and E. E. Forster, "Effectiveness-based design of an aircraft considering mission uncertainties," *Journal of Aircraft*, pp. 1–12, 2019.

[39] D. Mavris and A. Sudol, "Formulation and implementation of a method for technology evaluation of system of systems," in *31st Congress of the International Council of the Aeronautical Sciences*, Georgia Institute of Technology, Belo Horizonte, Brazil, Sep. 2018.

[40] L. Milevski, "Grand strategy and operational art: Companion concepts and their implications for strategy," *Comparative Strategy*, vol. 33, no. 4, pp. 342–353, 2014.

[41] *Analysis of Alternatives (AOA) Handbook: A Practical Guide to the Analysis of Alternatives*, Office of Aerospace Studies, Headquarters Air Force HAF/A5R-OAS 1655 1st Street SE Kirtland AFB, NM 87117, Jul. 2016.

[42] A. Borshchev and A. Filippov, "From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools," in *Proceedings of the 22nd international conference of the system dynamics society*, Citeseer, vol. 22, 2004.

[43] A. J. Turner, "A methodology for the development of models for the simulation of non-observable systems," PhD thesis, Georgia Institute of Technology, 2014.

[44] C.-Y. Fan, P.-S. Fan, and P.-C. Chang, "A system dynamics modeling approach for a military weapon maintenance supply system," *International Journal of Production Economics*, vol. 128, no. 2, pp. 457–469, 2010.

[45] J Shinar and R Tabak, "New results in optimal missile avoidance analysis," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 5, pp. 897–902, 1994.

[46] T. Shima and J. Shinar, "Time-varying linear pursuit-evasion game models with bounded controls," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 3, pp. 425–432, 2002.

[47] T. Shima, "Optimal cooperative pursuit and evasion strategies against a homing missile," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 414–425, 2011.

[48] R. W. Carr and R. Cobb, "An energy based objective for solving an optimal missile evasion problem," in *AIAA Guidance, Navigation, and Control Conference*, 2017, p. 1016.

[49] S. Kang, H. J. Kim, and M.-J. Tahk, "Aerial pursuit-evasion game using nonlinear model predictive guidance," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 7880.

[50] P.-O. Siebers, C. M. Macal, J. Garnett, D. Buxton, and M. Pidd, "Discrete-event simulation is dead, long live agent-based simulation!" *Journal of Simulation*, vol. 4, no. 3, pp. 204–210, 2010.

[51] G. S. Fishman, *Discrete-event simulation : modeling, programming, and analysis*, ser. Springer series in operations research. New York: Springer, 2001, ISBN: 9780387951607.

[52] W. K. V. Chan, Y.-J. Son, and C. M. Macal, "Agent-based simulation tutorial-simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation," in *Proceedings of the 2010 winter simulation conference*, IEEE, 2010, pp. 135–150.

[53] R. R. Hill, J. O. Miller, and G. A. McIntyre, "Simulation analysis: Applications of discrete event simulation modeling to military problems," in *Proceedings of the 33nd conference on Winter simulation*, IEEE Computer Society, 2001, pp. 780–788.

[54] S. A. Alexander, W. R. Brinkley, J. M. Cohen, T. M. Roberts, P. Beery, J. Bubulka, M. C. Kenfield, and J. M. Quilenderino, "Influence of foreign humanitarian assistance/disaster relief in a coastal nation," PhD thesis, Monterey, California. Naval Postgraduate School, 2011.

[55] J. Cohen, J. Quilenderino, J. Bubulka, and E. P. Paulo, "Linking a throughput simulation to a systems dynamics simulation to assess the utility of a us navy foreign humanitarian aid mission," *Defense & Security Analysis*, vol. 29, no. 2, pp. 141–155, 2013.

[56] C. Macal and M. North, "Introductory tutorial: Agent-based modeling and simulation," in *Proceedings of the Winter Simulation Conference 2014*, IEEE, 2014, pp. 6–20.

[57] S. E. Gordon, "A stochastic agent approach (saa) for mission effectiveness," PhD thesis, Georgia Institute of Technology, 2018.

[58] S. A. Tangen, "A methodology for the quantification of doctrine and materiel approaches in a capability-based assessment," PhD thesis, Georgia Institute of Technology, 2009.

[59] F. A. Tillman, C. H. Lie, and C. L. Hwang, "Simulation model of mission effectiveness for military systems," *IEEE Transactions on Reliability*, vol. 27, no. 3, pp. 191–194, 1978.

[60] I. Nathan, "Mission effectiveness model for manned space flight," *IEEE Transactions on Reliability*, vol. 14, no. 2, pp. 84–93, 1965.

[61] C. H. Lie, W. Kuo, F. A. Tillman, and C. Hwang, "Mission effectiveness model for a system with several mission types," *IEEE transactions on reliability*, vol. 33, no. 4, pp. 346–352, 1984.

[62] M. K. Smotherman and R. M. Geist, "Phased mission effectiveness using a non-homogeneous markov reward model," *Reliability Engineering & System Safety*, vol. 27, no. 2, pp. 241–255, 1990.

[63] B. Wade and P. Chang, "New measures of effectiveness for the air and missile defense simulation community," *Phalanx*, vol. 48, no. 4, pp. 49–53, 2015.

[64] N. Sproles, "Formulating measures of effectiveness," *Systems Engineering*, vol. 5, no. 4, pp. 253–263, 2002.

[65] ——, "Identifying success through measures," *Phalanx*, vol. 30, no. 4, pp. 16–31, 1997.

[66] ——, "Coming to grips with measures of effectiveness," *Systems Engineering*, vol. 3, no. 1, pp. 50–58, 2000.

[67] J Joint Staff and V. Suffolk, "Commander's handbook for assessment planning and execution," 2011.

[68] L. Padgham and M. Winikoff, *Developing intelligent agent systems: A practical guide*. John Wiley & Sons, 2005, vol. 13.

[69] J. H. Sheehan, P. H. Deitz, B. E. Bray, B. A. Harris, and A. B. Wong, "The military missions and means framework," Army Material Systems Analysis Activity, Aberdeen Proving Ground, MD, Tech. Rep., 2004.

[70] (Oct. 2019). Tactics, The Merriam-Webster.com Dictionary.

[71] J. Staff, "DoD Dictionary of Military and Associated Terms," *Joint Pub*, pp. 1–02, 2019.

[72] A. P. Frits, "Formulation of an integrated robust design and tactics optimization process for undersea weapon systems," PhD thesis, Georgia Institute of Technology, 2005.

[73] M. D. Devaney, "Plan recognition in a large-scale multi-agent tactical domain," PhD thesis, Georgia Institute of Technology, 2003.

[74] F. Zwicky, "Discovery, invention, research through the morphological approach," *The Macmillian Company*, 1969.

[75] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[76] P. Deadman, D. Robinson, E. Moran, and E. Brondizio, "Colonist household decisionmaking and land-use change in the amazon rainforest: An agent-based simulation," *Environment and Planning B: Planning and Design*, vol. 31, no. 5, pp. 693–709, 2004.

[77] T. Francisco and G. M. Jorge dos Reis, "Evolving predator and prey behaviours with co-evolution using genetic programming and decision trees," in *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, 2008, pp. 1893–1900.

[78] I. Sakellariou, "Agent based modelling and simulation using state machines.," in *SIMULTECH*, 2012, pp. 270–279.

[79] V. Volovoi, *Abridged petri nets*, 2013. arXiv: 1312.2865 [cs.OH].

[80] S. Banisch, *Markov Chain Aggregation for Agent-Based Models*. Springer International Publishing, 2016.

[81] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[82] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2018, ISBN: 9780262352703.

[83] P. J. Haas and G. S. Shedler, "Stochastic petri net representation of discrete event simulations," *IEEE Transactions on Software Engineering*, vol. 15, no. 4, pp. 381–393, 1989.

[84] K. Åström, "Optimal control of markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965.

[85] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[86] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a bdi-architecture," in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, ser. KR'91, Cambridge, MA, USA: Morgan Kaufmann Publishers Inc., 1991, 473–484, ISBN: 1558601651.

[87] M. Bratman *et al.*, *Intention, plans, and practical reason*. Harvard University Press Cambridge, MA, 1987, vol. 10.

[88] C. A. L. Go and W.-H. Lee, "An intelligent belief-desire-intention agent for digital game-based learning," in *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, P. Melin, O. Castillo, E. G. Ramírez, J. Kacprzyk, and W. Pedrycz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. 9, pp. 677–685, ISBN: 978-3-540-72432-2.

[89] M. Winikoff, L. Padgham, and J. Harland, "Simplifying the development of intelligent agents," in *AI 2001: Advances in Artificial Intelligence*, M. Stumptner, D. Corbett, and M. Brooks, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 557–568, ISBN: 978-3-540-45656-8.

[90] A. Haddadi, "Belief-desire-intention agent," *Foundations of distributed artificial intelligence*, vol. 9, p. 169, 1996.

[91] P. R. Cohen and H. J. Levesque, "Intention is choice with commitment," *Artificial intelligence*, vol. 42, no. 2-3, pp. 213–261, 1990.

[92] P. Caillou, B. Gaudou, A. Grignard, C. Q. Truong, and P. Taillandier, "A simple-to-use bdi architecture for agent-based modeling and simulation," in *Advances in Social Simulation 2015*, Springer International Publishing, 2017, pp. 15–28.

[93] D. Galinec and D. Macanga, "Observe, orient, decide and act cycle and pattern-based strategy: Characteristics and complementation," in *Central European Conference on Information and Intelligent Systems*, Faculty of Organization and Informatics Varazdin, 2012, p. 371.

[94] J. Boyd, *A discourse on winning and losing*. Air University Press, Curtis E. LeMay Center for Doctrine Development and …, 2018.

[95] H. Hillaker. (Jan. 2015). Tribute to John R. Boyd. Originally published in the July 1997 issue of Code One Magazine, Code One Online.

[96] S. C. Sari, A. S. Prihatmanto, *et al.*, "Decision system for robosoccer agent based on ooda loop," in *2012 International Conference on System Engineering and Technology (ICSET)*, IEEE, 2012, pp. 1–7.

[97] F. Fitzek and M. Katz, *Cognitive Wireless Networks: Concepts, Methodologies and Visions Inspiring the Age of Enlightenment of Wireless Communications*. Springer Netherlands, 2007, ISBN: 9781402059797.

[98] S. K. Das, "Modeling intelligent decision-making command and control agents: An application to air defense," *IEEE Intelligent Systems*, vol. 29, no. 5, pp. 22–29, 2014.

[99] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences*, vol. 99, no. Supplement 3, pp. 7280–7287, 2002.

[100] E. Sonmezocak and S. Kurt, "Optimum route planning and scheduling for unmanned aerial vehicles," Naval Postgraduate School Monteray CA, Tech. Rep., 2008.

[101] R. W. Carr, "Optimal Control Methods For Missile Evasion," PhD thesis, Air Force Institute of Technology, 2017.

[102] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1768–1778, 2019.

[103] P. Dayan and N. D. Daw, "Decision theory, reinforcement learning, and the brain," *Cognitive, Affective, & Behavioral Neuroscience*, vol. 8, no. 4, pp. 429–453, 2008.

[104] S. Parsons and M. Wooldridge, "Game theory and decision theory in multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 3, pp. 243–254, 2002.

[105] A. Klabunde and F. Willekens, "Decision-making in agent-based models of migration: State of the art and challenges," *European Journal of Population*, vol. 32, no. 1, pp. 73–97, 2016.

[106] C. Langlotz, E. H. Shortliffe, and L. M. Fagan, "Using decision theory to justify heuristics.," in *AAAI*, 1986, pp. 215–219.

[107] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex systems*, vol. 6, pp. 333–362, 1991.

[108] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[109] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning

algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[110] K.-H. Park, Y.-J. Kim, and J.-H. Kim, "Modular q-learning based multi-agent cooperation for robot soccer," *Robotics and Autonomous Systems*, vol. 35, no. 2, pp. 109–122, 2001.

[111] M. Ponsen, P. Spronck, H. Munoz-Avila, and D. W. Aha, "Knowledge acquisition for adaptive game ai," *Science of Computer Programming*, vol. 67, no. 1, pp. 59–75, 2007.

[112] S. N Sivanandam, *Introduction to genetic algorithms / S.N. Sivanandam, S.N. Deepa.* Berlin ; New York: Springer, 2008, ISBN: 9783540731894.

[113] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[114] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, *Evolution strategies as a scalable alternative to reinforcement learning*, 2017. arXiv: 1703.03864 [stat.ML].

[115] O. Chebbi and J. Chaouachi, "Effective parameter tuning for genetic algorithm to solve a real world transportation problem," in *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, IEEE, 2015, pp. 370–375.

[116] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[117] N. Ono and K. Fukumoto, "A modular approach to multi-agent reinforcement learning," in *Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments*, Springer, 1996, pp. 25–39.

[118] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[119] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*, Springer, 2017, pp. 66–83.

[120] K. Jolly, K. Ravindran, R Vijayakumar, and R. S. Kumar, "Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 589–596, 2007.

[121] D. Daberkow and D. Mavris, "An investigation of metamodeling techniques for complex systems design," in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002, p. 5457.

[122] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," *Cadence Design Systems Inc.: San Jose, CA, USA*, pp. 1–12, 2015.

[123] D. D. Daberkow and D. N. Mavris, "New approaches to conceptual and preliminary aircraft design: A comparative assessment of a neural network formulation and a response surface methodology," *Winter Aviation Conference*, 1998.

[124] H. Scheffé, *The Analysis of Variance*, ser. Wiley Classics Library. Wiley, 1999, ISBN: 9780471345053.

[125] L. Wilkinson, "Revising the pareto chart," *The American Statistician*, vol. 60, no. 4, pp. 332–334, 2006.

[126] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[127] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *In Proceedings of the Seventeenth International Conference on Machine Learning*, Citeseer, 2000.

[128] H. R. Berenji and D. Vengerov, "Advantages of cooperation between reinforcement learning agents in difficult stochastic problems," in *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*, IEEE, vol. 2, 2000, pp. 871–876.

[129] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," *AAAI/IAAI*, vol. 2002, pp. 326–331, 2002.

[130] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[131] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in multi-agent systems and applications-1*, Springer, 2010, pp. 183–221.

[132] M. Bowling, "Convergence and no-regret in multiagent learning," in *Advances in neural information processing systems*, 2005, pp. 209–216.

[133] S. D. Whitehead, "A complexity analysis of cooperative mechanisms in reinforcement learning.," in *AAAI*, 1991, pp. 607–613.

[134] P. Xuan and V. Lesser, "Multi-agent policies: From centralized ones to decentralized ones," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, 2002, pp. 1098–1105.

[135] H. Wei, X. Liu, L. Mashayekhy, and K. Decker, "Mixed-autonomy traffic control with proximal policy optimization," in *IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–8.

[136] D. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of uavs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[137] Y.-J. Lai, T.-Y. Liu, and C.-L. Hwang, "Topsis for modm," *European journal of operational research*, vol. 76, no. 3, pp. 486–500, 1994.

[138] B. Roth and D. Mavris, "Evaluation and selection of technology concepts for a hypersonic high speed standoff missile," Georgia Institute of Technology, Tech. Rep., 1999.

[139] M. Behzadian, S. K. Otaghsara, M. Yazdani, and J. Ignatius, "A state-of the-art survey of topsis applications," *Expert Systems with applications*, vol. 39, no. 17, pp. 13 051–13 069, 2012.

[140] H. Jiménez and D. N. Mavris, "Conceptual design of current technology and advanced concepts for an efficient multi-mach aircraft," *SAE transactions*, pp. 1343–1353, 2005.

[141] S. Briceno, M. Buonanno, I. Fernández, and D. Mavris, "A parametric exploration of supersonic business jet concepts utilizing response surfaces," in *AIAA's Aircraft Technology, Integration, and Operations (ATIO) 2002 Technical Forum*, 2001, p. 5828.

[142] P. T. Biltgen, D. N. Mavris, and G Molter, "A methodology for technology evaluation and capability trade-off for complex system architectures," in *25th International Congress of the Aeronautical Sciences*, 2006.

[143] J. Xie, Y. Cai, M. Chen, and D. N. Mavris, "Integrated sizing and optimization of hybrid wing body aircraft in conceptual design," in *AIAA Aviation 2019 Forum*, 2019, p. 2885.

[144] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, 2019. arXiv: 1509.02971 [cs.LG].

[145] M. L. Koga, V. Freire, and A. H. R. Costa, "Stochastic abstract policies: Generalizing knowledge to improve reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 77–88, 2015.

[146] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Learning without state-estimation in partially observable markovian decision processes," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds., San Francisco (CA): Morgan Kaufmann, 1994, pp. 284–292, ISBN: 978-1-55860-335-6.

[147] S. D. Whitehead and D. H. Ballard, "Learning to perceive and act by trial and error," *Machine Learning*, vol. 7, no. 1, pp. 45–83, 1991.

[148] L. Chrisman, "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach," in *AAAI*, Citeseer, vol. 1992, 1992, pp. 183–188.

[149] N. Heess, G. Wayne, D. Silver, T. P. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," *CoRR*, vol. abs/1510.09142, 2015. arXiv: 1510.09142.

[150] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, pp. 1889–1897.

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

[152] A. Lotov, V. Bushenkov, and G. Kamenev, *Interactive Decision Maps: Approximation and Visualization of Pareto Frontier*, ser. Applied Optimization. Springer US, 2013, ISBN: 9781441988515.

[153] R. Chiong, J. H. Sutanto, and W. J. Jap, "A comparative study on informed and uninformed search for intelligent travel planning in borneo island," in *2008 International Symposium on Information Technology*, IEEE, vol. 3, 2008, pp. 1–5.

[154] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[155] C. J. C. H. Watkins, "Learning from delayed rewards," PhD thesis, King's College, Cambridge United Kingdom, 1989.

[156] H. Van Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[157] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, *et al.*, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv preprint arXiv:2006.05990*, 2020.

[158] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[159] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[160] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn what not to learn: Action elimination with deep reinforcement learning," *arXiv preprint arXiv:1809.02121*, 2018.

[161] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[162] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[163] A. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[164] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[165] O. Delalleau, M. Peter, E. Alonso, and A. Logut, "Discrete and continuous action representation for practical rl in video games," *arXiv preprint arXiv:1912.11077*, 2019.

[166] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[167] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[168] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 2016, pp. 2829–2838.

[169] J.-S. Byun, B. Kim, and H. Wang, "Proximal policy gradient: Ppo with policy gradient," *arXiv preprint arXiv:2010.09933*, 2020.

[170] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 151–160.

[171] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[172] J. Schulman, X. Chen, and P. Abbeel, "Equivalence between policy gradients and soft q-learning," *arXiv preprint arXiv:1704.06440*, 2017.

[173] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 1861–1870.

[174] C. C.-Y. Hsu, C. Mendler-Dünner, and M. Hardt, *Revisiting design choices in proximal policy optimization*, 2020. arXiv: `2009.10897 [cs.LG]`.

[175] N. Barhate, *Minimal pytorch implementation of proximal policy optimization*, `https://github.com/nikhilbarhate99/PPO-PyTorch`, 2021.

[176] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[177] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[178] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" In *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 2146–2153.

[179] B. Yuan, "Efficient hardware architecture of softmax layer in deep neural network," in *2016 29th IEEE International System-on-Chip Conference (SOCC)*, 2016, pp. 323–326.

[180] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of mappo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.

[181] B. Wang, W. Shi, and Z. Miao, "Confidence analysis of standard deviational ellipse and its extension into higher dimensional euclidean space," *PLOS ONE*, vol. 10, no. 3, pp. 1–17, Mar. 2015.

[182] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[183] P. Filzmoser, R. G. Garrett, and C. Reimann, "Multivariate outlier detection in exploration geochemistry," *Computers & Geosciences*, vol. 31, no. 5, pp. 579–587, 2005.

[184] M. J. Mazarr, K. L. Best, B. Laird, E. V. Larson, M. E. Linick, and D. Madden, "The us department of defense's planning process: Components and challenges," RAND Corporation Santa Monica United States, Tech. Rep., 2019.