2022-12

BEHAVIORAL COMPOSITION FOR HETEROGENEOUS SWARMS

Reimer, Beau

Monterey, CA; Naval Postgraduate School

https://hdl.handle.net/10945/71532

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

BEHAVIORAL COMPOSITION FOR HETEROGENEOUS SWARMS

by

Beau Reimer

December 2022

Thesis Advisor:	Duane T. Davis
Co-Advisor:	Kathleen B. Giles

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE BEHAVIORAL COMPOSITION FOR HETEROGENEOUS SWARMS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Beau Reimer	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

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

13. ABSTRACT (maximum 200 words)

Research into swarm robotics has produced a robust library of swarm behaviors that excel at defined tasks such as flocking and area search, many of which have potential for application to a wide range of military problems. However, to be successfully applied to an operational environment, swarms must be flexible enough to achieve a wide array of specific objectives and usable enough to be configured and employed by lay operators. This research explored the use of the Mission-based Architecture for Swarm Composability (MASC) to develop mission-specific tactics as compositions of more general, reusable plays for use with the Advanced Robotic Systems Engineering Laboratory (ARSENL) swarm system. Three tactics were developed to conduct autonomous search of a geographic area and investigation of generated contacts of interest. The tactics were tested in live-flight and virtual environment experiments and compared to a preexisting monolithic behavior implementation completing the same task. Measures of performance were defined and observed that verified the effectiveness of solutions and confirmed the advantages that composition provides with respect to reusability and rapid development of increasingly complex behaviors.

14. SUBJECT TERMS robotic swarm, composable behavior, Mission-based Architecture for Swarm Composability, MASC, Advanced Robotic Systems Engineering Laboratory, ARSENL	**15. NUMBER OF PAGES** 111
	16. PRICE CODE

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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

BEHAVIORAL COMPOSITION FOR HETEROGENEOUS SWARMS

Beau Reimer
Lieutenant Commander, United States Navy
BS, University of Maryland, Baltimore County, 2003
BS, University of Maryland, University College, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2022

Approved by: Duane T. Davis
Advisor

Kathleen B. Giles
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Research into swarm robotics has produced a robust library of swarm behaviors that excel at defined tasks such as flocking and area search, many of which have potential for application to a wide range of military problems. However, to be successfully applied to an operational environment, swarms must be flexible enough to achieve a wide array of specific objectives and usable enough to be configured and employed by lay operators. This research explored the use of the Mission-based Architecture for Swarm Composability (MASC) to develop mission-specific tactics as compositions of more general, reusable plays for use with the Advanced Robotic Systems Engineering Laboratory (ARSENL) swarm system. Three tactics were developed to conduct autonomous search of a geographic area and investigation of generated contacts of interest. The tactics were tested in live-flight and virtual environment experiments and compared to a preexisting monolithic behavior implementation completing the same task. Measures of performance were defined and observed that verified the effectiveness of solutions and confirmed the advantages that composition provides with respect to reusability and rapid development of increasingly complex behaviors.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AI	artificial intelligence
ARGoS	Autonomous Robots Go Swarming
ARSENL	Advanced Robotic Systems Engineering Laboratory
CBD	Component-Based Development
C2	command and control
DARPA	Defense Advanced Research Projects Agency
HSI	human-system interface
LOCUST	Low Cost UAV Swarming Technology
MASC	Mission-based Architecture for Swarm Composability
MRTA	multi-robot task allocation
NAVAIR	Naval Air Systems Command
NPS	Naval Postgraduate School
OFFSET	OFFensive Swarm-Enabled Tactics
ONR	Office of Naval Research
SITL	software-in-the-loop
STP	Skills, Tactics, and Plays
UAS	unmanned aerial systems
UAV	unmanned aerial vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

First and foremost, I cannot adequately express the sincere gratitude for my advisors, Dr. Duane Davis and CDR Kathleen Giles, and their enduring support up to very end of this journey. Their unwavering optimism, confidence, and encouragement toward completion of this thesis was the foundation that sustained my progress through even the most challenging times.

To my wife Rebecca and two daughters, thank you for your unquestionable love and support. You never questioned or complained when family time was sacrificed for study or writing, but instead encouraged me to keep going. I hope to make the lost weekends up to you many times over in the years to come.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Motivation

The U.S. military has a long history of research and utilization of unmanned aerial vehicles (UAVs), predating World War II. More recently, introduction of the Predator UAV, first as a reconnaissance platform and later as an armed combatant revolutionized modern warfare. The advantages of using a Predator-type UAV are numerous and well documented, however this class of systems does not provide a universal solution for every type of mission. System availability, portability, logistic and maintenance requirements, manpower specialization, and ethical implications are just some of the factors that limit the propagation of UAVs to all levels of the operational force. Some believe that the next revolution in warfare will stem from swarm technologies: large numbers of low-cost autonomous vehicles that employ cooperative behaviors and decentralized control to achieve mission objectives [1]–[3].

Considerable work to extend the behavior, capabilities, and command and control (C2) of UAV swarms has been conducted in the last decade. Previous research within the Naval Postgraduate School's Advanced Robotic Systems Engineering Laboratory (ARSENL) group has advanced development of mission-focused approaches to C2 [4] and swarm autonomy via decentralized dynamic task allocation [5]. However, designing swarm systems still presents unique challenges in describing high-level behaviors and objectives and implementing them in a robust distributed system of robotic agents. Current behavior implementations tend to be monolithic, and effective design requires expert programming. This research explores methods for composing primitive swarm behaviors within a hierarchical mission-oriented framework to achieve complex mission objectives autonomously. The application of behavioral composition techniques within a mission-oriented framework promotes simplified behavior development and reuse with potential to accelerate creation of complex mission-focused swarm behaviors for military application.

1.1.1 The Case for Swarms

Swarming vehicles do not provide benefits and capabilities that will completely replace the current unmanned aerial systems (UAS), but they do have the potential for subsuming many current capabilities. More interesting, however, is a swarm's potential to force a technological shift in the conduct of warfare. In [1] Arquilla presents swarming as form of warfare founded on small units of highly connected and maneuverable capabilities that can quickly disperse and aggregate in clusters. Scharre [2] and Hurst [3] expound on the concept of swarm warfare with analysis of robotic swarm attributes and their potential impact on modern battlefield. When compared with current UAS, swarms differentiate themselves via vehicle heterogeneity that provides a wide range of capabilities not available in single UAV. In particular, the speed of decision making and execution that is enabled by cooperative behavior and decentralized control, possibly augmented with artificial intelligence (AI), give swarm systems potentially decisive operational advantages. Or as noted in [2], "having the most intelligent algorithms may be more important than having the best hardware."

The DOD has recognized the possibilities inherent in swarming technologies and lists swarm behavior as one of the AI technologies with "massive potential to advance unmanned systems" [6]. The recognition is further manifested in Defense Advanced Research Projects Agency (DARPA) programs such as OFFensive Swarm-Enabled Tactics (OFFSET) [7]. The service components are individually exploring swarm technologies as well [8] with successful proof of concept demonstrations including the Office of Naval Research (ONR) Low Cost UAV Swarming Technology (LOCUST) [9], the Naval Air Systems Command (NAVAIR) Perdix system [10], and NAVAIR's acquisition of DARPA's CODE program [11]. These programs explore not only the development of autonomous cooperative systems but the C2 and human-system interface (HSI) requirements associated with these systems as well. Both swarm C2 and HSI have been identified as being at the crux of recommended research to bring the technology into the reach of military applications [2], [6], [8].

1.1.2 Swarm Characteristics

The study of swarm robotics arose from earlier research in the area of cellular automata. A cellular automaton can generally be described as a mathematical model for a group of cells in which individual cell states are determined by some function of their neighbors' states over time [12]. The utility of a cell by itself is limited, but a group of cells can effectively simulate natural and biological patterns, and some cellular automata have been shown to be able to simulate any computational machine [12].

Early work by Beni [13] applied the concept of the automaton to robotics. His work with Wang [14] coined the term "swarm intelligence" as "systems of non-intelligent robots exhibiting collectively intelligent behavior evident in the ability to unpredictably produce specific ordered patterns of matter in the external environment." The definition was later refined, to state that an intelligent swarm is "a group of non-intelligent robots capable of universal material computation" [15]. This collective intelligence concept is a basic property of swarms. For this research the notion can be simplified to state that a swarm is a collection of UAVs that are individually capable only of simple behaviors, but that are capable of producing specific additional and more complex behaviors when aggregated into a collective system.

Ensuring that the collective behaviors in a swarm produce the ultimate intended behavior is a foundational concept in swarm engineering [16]. Swarm behaviors often rely on emergence which can be defined as the realization of a system-wide behavior that is produced from the collective behaviors of individual agents [17]. While emergence is a desirable and fundamental characteristic for swarms, it is not easily predicted. Unexpected emergent behaviors can manifest with potential negative consequences that degrade trust in the system [16], [17]. Emergent behaviors are studied extensively not only in regards to swarm intelligence, but also more generally in the context of multi-agent systems covering a broad range of applications including economics, logistics, and engineering. As such there is a large base of behavioral problems with emergent algorithm solutions that have become fundamental in the field. Directly applicable to the field of robotic swarming is the collection of biologically-based behaviors such as flocking [18], ant and bee colony optimization [19], and particle swarm optimization [20], [21]. In each of these problems the individual agent, a single UAV in this research, decides its own best action based on local knowledge and limited knowledge of the rest of the swarm.

The decentralized control and collective behaviors enable the key swarm attributes commonly cited in literature: adaptability (flexibility), robustness, and scalability [22]. Precise definitions of these attributes as they apply to swarm robotics and swarm intelligence are provided in [22]–[24]. In general, adaptability is the result of emergent behavior and the swarm's ability to achieve the range of tasks in a dynamic environment. Robustness stems from decentralized control in that individual swarm agents can still make an appropriate decision regardless of other agents' failures; that is, the swarm can still collectively complete the behavior even if individual agents fail. Scalability is similar in that the size of the swarm should be adjustable as required to complete particular objectives within constraints.

1.1.3 Robotic Swarms

Practical implementations of swarms have progressed quickly over the last decade with the proliferation of low-cost robotics and communication components. Open source robots such as the Kilobot [25] are readily available and more advanced platforms like the Naval Postgraduate School (NPS) ARSENL's Zephyr II fixed wing and Mosquito Hawk quadcopter UAVs can be easily fabricated [26]. Simulation environments like Autonomous Robots Go Swarming (ARGoS) [27], Open Robotics' Gazebo simulator [28], and ArduPilot's software-in-the-loop (SITL) environment [29] are freely available for testing behaviors in conjunction with physical systems to speed up the pace of development.

C2 systems for physical robotic swarms are not as well developed as the simulators, and there are few overarching frameworks for managing swarms. Notable frameworks include the Aerostack [30] for UAVs, and ARSENL's Mission-based Architecture for Swarm Composability (MASC) [4] based framework. Additionally, C2 requirements are closely linked to the study of HSI for swarms. Swarms present unique challenges to human interaction given the potential size of swarms and the complexity of behaviors relative to a person's cognitive abilities [31]. HSI is of particular interest for military applications due to the emergent nature of collective behaviors paired with the rigid operational control structures inherent in military environments [32]. The use of unmanned vehicles in offensive maneuver is already a source of ethical debate, and the autonomous nature of swarms can only compound that issue [33].

1.1.4 Current State of the Art

Significant current research efforts into multi-robot system and multi-robot task allocation (MRTA) are focused on enabling planning and execution of complex behaviors in robotic swarms. Multi-robot systems often rely on task allocation techniques and high-level planning to determine individual vehicle actions required to achieve global swarm objectives. Tasked robots may utilize swarm intelligence and emergence to achieve subtask objectives, but more deliberative approaches are also possible. Khaldi [23] and Arnold, et al. [34] provide a comparison and analysis of multi-robot systems and swarm robotics with an emphasis on the application of swarm intelligence.

Advances in MRTA are enabling increasingly complex task domains through aggregate capability matching that accounts for time and task precedence constraints [35], [36]. Previous research by the NPS's ARSENL group, for instance, has advanced the development of decentralized, market-based task allocation [5], [37] that has been successfully demonstrated in complex multi-domain swarm operations [26]. ARSENL has successfully employed large swarms to cooperatively execute well-defined complex tasks. Effective control of these systems still requires real-time operator oversight.

Recent papers like [38] apply machine learning and AI techniques to swarm systems to enable behavior development. There are relatively few works in this realm and goal of achieving advanced swarm autonomy that operates by "providing commander's intent and the system is able to figure out from that commander's intent what the system is able to do" [11] is still quite distant.

1.2 Research Objectives

The objective of this research is to implement and evaluate hierarchical MASC-based solutions for combining distinct plays capable of autonomous search and investigate tasks into more robust tactics for execution on a heterogeneous swarm. Market-based task allocation is adapted to assign behavior roles to vehicles participating in the tactics. This thesis hypothesizes that composing simple behaviors in this manner can achieve comparable performance characteristics to more monolithic behaviors and that the approach is broadly applicable to the creation of mission-oriented tactics in general. This objective provides a step towards the MASC goals of promoting simple behavior design and reuse and creation of increasingly capable tactics for mission application.

Analysis of developed solutions is conducted to validate the use of composite task-allocation methods and provides a basis for recommendations for future implementation and research into heterogeneous multi-UAV swarm performance and C2. In particular, this thesis addresses the following research questions.

- Does a swarm operating with composed behaviors have comparable performance to a swarm utilizing monolithic behaviors?
- What, if any, benefits with regards to effectiveness and usability do composable behaviors provide over more monolithic versions?
- What measures of performance are appropriate for comparing behavior implementations?
- How can the suggested behavior development approach be extended to support development of behaviors applicable to arbitrary tasks?
- Do the developed methods provide the required flexibility and interfaces to be included in a larger mission control framework?

The scope of this research was limited to the utilization of primitive plays and algorithms to compose robust tactics. It did not explore or develop algorithms for execution of behaviors on individual vehicles. Similarly, it was not the intent of this work to research alternative algorithms for decentralized task allocation within a cooperative swarm.

1.2.1 Methodology

Three swarm behaviors conforming to the idea of a MASC *tactic* composed of more primitive *play*s as described in [4] were developed. The tactics are implemented as compositions of existing ARSENL plays. Each tactic is comprised of a search play that directs a vehicle's participation in a coordinated area search and an investigate play that directs a vehicle in the coordinated investigation of one or more contacts of interest. Both plays utilize previously developed auction algorithms for task allocation [37]. The tactics dynamically assign each vehicle to one of the plays, and only the assigned play is used to control of the vehicle at any given time.

The tactics were developed for use with heterogeneous swarms composed of vehicles with unique characteristics affecting their suitability for execution of the search and investigate behaviors. The tactics utilize market-based approaches (i.e., auction algorithms) to account for individual vehicle capabilities and are described as follows:

- `SearchTacticStatic`. A tactic in which the searcher and investigator roles are statically assigned to specific vehicles when the behavior is initialized. Assignments provide for a minimum number of searchers and effectively prioritize assignments by aircraft type (e.g., faster fixed-wing UAVs assigned to be searchers).
- `SearchTacticDynamic`. With this tactic, all vehicles start off in a `search` role but can dynamically switch between searcher and investigator roles as the behavior progresses. A single item auction is used to reassign roles as contacts are encountered. Vehicles for whom a role change is required delay execution of the transitions to the new roles until after the currently assigned task is completed.
- `SearchTacticImmediate`. This tactic implements the same assignment method as the `SearchTacticDynamic` tactic; however, transitions between searcher and investigator roles occur immediately rather than upon completion of a currently assigned task. That is, an in-progress search cell or investigation task will be aborted if a vehicle is required to switch roles.

1.3 Thesis Organization

This thesis is organized into five chapters. Chapter 1 discusses the current state of aerial swarm systems, their relevance to the DOD, and motivation for this research. Chapter 2 provides more detailed discussion of relevant swarm research areas and their relationships to this research. Chapter 3 describes the implementation of the composed behaviors and compares them to previously implemented monolithic behaviors. Chapter 4 describes the experimentation process that was utilized and discusses the data collected to provide a comparison between the implemented tactics performance and the theoretical optimal performance. Finally, Chapter 5 provides the conclusions of this work and suggestions for future work in this area.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Literature Review

In this chapter we discuss relevant research upon which this research builds including multi-UAV C2 frameworks, behavior composition, MRTA, and auction algorithms. Additionally, we present the MASC framework and ARSENL libraries in the context of their use in implementing and evaluating the described behaviors.

2.1 Frameworks for Multi-Robot Command and Control

Well-defined frameworks for modeling and engineering swarm systems assist in the design and operation of these complex systems by facilitating the description of relationships that connect high-level goals to the intricate details of software and electronic design. In swarms, complex group behaviors are often realized by interactions between individual agents executing simple autonomous behaviors. Ensuring that the correct behavior is achieved and that unwanted emergent behaviors do not manifest is of paramount importance if system trust is to be achieved [16]. In addition, characteristics of flexibility, robustness, and scalability that can make the swarm usable across a range of operations present additional considerations for the system designers and operators. These various considerations are influenced by a range of disciplines and do not have a single comprehensive solution [39]. Considering the range of requirements, a successful framework will help bridge the gaps between swarm behavior developers and engineers and swarm operators. Ultimately, this will better enable the system to achieve the desired results.

2.1.1 Hierarchical Multi-Robot Frameworks

Brooks' subsumption architecture [40], a hierarchical, task-based control system, was proposed in 1986 as an alternative to control approaches relying on traditional AI. The subsumption architecture and its immediate successors were intended for single-robot systems, but the approach provides advantages such as eliminating the need for central control that are directly applicable to swarm systems as well. Brooks described this approach as "levels of competence" [40] aligned to a bottom-up design process where higher levels of competence were added as needed.

More recently, researchers with the Oak Ridge National Laboratory [41] proposed the ALLIANCE architecture for use with heterogeneous robot teams. The proposal outlines many of the considerations and obstacles associated with the application of a control framework to heterogeneous multi-robot systems. Of note, it pays significant attention to the decomposition and distribution of problems across the system to enable decentralized autonomy and achieve group goals in a fault tolerant and reliable fashion. The ALLIANCE architecture achieves this robustness through a layered approach where agents maintain behavior sets from which appropriate behaviors are selected based on the locally perceived current situation. Limitations were noted for overall coordination of subtasks, however [41].

Both the subsumption and ALLIANCE architectures approach behavior development in a bottom-up fashion. That is, simple primitives are iteratively combined into increasingly complex behaviors. McLurkin [42], on the other hand, suggested that a top-down approach to designing multi-robot swarms was needed while also noting the difficulty of such an approach. McClurkin's observations acknowledge the problem of attempting to program a large number of systems to achieve a group objective by programming individual robots. In the end, the paper proposes a bottom-up, composable solution that attempts to "break group behaviors into smaller behaviors that can be combined to achieve a larger goal" [42] with an emphasis on designing to achieve predictable swarm behavior.

In the same year, Miller, et al. [43] proposed the use of a "playbook" as a human interface layer above the robot control system. Playbooks are based on the familiar concept of sports plays in which each participating agent has a specific role contributing to a larger objective. The concept was specifically applied to a military setting and sought to "enable a small unit soldier, with minimal training, to control or request services from multiple, heterogeneous UAVs" [43]. This top-down method is supported by an intermediate supervisory layer that decomposes the requested behavior into sequential sub-tasks for assignment via a hierarchical task network planning system. It was noted that the system supported the planning goal, but that the developed plans were not necessarily optimal [43]. The playbook concept forms the basis of the MASC approach described in Section 2.1.3.

Browning, et al. [44] extended the playbook model with a hierarchical planning method called Skills, Tactics, and Plays (STP) for multi-robot systems. STP defines a more structured approach to playbooks with a three-level hierarchy for selecting and executing behaviors. Plays are defined as group behaviors that can be selected from a playbook. Once selected, plays delegate roles, or behaviors, to individual robots. Each role is executed through a series of tactics to collectively achieve the play objectives. At the lowest level, each tactic is performed by the execution of skills according to tactic-specific state machines. This top-down approach achieves the usability goals for control and operation within a robot team, however development of the skills and appropriate state machines was challenging. Development of new tactics and plays was non-trivial [44].

More recent frameworks have sought to increase the levels of autonomy and the reusability of components across behaviors and vehicle types. The Aerostack framework, for instance provides for a layered approach in which each layer comprised of distinct systems associated with specific functional tasks [30]. Each system is based on composable processes with a standard interface that facilitates messaging between components using open source software. Aerostack operators direct the swarm by initiating high-level behaviors that are implemented through task trees that utilize specific robot skills. This framework achieves a high level of autonomy and ease of operator interaction but comes with the trade off of high system complexity.

2.1.2 Composition

The frameworks discussed in Section 2.1.1 leverage composition in both modeling and programming of behaviors to achieve abstraction through layering. That is, they use a hierarchical approach to distance high-level behaviors from the low-level primitives upon which they rely. With composition, systems are represented as structured aggregations of, and interactions between, subsystems or components. Composition is a foundational concept of object-oriented programming and Component-Based Development (CBD), and its study is prevalent in the swarm-related disciplines of agent-based modeling, multi-agent systems, and complex adaptive systems.

Davis and Anderson [45] cite model composability within the DOD as a requirement for component reuse and modeling of complex systems. However, following an analysis of prior composition efforts they assert that composition in systems modeling is more difficult than general software composition. Additionally they state that the idea of plug-and-play mechanics for composition is a fallacy in either domain and that successful composition inevitably requires specific efforts for adaptation. They do submit that continued study will improve composition adaptation and implementation methods.

While software development and system modeling are different disciplines, Bartholet, et al. [46] argue that the study of composition across the two are fundamentally similar and that application software techniques will benefit modeling efforts. The argument compares two general types of composition, syntactic and semantic, and identifies commonalities between them. Syntactic composition is attributed to software development with focus on implementations of exposed interfaces and data types within domains in order to build complex systems. Semantic composition, on the other hand, focuses on the validity of compositions and the resulting systems, which is relevant to modeling. [46] highlights the convergence of semantic results within both CBD and modeling and simulation and notes that the existing semantic challenges are shared between the two. This provides a basis for considering composability for both the system behavior modeling perspective and the software design perspective.

Lau and Rana [47] take a CBD-based approach in providing a survey of composition methods. They describe categorical view of composition that is comparable to [46] in that the highest level of modeling, the *CBD View*, specifies that component units conform to component models with distinctions similar to semantic modeling in [46]. In line with this thesis, they define the basic component of composition as a behavior and that "composition mechanisms compose pieces of behavior into larger pieces of behavior" [47]. The mechanisms: *containment*, *connection*, *extension*, and *coordination* define the basic composite relationships upon which systems are built. Of interest to this thesis, the defined coordination model is described as coordinating multiple behaviors such that the behaviors communicate only with the controller via defined connector or via exogenous composition. The behaviors composed under a controller are fully decoupled from each other, meaning that they do not have any interaction or dependencies directly with each other [47].

The coordination mechanism was discussed in a previous work by Lau and Ornaghi [48] that defined exogenous composition. In this work, exogenous composition is described as behavior encapsulation to support control via hierarchy. However, composing behaviors in this manner requires the behaviors and resulting composition be of the same type, without external dependencies. To maintain the hierarchical control, behaviors used in exogenous composition must not call or pass control to external behaviors. Control is passed only through the composition [48]. This is comparable to the `plugin_behavior` construct in the ARSENL software library.

Zhu, et al. [49] provide an examination of complex adaptive systems modeled as agent-based systems. In their approach, a three-level hierarchy comprised of component models (behaviors), agent models, and a system model is used to represent a multi-agent system. The hierarchy is comparable to the coordination mechanism described in [47], with the system model acting as the connector and the agents as the decoupled behaviors. However Zhu, et al.'s model control is not solely passed through the composition method as agents react to events that are distributed within their hierarchy [49]. Notably, Zhu, et al. extend the requirement for decoupling to the agents' sub-components in that an agent also acts as coordination mechanism for all of its systems. This composition model provides a complimentary perspective to system design for multi-agent systems when compared to non-agent based hierarchies that emphasize low-level interfaces and varying degrees of component coupling.

Weisel [50] goes so far as to provide a formal mathematical theory for determining the validity of composability, specifically for semantic, or model-based, composability. Validity in this case means that the model matches, or closely matches, the system it was designed to represent at all points of simulation. In this context, a model is formally defined as a computable function mapping from a state model to an output model. Weisel then asserts that "because models have been defined as computable functions, composition of models becomes composition of functions [and that because] the set of computable functions is closed under composition, any set of models can be composed if the composition exists" [50]. Unfortunately, this notion of composition does not guarantee that the resultant composed behavior will be valid in terms of the otherwise valid model.

Finally, Sarjoughian, et al. [51] examine composability through the lens of a multi-layer modeling concept founded on formal mathematical and algorithmic representations of distinct systems and present optimization or process models as an example. Composition methods are then classified into mono, super, meta, and poly methods that provide structure for composing heterogeneous formalisms. Composition methods are described in conjunction with the use of Knowledge Interchange Brokers, and adapters required for model interactions with emphasis on the role of domain knowledge. This analysis applies to the structure of frameworks which are multi-level and compose heterogeneous models. Additionally, the authors' observation that "all of the composability approaches lend themselves to distributed execution," [51] makes their findings potentially applicable to a swarm operation.

The works presented in this subsection are a sample of the broad study of composition within the considered domains. While none were swarm focused, the discussions related to composition methods within hierarchical systems and multi-agent systems are closely related and applicable to the objective of designing and composing behaviors and supporting algorithms.

2.1.3 Mission-based Architecture for Swarm Composability (MASC)

The work presented in this thesis relies heavily on the MASC framework [4] as utilized by the ARSENL multi-UAV system. Here we present an overview of MASC and describe its relationship to the ARSENL system.

The impetus behind the creation of the MASC framework was the intersection between swarm robotics with military C2 and the desire for a system design that enables construction of mission packages through a hierarchy of reusable modular, composable behaviors [4]. This objective is influenced by design constraints that were introduced in Chapter 1 and Section 2.1 such as mission planning processes and requirements, adherence to doctrine, system trust, and human-swarm interaction. These considerations were not a focus of the work of this thesis but are mentioned here to frame the goals of the tactics described in Chapter 3. Ultimately this design promotes the usability of robotic swarms through all phases of mission planning and execution.

The MASC hierarchy is comprised of five levels that represent behaviors with different granularities: missions, phases, tactics, plays, and algorithms [4]. This conceptualization differs from other approaches but is founded on the work of other multi-robot frameworks discussed in Section 2.1.1. Elements at each level are composed of one or more sub-level objects that in combination achieve higher-level element's goals. A brief description of each level follows; however this research focuses on composability of MRTA functions limited to the tactic and play levels.

Mission Mission definitions are the top level of the hierarchy and describe the overall objective. The mission associated with this thesis, for instance, is the conduct of a search of a predefined area and the investigation of any contacts of interest.

Phase A mission is comprised of seven sequential, temporal phases: *staging*, *mission planning*, *preflight*, *ingress*, *on station*, *egress*, and *postflight*. These phases encompass the full scope of mission planning and execution and largely align with the phases associated with an ARSENL event [52]. The work of this thesis is limited to actions within the on station phase.

Tactics A tactic is behavior that "commands the ordered formation and employment of individual agents to perform a specific task as a cooperative group" [4]. In short, a tactic applies the high-level logic necessary to achieve a single desired objective. Tactics associated with this research, for instance, direct the swarm as it searches the area of interest and responds to contacts as they are identified.

Plays Plays define the cooperative behaviors that perform a single well defined and reusable action that is potentially useful in multiple tactics. Plays are the discrete activities that when combined effect the aggregate actions to achieve a tactical objective. This research relies on a search play that divides a large area into smaller cells that are searched by swarm members and an investigation play that directs a chosen vehicle to a contact of interest.

Algorithms Algorithms make up the lowest level of the hierarchy. They consist of the functions that provide common control sequences and that assist in geospatial awareness, motion planning and positioning, robotic control functions, communications, and task allocation. Examples of algorithms upon which this work relies include path planning and transit to a contact of interest and pattern execution over a search cell.

The MASC framework provides a single approach that balances the top-down and bottom-up design methods utilized in mission planning and swarm engineering. Mission planners can think in terms of mission objectives and their decomposition into tactics while developers think in terms of algorithms and plays that can be combined into tactics. The design method also provides for modular, composable, and reusable components that can enable the full spectrum of mission planning and operations [4].

2.2 Individual Task Assignment within the Swarm

An implicit aspect of the MASC framework is the notion that discrete elements of the swarm may execute different tactics or plays simultaneously in support of higher-level goals. That is, a tactic may call for heterogeneous plays to be assigned to subswarms. This necessitates logic for swarm division and subtask assignment. This process is referred to as MRTA, a research area focused on the assignment of goals or tasks to specific agents or groups of agents. This deliberative, task-specific organization is distinct from swarm behaviors such as flocking that rely on emergence associated with local agent interactions [53]. This section reviews MRTA concepts with an emphasis on auction-based approaches utilized in the plays described in Section 2.3.2.

2.2.1 Multi-Robot Task Allocation (MRTA)

An overview and taxonomy of MRTA is provided by Gerkey and Matarić [54]. This taxonomy describes MRTA as an optimization problem that maximizes benefit or minimizes cost. As such, utility is the basic attribute required for making optimization decisions, and each robot estimates its own utility for a task based on its anticipated cost to perform the task and its suitability for completing the task.

Gerkey and Matarić [54] propose classification of a problem set into robot-task-assignment categories that can be collectively examined for application of optimization techniques. Classifications are binary with robots classified as able to perform a single task (ST) or multiple tasks simultaneously (MT), tasks classified as to whether they require one robot (SR) or multiple robots (MR) to complete, and assignments classified as to whether they require immediate assignment with no future planning (IA) or information is available to support assignment or scheduling in the future (TA). Thus, a problem's classification is represented by a tuple such as ST-SR-IA. This particular classification would describe a multi-robot problem in which each robot is capable of one task, where each task is to be completed by a single robot, and tasks are to be allocated immediately without planning for the future.

Gerkey and Matarić's taxonomy [54] does not directly address creation of subswarms as would be used in the control of a tactic but does assess the ST-MR-IA classification as being equivalent to *swarm coalition formation*. Swarm coalitions are formed when the task requires capabilities of more than one robot, such as the need for different sensors or physical domain actions. Coalition formation for a multi-agent system is discussed in [55] in which the authors also note the improved efficiency of groups relative to single agents when completing tasks. Agents in [55] are characterized by a capability vector while independent tasks have specific capability vector requirements. Coalitions are formed such that the union of the group's capability vectors satisfies the required capability vector of the tasks assigned to the group. The optimization problem then becomes a maximization of aggregate coalition utility in addition to agent utility.

Korsah et al. [53] expand Gerkey and Matarić's taxonomy to include consideration of task decomposition and constraints for task dependencies such as sequencing requirements and intertask dependencies. Additionally the concept of synergy and group utility is presented as an additional measure for coalition comparison. Task relationships and dependencies amount to an additional layer applied to the tuple representation proposed in [54] to represent effects on task utility values imposed by the constraints. The proposed categories are *no task dependencies*, *in-schedule dependencies* (i.e., dependencies between tasks to which a single agent has been assigned), *cross-schedule dependencies* (i.e., dependencies on another robot's task), and *complex dependencies* (i.e., dependencies between complex tasks). The definitions associated with task composition are drawn from [56] which defines *elemental tasks*, *simple tasks*, *compound tasks*, and *complex tasks*. These definitions can be used to describe relationships between decomposed subtasks and robot allocations for the development of task dependency categories.

Implementations can further be classified according to swarm organization and information flow. Organizationally, the *centralized paradigm* appoints one robot to receive, process, and disseminate tasking information to the swarm. In the *decentralized paradigm* each agent is responsible for information processing, determination of its tasking, and communicating with other swarm members when necessary [36].

Some swarm behaviors may require global information to be available to individual agents while others may rely solely on local knowledge. Johnson, et al. [57] provide examples relating organizational paradigms, information assumptions, and their effects on multi-robot systems beyond the pure optimization problem. In particular, even the most basic scenario requires some level of on-line communication and planning, and this requirement imposes computational and time constraints. Additionally, when the communications environment is poor, global information is difficult to synchronize, and both centralized and decentralized planning efforts suffer as a result.

The taxonomical conceptualization of MRTA described in this section provides a basis for discussion of MRTA implementations. The next subsection presents a brief overview of MRTA optimizations and explanation of the market-based techniques including the ARSENL auction algorithm utilized in the plays and tactics described in Chapter 3.

2.2.2 Optimization in Swarm Task Allocation

MRTA is an optimization problem as stated in Section 2.2.1 and many techniques have been applied to the problem in general to fit the various taxonomies and organizational paradigms. Optimization in this respect is studied under combinatorial optimization with emphasis on linear programming techniques, namely the assignment problem which has linear programming solutions [54], [58]. Non-linear techniques incorporating heuristic or relaxation techniques are also applicable for distributed solutions and swarm taxonomies, such as coalition forming, that are more difficult and require solutions that balance computation and time with optimization [35], [54], [55], [59], [60].

In practical application, [54] clarifies the definition of optimization beyond the pure mathematical sense that given “union of all information available in the system it is impossible to construct a solution with higher overall utility” which accounts for the environment and information factors discussed in 2.2.1. Viguria and Howard [61] emphasized this concept with a probabilistic analysis noting that “In most of the cases, the quality of the solution (closeness to the optimal solution) depends on the information accessibility.”

Within the optimization problem set, the most basic solution is the greedy heuristic approach, which selects the best choice at the time based on current knowledge. This approach is suboptimal in most cases and measured as a competitive ratio defining the minimum expected performance compared to the optimal [54]. The greedy approach is ubiquitous throughout the literature and provides a useful comparison when assessing trade-offs in more complex heuristic approaches.

2.2.3 Auction-based Task Allocation

Market-based approaches have become prolific throughout MRTA implementations. The application of market-based approaches is credited to Smith [62] and the contract net protocol which assumes that a distributed agent structure with no *a priori* or global knowledge of tasks can achieve a global objective. Agents are dual purposed as managers who bid for the tasks and monitor results, and contractors who execute or decompose and subcontract tasks. Dias, et al. [56] provide a formal definition of these aspects in that the global objective is decomposable, that a global objective function exists defining the desired solution, and that individual objective functions manage contribution to the global with respect to individual preferences and constraints.

Auctions are a common implementation of market-based MRTA. There are many forms of auctions including single-item, multi-item, and combinatorial auctions with both centralized and decentralized implementations [56]. Single-item auctions are characterized by a single bidder-to-task pairing, whereas in multi-item auctions bidders can place bids on multiple tasks simultaneously [63]. Combinatorial auctions package tasks to be sold as multi-task bundles as opposed to individual tasks. Multi-item and combinatorial auctions present opportunities in the MR, MT, and TA taxonomies, however they come with distinct challenges in implementation and optimization [63], [64]. The remainder of the auction discussion will be limited to single-item auctions that can be applied iteratively to task sets.

Bertsekas [58], [59] provided a method to apply the auction algorithm directly to the assignment problem through a relaxation method incorporating ε-complimentary slackness (ε-CS) to the pricing scheme. In this auction method bids are increased by an amount no less than ε to ensure that the price vectors for available tasks are maintained at a minimum throughout bidding and assignment. Bertsekas shows that when utilizing this auction method that a complete assignment, where each distinct object is assigned a complete task, will occur and that assignment will be optimal when $\varepsilon < 1/N$, where N is the number of agents that can be assigned tasks [59].

Bertsekas' auction algorithm provides an optimal assignment solution in a distributed system with shared global knowledge. However, Stentz [60] highlights obstacles to achieving desired optimality when using market-based algorithms on multi-robot systems that have imperfect global knowledge. Additionally, Viguria [61] provides analysis of market-based global cost divergence from the expected optimum when all information is not known at the start. Other considerations when implementing auctions include defining appropriate utility functions and making accurate cost estimates and task decomposition schemes which may "not consider the complete solution space and may find highly inefficient solutions" [56].

Hopchak [37] implemented a distributed variation of Bertseka's auction algorithm in the ARSENL software architecture using a consensus mechanism to synchronize global swarm and auction states. The implementation conducts an area search function with a heterogeneous fixed-wing UAV swarm by first decomposing the search area into search cell tasks and auctioning each cell. Campbell [5] expanded this work to include both fixed-wing and quadcopter UAVs and dynamic task allocation for investigation tasks that are added over the course of the area search. In this implementation the utility functions were modified to introduce a preference modifier for pairing task types to specific vehicle capability.

2.2.4 Auction Coalition Implementations

There are numerous works implementing auction-based approaches to the MRTA problem in both ST-SR-IA and swarm coalition applications. In swarm coalitions groups of robots are assigned to a task to capitalize on synergy of aggregated robot capabilities. This is analogous to the creation of subswarms for assignment to roles within a hierarchy.

Stentz and Dias [60], for instance, propose a market-based approach to coalition formation and introduce several coalition concepts. One is that agents bid not only on tasks but also on the services. That is, they bid on both preferred tasks that they will complete and services that they want other agents to provide. Vig and Adams [64], on the other hand, propose using a combinatorial auction framework in which the tasks are the bidding agents competing for robots in order to form coalitions with the required resources.

Liu and Shell [65] propose manipulation of a distributed swarm utility matrix to create a task-agent grouping, or hypergraph, representation of subswarms. This is accomplished through *coarsening*, or dropping low utility values, and then conducting matrix partitioning to identify groups. Subswarms can assign tasks via decentralized solutions like auctions, or they can recursively divide the tasks into new hypergraphs. This is a comparatively simple implementation for subswarms; however, it is not fully decentralized as the updates to the utility matrix upon which it relies are conducted through lead and sub-lead robots.

Further, dynamic updates are only allowed to modify existing tasks and not to introduce new tasks [65].

Guerrero and Oliver [66] utilize a double round auction in which work capacity and group capacity are introduced to provide a mechanism for incorporating time constraints. A selected leader first solicits bids for capacity from available robots and then calculates the necessary composition in a greedy manner to meet constraints. A second auction is then conducted in which the leader presents bids to the selected robots to form the coalition. Ifran and Farooq [67] took a similar approach that incorporates task workload and robot work capacity. In their proposal, leaders are first elected via auction, then the coalitions are formed via auction so that coalition capacity satisfies workload requirements. Their approach also introduces intra-coalition swapping to allow for exchanging robots between coalitions when work capacity is not met or when there are dynamic changes to the work environment. This exchange is accomplished via auction by coalition leaders and presented results demonstrate improved matching of coalition capacity to workload requirements [67].

In [68], de Mendonça and Nedjah propose a local dynamic task allocation method that ensures a desired global proportion of homogeneous robots are assigned to available tasks. Robots maintain localized task assignment tables for the swarm robots and known tasks. If there are more robots assigned to a task than the desired proportion dictates, then that task has over-allocated resources. The robots determine transfer options locally to balance the proportion of robots allocated to a task. Transfer decisions rely on a combination of known differences in global proportion requirements and robot id to avoid over-correction. This simple localized technique was shown to provide fast convergence to desired task allocation that lends itself to easy comparisons in a hierarchical organization [68].

Kose, et al. [69] expanded on previous work that assigned roles to members of a robot soccer team using market-based mechanisms. Their extension allows dynamic, multi-role assignments after the initial assignment. Assignment is supported by reinforcement learning mechanisms and a generalization of the role definitions. The reinforcement learning utilizes a state vector representation of the perceived game state in addition to the auction costs to dynamically determine the set of robots best suited for the attack and defend roles.

These MRTA implementations provide background and fundamental works in the domain; however, none are completely analogous to the hierarchical task decomposition of a framework like MASC. While this thesis does not require the task and capability planning mechanisms of advanced coalition formations, the segmentation schemes provide insight to techniques that can be adapted to the control of tactics and subswarms in the MASC framework. In particular, at the tactic level, decentralized single-item auctions are utilized to form subswarms to which specific plays are assigned.

2.3 The Advanced Robotic Systems Engineering Laboratory (ARSENL)

This research was conducted on the NPS ARSENL UAV swarm system and utilized their behavior library based on the MASC framework [26]. In this work, the search and investigation components are implemented as MASC *plays* that are combined into *tactics* as described in Chapter 1. We use the term *role* to describe the objective of the play assigned to an individual swarm agent at a particular time and the term *behavior* to refer to the swarm (or subswarm) objective more broadly.

The NPS ARSENL conducts research and concept development for robotic unmanned systems. Its history of innovation in swarm development is highlighted by events such as the 50 fixed-wing UAV demonstration in 2015 [70] and complex multi-domain field experimentation in 2021 [26]. ARSENL swarm vehicles operate using common open architecture and open-source hardware and software components. The swarm is comprised of heterogeneous, multi-domain vehicles, including fixed-wing aircraft, quadcopters, and ground vehicles [26]. ARSENL experiments are conducted in both live-fly events at Camp Roberts California and in a robust SITL simulation environment [71].

2.3.1 ARSENL Software

The ARSENL on-UAV software implements the MASC framework for mission planning and operations. Figure 2.1 depicts the state sequence for a mission that loosely corresponds to the MASC phase level, with the `Swarm Ready` state representing the On Station MASC phase. This is the state in which the mission tactics and plays are executed.

In the ARSENL software architecture, the tactic and play implementations are contained

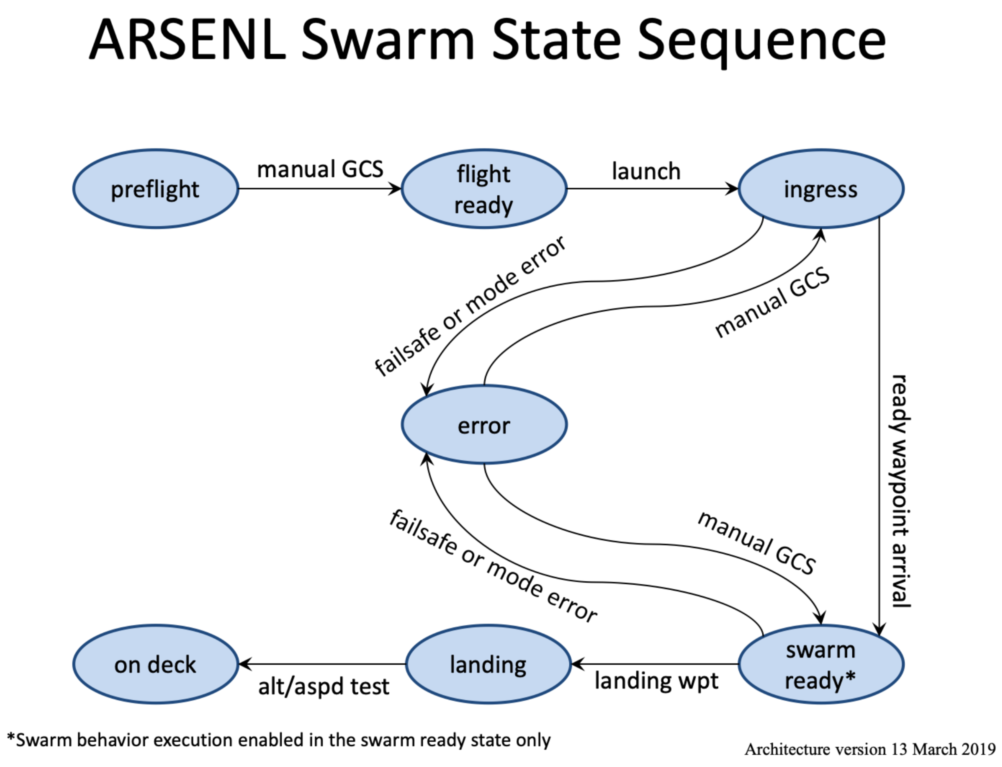

Figure 2.1. ARSENL Swarm State Sequence. Adapted from: [52]

in their own libraries: `arsenl_tactics` and `arsenl_plays`. Tactics and plays both implement the abstract `plugin_behavior` which provides the general interface for swarm tasking and control.

The majority of behaviors implemented in the ARSENL library exist as plays. These include behaviors such as `DirectTransit`, `SequencedLanding`, `SwarmSearch`, `Contact Investigation`, and `CuedSearch`. Each of these plays is a self-contained behavior that relies on one or more algorithms. Each can be executed in isolation or incorporated into a tactic. A number of these plays might be better implemented as tactics since they implicitly incorporate the functionality of other plays. The `cued_search` play, for instance both searches a geographic area and investigates contacts of interest.

The MASC algorithm level is represented by several supporting libraries. Notably, algorithms that support characteristics described in Section 1.1.2 such as path planning, transit, collision avoidance, as well as biologically inspired algorithms like flocking are provided to support the basic swarm maneuvering. Additionally, utilities supporting specific tasks such as decomposing a geographic area into cells are available at this level. The ARSENL algorithms library also implements a number of task allocation strategies including single item auctions.

2.3.2 ARSENL Goals

As noted, the `cued_search` play highlights a case of reproduction of code that is not aligned with the reusability and composability goals of MASC. The work in this thesis reproduces and extends the `CuedSearch` functionality in the form of the tactics described in Section 1.2.1. These tactics utilize compositions of the `SwarmSearch` and `ContactInvestigation` plays in line with the intent of the MASC framework.

While outwardly simple, this decomposition is complicated by the need to implement the task allocation mechanism to support the hierarchical architecture. That is, it is not enough to simply build a tactic as a consolidation of plays. Assignment of roles (i.e., plays) and parameterization of those plays must also be implemented within the tactic.

The primary goal of this research is the implementation of tactics performing complex *search and investigate* behaviors that are created through composition of primitive search and investigate plays in a hierarchical C2 framework. The tactics utilize a market-based approach to dynamically assign the primitive behavioral roles to participating swarm members. Live demonstration and simulation of the composed tactics to similar monolithic behaviors utilizing market-based approaches provides empirical evidence that a composed behavior has similar performance characteristics as the monolithic behavior and provides flexibility for creation of increasingly complex behaviors.

2.4 Conclusion

This chapter provided a literature review that highlighted many of the considerations associated with achieving the goals of this research. Namely, considerations of global utility, individual utilities, organizational paradigm, and trade-offs with the allocation methods and optimality were shown to be of importance. Chapter 3 will describe the application of these concepts, most importantly MASC and MRTA, to the development of specific search-and-investigate tactics for the ARSENL swarm system.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Implementation

This chapter details the development of the three tactics described in Section 1.2.1 using the ARSENL codebase. Each tactic combines existing ARSENL plays into one ARSENL tactic. Design choices related to composition and models discussed in Chapter 2 are highlighted. The first part of the chapter describes the mechanics of each play and provides a description of the `CuedSearch` play that is used as the comparison baseline. Options for the tactic implementation are then discussed. The remainder of the chapter describes the three implemented tactics themselves and focuses on their state machines and auction metrics. Finally, specific barriers to implementation are discussed with respect to creating composable behaviors with multiple auctions.

3.1 Existing Advanced Robotic Systems Engineering Laboratory (ARSENL) Functionality and Behaviors

The existing plays in the ARSENL library used for composition in this experiment were `SwarmSearch` and `ContactInvestigation`. Each play relies on algorithms and utilities provided by the ARSENL codebase `arsenl_algorithms` and `arsenl_utilities` packages.

3.1.1 Plug-in Methods

All ARSENL behaviors, including plays and tactics, inherit from the `PluginBehavior` class. This class provides member variables and methods that are common across swarm behaviors such as variables containing records for all swarm participants and methods for passing essential information to other swarm participants. The `PluginBehavior` class also contains virtual methods that child classes (i.e., plays and tactics) use to implement behavior-specific functionality. Of particular interest are the `parameterize` method that is used to set specific execution values, the `maneuver_command` method that directs the vehicle as the behavior is executed, and the `process_behavior_data` method that processes messages received from other swarm members.

The `parameterize` method is executed once upon initiation of a behavior. It is used to initialize the behavior's parameters to satisfy a specific set of objectives (e.g., the search of a particular geographic area) as directed by the swarm operator. The initial behavioral state is set prior to the first iteration of the behavior's control loop. With respect to composition and transition between behaviors, parameters may be passed through the composition hierarchy as appropriate by calling contained plays' `parameterize` methods.

The `maneuver_command` method is executed once with each iteration of the UAV's 10 hertz control loop [52]. Its primary purpose is to provide maneuver commands for the UAV; however, supporting algorithms such as checking and updating vehicle and component state values are also typically applied within this method to ensure synchronization among components. With regards to tactic implementation, role assignment and activation of contained plays are the most important tasks that must be completed by this function.

The `process_behavior_data` method is used to process messages received from other swarm participants. Since message receipt is asynchronous, the `process_behavior_data` method is not run within the confines of the maneuver control loop. Rather, it is executed as a callback when messages are received. State changes or actions stemming from messages can be made from within the method, but actions such as role changes that are associated with state changes should be effected during the control loop execution by the `maneuver_command` method. Implementing composed behaviors requires consideration of the appropriate processing and dissemination of messages between the tactic and composed plays. Messages may be passed to the play associated with the tactic's current role (and possibly other included plays) by invocation of its `process_behavior_data` method.

3.1.2 Role and Task Auctions

The tactics and plays in this research utilize single-item auctions for task allocation. The auction algorithms used in this work were adapted from those described in [37] and [5] and are broadly described in Section 2.2.3. This section details the current implementation of the auctions within the ARSENL library and outlines their utilization for both task allocation and task execution in a composition framework. More detailed information on the auction bidding and concurrency mechanisms in support of search and pounce actions are available in [37] and [5] respectively.

Resource Records

At the tactic level, auctions are utilized to assign individual UAVs to roles that are effected by the contained plays. Available role assignments are represented as auction resources for which the vehicles bid. Within the ARSENL auction library, this is implemented through a `ResourceRecord` class [5]. The `ResourceRecord` class contains a reference to the associated object for auction (e.g., a search cell or pounce target) and auction-specific parameters such as the auction ID, the resource ID, current price, and resource status. The resource status within the `ResourceRecord` equates to the current task's state and is used for both conducting an auction and managing tasks in hierarchical composition.

`ResourceRecord`s are contained in a `ResourceSet` class object which is instantiated as a member object of a specific auction. The `ResourceSet` class contains basic accessor and mutator functions for `ResourceRecord`s. These methods are used for all manipulation and querying of individual `ResourceRecord`s during an auction's conduct.

Utility Values

An `ResourceRecord`'s utility value captures the value of the resource to a specific UAV and is a required member for a `ResourceRecord`. It is calculated locally and used in the vehicle's bid determination. Utility values are generated by the `SingleItemAuction` instance using a function that is provided by the supported tactic or play. This allows the behavior to evaluate available resources as required to account for different types of tasks and vehicles.

The utility function implemented by the tactics and plays in this work utilizes the basic formulas described in [5] that incorporate a base value for the task, a multiplicative modifier for each airframe type, and a calculation of the estimated time to complete the task. For each vehicle, i, the utility value for a resource, r is calculated as

$$v_{r_i} = v_r - (m_a * t_r) \tag{3.1}$$

where v_{r_i} is the local utility value, v_r is the task's base value, m_a is the vehicle-specific airframe multiple, and t_r is the estimated time required to complete the task. The estimated time to complete the task is calculated as

$$t_{r_i} + t_{t_r} + t_r \tag{3.2}$$

where t_{r_i} is the estimated time required to complete a currently assigned task, t_{t_r} is the estimated transit time to the location of the resource task, and t_r is the estimated time required to complete the resource task.

The base task value captures the raw importance of a task in terms of its contribution to the behavior's overall objective. In this work the base values were drawn from [5] and set to 250 for `SwarmSearch` tasks and 350 for `ContactInvestigation` tasks in order to slightly prioritize contact investigation over area search.

Fixed-wing and quadcopter airframe multiples of 1.0 and 3.0 respectively were uses for `SwarmSearch` tasks. Conversely, fixed-wing and quadcopter airframe multiples of 3.0 and 1.0 were respectively used for `ContactInvestigation` tasks. This makes the search tasks more costly for quadcopter UAV and pounce tasks more costly for fixed-wing UAV and intentionally biases their respective preferences towards the task type for which they are most suited (i.e., faster fixed-wing UAVs prefer search tasks and hover-capable quadcopters prefer investigations). These multiples are slightly higher than those used by Campbell to account for the transit distance to outweigh airframe suitability when assigning pounce tasks in larger search areas [5].

As described above the utility functions leverage swarm heterogeneity via the airframe multiple to match UAV types to tasks they are best suited for, and relative task importance is captured by the different base values so that more valuable tasks can be prioritized. As shown by [5], implementing this combination of utility metrics in a single-item auction achieves the desired task allocation performance characteristics when utilized by a heterogeneous swarm. An adaptation of Campbell's work is implemented in the ARSENL library as the `CuedSearch` play. This play is used as a baseline for comparison to the tactic implementations described in the remainder of this chapter.

3.1.3 Swarm Search

The `SwarmSearch` play is based on [37] and directs swarm vehicles to complete a full search of a rectangular geographic area. It is initialized with coordinates that define the geographic area, divides the area into rectangular search cells, iteratively allocates cells to individual vehicles via auction, and maneuvers the vehicles to effect the search of their assigned cells. The behavior derives a search pattern for a cell as a series of geographic waypoints that are transited by the UAV. Upon reaching the last waypoint, the search cell task, is marked as complete and a new auction is commenced if there are unassigned cells remaining. The play continues until all cells have been completed.

The geographic search area is represented by a `RectangleSearchArea` object that implements the algorithm for dividing the search area into cells. The algorithm divides the area into equally sized cells of length L_{cell} and width W_{cell} as

$$L_{cell} = \frac{L_{area}}{\left\lceil \frac{L_{area}}{L_{max}} \right\rceil} \tag{3.3}$$

$$W_{cell} = \frac{W_{area}}{\left\lceil \frac{W_{area}}{W_{max}} \right\rceil} \tag{3.4}$$

where L_{area} and W_{area} are the length and width of the search area respectively, and L_{max} and W_{max} are the maximum length and width of each search cell.

The resulting `RectangleSearchCell` objects include a method to generate a search path through the cell based on a vehicle-specific sweep width and search altitude. The generated path is stored in a `WaypointSequencer` object that includes methods to direct the vehicle along the path. The path generation method is not called until the vehicle commences execution of the search task. This allows the derived search pattern to account for the UAV's transit to the cell.

The ARSENL framework provides a number driver classes for controlling vehicle transits. Among these is a `SearchPatternDriver` that is utilized by the `SwarmSearch` behavior to control the movement of the UAV as it completes the search of a single cell. The driver is a member of the `SearchCellResource` object, a subclass the of the `ResourceRecord` described in Section 3.1.2, rather than the search area or search cell object. The `SearchCellResource` objects are generated by the behavior after the creation of the search area and derivation of the search cells and used to initialize a `SingleItemAution` object.

The `SwarmSearch` play's `SingleItemAuction` object is initialized during `parameterize` execution. Each UAV participates in the swarm auction completing assigned tasks as described above until the search of the area is complete. Progression through the auction and execution of maneuver via the active resource's driver are implemented within play's `maneuver_command` method, which is called once for every iteration of the UAV's main control loop.

3.1.4 Contact Investigation

The `ContactInvestigation` play is used to direct the investigation of a contacts (i.e., a "pounce"). The pounce action consists of a direct transit to a reported contact's geographical location and a loiter for a predetermined amount of time. The pounce is considered complete after the UAV has been "on top" of the contact for the specified amount of time. As with `SwarmSearch`, contacts are allocated to swarm members as tasks via auction. Unlike the `SwarmSearch` behavior, `ContactInvestigation` is initiated with an empty list of contacts, and contacts are added as they are identified and reported via contact reports from other swarm members.

The overall design of the `ContactInvestigation` play is less complex than the `SwarmSearch` play since there is no large task (i.e., search area) that must be decomposed into smaller tasks (i.e., search cells). It does, however, have to dynamically add new contacts as they are reported. Individual contacts are represented as `Contact` objects that specify the geographic location and altitude of the contact as well as the time at which it was reported. Each `Contact` object is created when the contact is reported and incorporated into an `InvestigationResource` object, another subclass of the `ResourceRecord` class, that is added to the behavior's `SingleItemAuction` instance. The `InvestigationResource` object contains supporting members and methods, including a `ContactInvestigationDriver` that serves the same purpose as the

SearchCellResource class's SearchPatternDriver.

The code for adding newly reported contacts to the auction is contained in the process_behavior_data method that is invoked upon receipt of behavior-related messages from other UAVs (i.e., contact report messages in this case). The Contact object and associated resource are added to the play's SingleItemAuction. Addition of a new contact also requires immediate initiation of a new auction process or restart of an in-progress auction. The process_behavior_data method effects this by changing an auction_state variable that causes an auction restart upon the next execution of the play's maneuver_command method.

3.1.5 CuedSearch

The CuedSearch play is a monolithic combination of the SwarmSearch and Contact Investigation play functionality based on the work presented in [5]. It utilizes the resource classes discussed in Sections 3.1.3 and 3.1.4 to enable allocation of both SearchCellResource and InvestigationResource objects by the same single-item auction.

The CuedSearch parameterize method instantiates a SingleItemAuction object and initializes it with SearchCellResource objects in the same manner as the SwarmSearch play. The process_behavior_data method functions exactly as the ContactInvestigation version and dynamically adds InvestigationResource objects to the auction as new contacts are reported.

The methods controlling maneuver state, auction progression, and task status for the CuedSearch play are similar to those of the other plays. One minor exception is that the utility function that calculates task values must account for both types of resources.

3.2 Contact Generation and Reporting

To ensure consistency across all of the experiments, searchers were designed to "discover" contacts in predetermined cells over the course of the search. The cells in which contacts were to be identified were incorporated into the CuedSearch play and the tactics discussed in Section 3.4 so that the vehicles that eventually executed those search tasks could report the contacts appropriately. Section 4.1 provides more detail regarding the specific placement of contacts within the search cells.

3.3 Tactic Development Considerations

Developing tactics from composable, reusable plays presented several design challenges and key questions. In particular, automation via task allocation had to properly combine local and global information and objectives; developed models needed to promote the hierarchical integration and reuse of the task allocation method; and tactics were subject to the constraint that individual UAVs execute only one play at a time. These considerations align solutions to an ST-SR-IA taxonomy and not a ST-MR-IA coalition as discussed in Section 2.2.1. This section expands on these principles and requirements and examines methods to compose the plays described in Section 3.1 into higher-level tactics in the ARSENL framework.

3.3.1 Play and Tactic Composition

At the most basic level, the tactic must implement logic to determine what play to execute at any given time. Chapter 2 discussed relevant composition mechanisms and examples to this end, including the coordinator mechanism and system-agent-component model. From these examples, it is evident that the tactic should delegate or pass control of the UAV to a component play until either rescinding control and assigning control to another play or receiving control back from the play when appropriate conditions are met. The former option is achievable through direct action implemented in the tactic's control logic (i.e., its `maneuver_command` method). The latter option can be achieved by implementing it in the play's control loop or via an event-based framework. Each of these approaches lends itself to specific architectures. This research implemented a hybrid approach using direct control of play selection by the tactic and an event-based system for play-to-tactic notifications. Section 3.4.1 details the implementation.

The case was made in Section 2.1.2 that optimization and allocation techniques, an auction in this research, can be considered a component within the model and as such should be added in accordance with the composition mechanisms. However, as noted in Section 3.1 the ARSENL auction utilized in `SwarmSearch` and `ContactInvestigation` is tightly coupled with the play's control loop and local UAV maneuver state. For this reason this research chose to use an exogenous composition in which control is passed to a play once per cycle and returned to the tactic after the play's maneuver is determined. Thus, when a behavior is implemented as a tactic, the tactic explicitly calls the active play's `maneuver_command` method to effectively cede control to the play for one control cycle.

3.3.2 Play and Tactic Objectives

The ultimate objective of the tactics remains the effective assignment of swarm vehicles to roles that maximize the total utility. When composed, the single-item auctions used in the `SwarmSearch` play and `ContactInvestigation` play are independent auctions contained within the respective plays. These auctions maximize utility for the local objective of the play but do not account for broader objectives of the tactic. That is, these auctions are distinct, have no awareness of other auctions or tasks, and require that the participants in the auction be known. Additionally, the play's task state is intrinsically tied to the auction and UAV maneuver states. This segregation creates several issues that must be addressed by the tactic to synchronize play- and tactic-level knowledge on a single UAV. Tactics and plays must also synchronize required knowledge with the rest of the swarm. Swarm-wide synchronization has been addressed in the ARSENL system through existing consensus algorithms [37], [71]. Synchronization of the tactics and the plays of which they are composed, on the other hand, must be dealt with as a tactic is developed.

Section 2.1.2 covered suggestions from Lau [48] regarding information flow for an exogenous composition. For the ARSENL cases this can occur via defined communication mechanisms or through parameterization of the `maneuver_command` method with a state vector or via a return value. The `maneuver_command` method is an important element within the hierarchical ARSENL framework with specific return-value requirements, so this research did not choose to modify it. It does, however, present a compelling case for information flow in a composition through passed parameters. Chapter 5 recommends exploration of this concept. Ultimately this research chose to incorporate communication mechanisms into the composed plays to allow direct inspection by the tactic and to provide key event notifications as described in Section 3.4.

Since physical control of the UAV is the responsibility of the play associated with the assigned role, the main function of the tactic control logic becomes the management of the role-allocation process on each UAV. In this context, the roles, then, can be considered tasks that are managed by the tactic, and the tactic itself can be developed to solve the MRTA problem. To accomplish this in a distributed manner, all UAVs must agree on the role requirements (i.e., the number of vehicles to be assigned to each role), and the agreed-upon requirements will make up the auction's available tasks. The discussions that follow are focused on the tactic allocation methods that are used by the tactics developed for this research.

3.3.3 Role Assignment Approaches

There are many possible solutions for determining each UAV's role within the tactic. In the simplest scheme roles can be determined once when the behavior is initiated. This type of role assignment is referred to as "static" for the remainder of the paper and implies that once assigned a role, the UAV remains in that role until behavior completion. More robust approaches to role assignment might allow for a UAV's role to change over the course of the tactic's execution. "Dynamic" determination will refer to an approach that allows for the assignment or reassignment of roles due to changes in task availability. Determining the appropriate number of UAVs for each role is the primary focus of dynamic swarm control at the tactic level.

Static Role Assignment

In the ARSENL framework static allocation is most easily executed when the tactic is initialized by the `parameterize` method. Once assigned, the vehicle will maintain that role until conclusion of the tactic. Each UAV then, participates in only one play, and therefore one auction for the duration of the tactic.

One approach to static allocation is to assign roles heuristically based on some metric or feature associated with the swarm or the UAVs. With this approach, Using the vehicle type is one example of this type of allocation. A search-and-investigate tactic might take this approach and assigns all fixed wing UAVs to the searcher role (i.e., `SwarmSearch` play) and all quadcopters to the pouncer role (i.e., `ContactInvestigation` play) in order to align the roles with the specific vehicle capabilities. This approach effectively segregates the swarm into homogeneous coalitions or subswarms that only indirectly interact.

This scheme is acceptable for some scenarios, but discounts any benefit from heterogeneity of the UAV swarm and limits potentially useful emergent behaviors that might arise as a result. It also assumes that the swarm will include a sufficient number every type of vehicle to satisfy the requirements of all roles.

Other approaches to static role assignment might include randomized assignment, more robust heuristics, and collaborative approaches such as auctions. These approaches might yield advantageous combinations that better leverage heterogeneous capabilities to meet system-wide role requirements. The exploration of these approaches, however, would amount to little more than a stepping stone to the dynamic role-assignment approaches discussed in the next section and is left to future work.

Dynamic Role Assignment

With dynamic role assignment a UAV can be assigned to any role at any point during execution of the tactic. For this research there is only one role available at initialization, so all UAVs are initially assigned to the searcher role. UAV transition to the pouncer role occurs as contacts are generated and the composition of available search and investigate tasks changes. The goal of assigning each UAV to a role in a way that attempts to maximize global utility can occur through any number of task allocation schemes including those discussed in Chapter 2. The tactic implementations in this research use a single-item auction for dynamic role assignment in attempt to achieve performance characteristics similar to those of the `CuedSearch` play.

The dynamic tactics created for this research utilize a greedy approach implemented with an auction to assign pouncer roles as contacts become available. This auction is only conducted when conditions of a role membership function are met to constrain the number of UAVs assigned to each role. The auction only accounts for transition tasks to the pouncer role since searching is the default.

Initial experimentation utilized coalition formation techniques that balanced the ratio of UAVs assigned to each role based on available tasks or quantifiable work requirements associated with each role. However, this is not directly analogous to the search-and-pounce tactic in which contact discovery depends on continued search progression. Stated differently, in rare cases this approach can over-allocate pouncers and halt search progress before eventually self correcting. In the final implementation, a function was utilized to limit the number of pouncers rather than continually balancing the pouncer-to-search ratio. The final tactic capped the number of vehicles assigned to the investigate role as

$$P_{max} = \min\left(\lfloor r_p * n \rfloor, n - 1\right) \tag{3.5}$$

where P_{max} is the maximum number of pouncers allowed at one time, and r_p is the preferred ratio of pouncers relative to the number of available UAVs, n. This ensures that at least one will be allowed to remain in the search task even if r equals 1.0 and that at least one pouncer will be allowed so long as $r_p * n$ is greater than or equal to 1.0.

This simple approach allows for component plays to conduct their auctions for task assignment while the tactic assigns additional UAVs to the role. The tactic's auction utilizes the same utility as the component play to identify the best candidates for transition to the pouncer role. When properly implemented, transitioned UAVs will remain in the pouncer role as long as work is available and will transition back to a searcher role once all available pounce work is complete.

3.4 Tactic Implementation

Three tactics were developed using plays, composition, and role-assignment methods discussed in this chapter: `SearchTacticStatic`, `SearchTacticDynamic`, and `SearchTacticImmediate`. This section discusses their implementation and the motivation behind the various design decisions.

3.4.1 Common Components

All three tactics rely on a number of common data structures and composition components. Role definitions are managed as string-ID pairs that are stored in a bidirectional lookup table, and lists of UAVs assigned to each role are stored as Python sets. The sets for both roles are stored in a Python dictionary that is keyed with the role IDs.

Exchanging role information as UAVs transition from one role to another is required to maintain UAV role awareness across the swarm. A `TacticState` message containing a

UAV's ID and an updated role ID was added to the ARSENL library to support this requirement. Three supporting methods were incorporated into the tactics to process these messages and manage the current role sets:

1. The `send_role_change` method is used to notify the swarm when a local role change occurs. It creates the `TacticState` message with the updated role information and uses the default `PlugInBehavior` base class method to broadcast the message to the rest of the swarm.
2. The `process_role_update` method processes `TacticState` messages received from other UAVs. It is called by the receiving UAV's `process_behavior_data` method and calls the `update_role_sets` method after unpacking the message to effect the update.
3. The `update_role_sets` method determines whether the message's UAV and role IDs represent a change from the current role sets. If so, it updates the role sets accordingly and ensures that the intersection of the two sets is empty. It also calls the methods for both component plays to update their participant lists.

The `ContactInvestigation` play was modified from the original ARSENL version to allow addition of locally generated contacts (i.e., contacts generated while UAV was in the searcher role) since vehicles do not receive their own `ContactReport` messages. This update simply duplicated functionality from the in `process_behavior_data` method that processes contacts reported by other swarm members.

Rather than patching the `SwarmSearch` play, all three tactics were designed to implement contact generation and reporting as described in Section 3.1.5 at the tactic level. Either approach would be functionally equivalent, but this design decision was based on the observation that that there is no role-to-role communication in the implementation. In particular, this choice recognizes that both the parent tactic and the component plays need to be aware of generated contacts.

The component plays were also updated to support event-based state change notification. Finally, two supporting classes, `Role` and `Auctioneer`, were developed to wrap component objects and provide accessor and mutator functions for play maneuver, task, and auction states.

The Role Class

A `Role` class was developed as a wrapper for the composed `SwarmSearch` and `ContactInvestigation` plays within the containing tactics. The `Role` class's primary function is to provide accessor methods for the play's underlying data, to abstract component play data for use by the parent tactic, and to provide a tactic interface for play parameterization and state data management during role transitions.

The `Role` class also provides methods that are specific to managing the component play's auction. When the UAV's role changes, these methods ensure that there is a graceful transition between plays by setting the auction and task states appropriately and notifying the swarm of any updates. The auction-management methods provided by the `Role` class are as follows:

1. The `exit_auction` method sets the conditions to transition out of an active component play. It ensures that the swarm is notified of any recently completed task, and in the case of search tasks being relinquished prior to transition, it updates the search cell's resource status to "available," notifies the swarm of the status change and then immediately calls the `force_auction_restart` method to force the remaining participants to auction the now available resource. Finally, it updates the component play's current and next task references to `None` so that the play does not restart a task if it transitions back to this role later.
2. The `force_auction_restart` method sets the conditions to initiate a new auction among the remaining role participants. Specifically, the composed play's auction state is set to `INACTIVE`, and the component play's auction restart method is called. The restart method sends an auction reset request to the remaining participants and changes the local auction state to `RESTART`. The restart ensures that any relinquished task is auctioned off immediately, and setting the play's auction state to `RESTART` ensures that an auction will be conducted immediately if the play is reactivated later as a result of a subsequent role transition.

The `Role` class also registers the tactic as an observer to the play to enable event-driven notification of state changes within the play. This approach was chosen as an alternative to adding parameters to the `maneuver_command` method to avoid changes that would affect the control approach discussed in Section 3.3.1. Play-level state change notification enables the tactic to initiate a role transition immediately upon completion of an assigned play task (i.e., prior to commencing a follow-on task) if desired. Event notification is implemented by an update to the composed plays' `compute_maneuver` methods.

The Auctioneer Class

An `Auctioneer` class was developed to provide an object and event-driven replication of the state-machine for auction execution and to provide accessor and mutator methods for frequently used auction information. Essentially a convenience class, the `Auctioneer` reduced the complexity of integrating the auction with the tactic control cycle. As such, its use was a design decision that facilitated the development process and was not essential to achieve the overall composition goals.

The primary advantage of the `Auctioneer` class is the encapsulation of specific methods that were replicated in the plays `SwarmSearch`, `ContactInvestigation`, and `CuedSearch` plays. However, many of the auction methods are actually assigned when the auction is initialized meaning that the implementing tactic or play must still provide some of the supporting methods and algorithms. In particular, the utility function `evaluate_auction_tasks` must still be provided since it implements the auction-specific utility calculations and task comparisons. Similarly, the `init_auction_tasks` method differs between auctions since different task types call for different initialization parameters. Chapter 5 provides suggestions for further encapsulating of these methods.

The `Auctioneer` class utilizes an abstract `AuctionState` class with four methods: `update`, `resume_auction`, `network_resume_request`, and `add_resource`. These methods implement state-specific requirements of the original play implementations in a manner that allows for transparent interaction with the tactic. Concrete classes are defined for each `Auction` class state and the abstract member functions are implemented with only the state-specific code. The advantage of this approach is that it decouples the auction state from the maneuver state.

Tactic Control

The composition of plays within the tactic makes implementation of the actual control function fairly straight forward. To execute the play, the tactic simply calls the active play's `compute_maneuver` method from the tactic's `compute_maneuver` method. As noted previously, the most important component of the tactic's control function is role assignment. Since this is implemented differently for each tactic, it will be covered with the specific tactics. There are, however, functions supporting role assignment that are common to all three tactics.

1. The `get_investigator_limit` method implements the role membership function and calculates the maximum number of allowed investigators as described in Section 3.3.3.
2. The `notify_role_end` method is an event-driven method called by the component plays observers. It is called when the play's maneuver and auction states are set to `COMPLETE` indicating all tasks associated with the role have been completed. When invoked, it initiates a transition to the other role if that role has remaining tasks. For all tactics this method is called by the composed `SwarmSearch` play so that searchers will always transition to pouncers when the search area is complete. With the `SearchTacticDynamic` and `SearchTacticImmediate` tactics, this function will also be invoked by the composed `ContactInvestigation` play to transition pouncers back to the searcher role when there are no remaining contacts to be investigated.
3. The `calculate_center_mass` method calculates a point central to the current swarm geographic distribution. This point is used as a "loiter" point for UAVs with no assigned task. The function ensures that both composed plays are using the same loiter point so that UAVs are in closer proximity to potential future tasks.

Finally, it is worth noting that the tactics' `proccess_behavior_data` and `parameterize` methods did not benefit from composition like the `maneuver_command` did. Instead these two methods needed to account for all requirements of the tactic and all of the composed plays. That is, these methods must ensure not only that initialization and message data handled correctly at the tactic level, but also that data is passed to component plays correctly. This meant that even when there was commonality among the tactics, small differences in initialization and communication requirements precluded reuse by multiple tactics.

The common components described in this subsection comprise the majority of the implementation details associated with the composition of plays within a tactic. Tactic implementations then, are primarily differentiated by their approaches to role assignment. The remainder of this chapter will discuss role assignment for each of the three tactics.

3.4.2 Static Role Assignment

The `SwarmTacticStatic` role-assignment implementation is the simplest of the three tactics. It was not developed to prioritize anything more than ease of implementation and there were few specific performance expectations. It does, however, provide a point of comparison to the more robust approaches of the other two tactics.

The `SwarmTacticStatic` role assignment occurs within the tactic's `parameterize` method, and roles do not change during execution except at the end of the behavior when there are no remaining `SwarmSearch` tasks. Role assignment utilizes the maximum pouncer ratio (Equation 3.5) with an r value of 0.5 to divide the swarm evenly between pouncers and searchers. Quadcopter UAVs are assigned to the pouncer role in vehicle-ID-order until the desired number of pouncers has been identified or there are no remaining quadcopters. If there are an insufficient number of quadcopters, fixed-wing UAVs are assigned to the pouncer role by vehicle ID as required. All UAVs not assigned to the pouncer role are assigned to the searcher role. The swarm composition of the experiments described in Chapter 4 always led to role assignments in which all quadcopters were assigned to the pouncer role and all fixed-wing UAVs to the searcher role.

No additional methods were required for this tactic. Of note, vehicles assigned to the pouncer role have no assignments until contacts are generated. This led to an assumption that under use of loitering vehicles might negatively affect search performance.

3.4.3 Dynamic Role Assignment

The dynamic role assignment that occurs in the `SearchTacticDynamic` and `SearchTacticImmediate` tactics and utilizes a single-item auction to identify suitable UAVs to transition to a pounce role as contacts are generated. For the `SearchTacticDynamic` tactic the actual transition to a new role does not occur until the selected UAV has completed any ongoing search task. The `SearchTacticImmediate` tactic, on the other hand, immediately transitions the selected UAV to the new role. The differences between these two approaches are presented in Section 3.4.4, but it is worth noting here that both tactics utilize similar auction processes to select UAVs for transition.

Dynamic assignment was implemented to replicate the semantics of the `CuedSearch` play. Since the role-assignment auction is conducted when contacts are generated to potentially transition searchers to a pouncer role, the process is analogous to UAVs winning available pounce tasks. In line with this reasoning, the `SearchTacticDynamic` tactic auction utilizes the same basic utility function from Section 3.1.2 that is used by the `ContactInvestigation` and `SwarmSearch` plays (i.e., Equations 3.1 and 3.2). From the standpoint of the searchers, pounce tasks equate to role transitions and are thus specified as `transition tasks` in the tactic auction. The auction that will eventually assign the newly reported contacts occurs within the `ContactInvestigation` play and will be limited to that role's participants (to include recently transitioned searchers).

The tactics consider each newly reported contact as a transition opportunity and conduct transition auctions to assign all available transition tasks. In the simplest case, one searcher is identified for transition to a newly available task. It is possible however, that the best-suited UAVs are already assigned a pouncer role. In these situations, UAVs already in the pouncer role will win the available transition tasks, and no transition will occur. Situations also exist in which multiple transition tasks are available, so multiple searchers might win transition tasks. When this occurs, a greedy approach is taken, and the UAVs that is first able to execute their transition will do so. Once the maximum number of pouncers is reached (i.e., per the role membership ratio), no further UAVs will be permitted to transition even if they previously won a transition task. When a pouncer completes an assigned task with no additional pounce tasks available, it will immediately transition back to a searcher role. This transition scheme is designed to mirror the semantics of the `CuedSearch` play in which UAVs executing investigation tasks are more likely to be awarded additional investigation tasks than UAVs executing search tasks.

The basic approach to transition auctions and post-auction transition of UAVs is presented in Algorithm 1. The algorithm specifically implements the `SearchTacticDynamic` semantics and requires completion of an in-progress search task before a transitioning to a pouncer role. As noted in the last paragraph, it also prevents transitions that would result in too many pouncer-role UAVs. The `current` and `next` variables represent transition tasks and are used to initiate transitions and update task status. A non-`NONE` value in the `current` task indicates that the UAV is already in the pouncer role, while a non-`NONE` `next` task indicates that the UAV is queued for transition to the pouncer role.

Algorithm 1 Dynamic Role Assignment

```
Require: $new\_task$
Require: $T_L = [...]$
   $T_L = T_L \cup new\_task$
   $Auction.state \leftarrow RESET$
   while $assignment$ = NULL do
      $assignment \leftarrow auction.update()$
   end while
   if $current\_role = SEARCHER$ then
      $next \leftarrow assignment$
      $assignment \leftarrow null$
   else
      $next \leftarrow NONE$
   end if
Require: $role.current.state \leftarrow COMPLETE$
   if $role = SEARCHER$ then
      $remaining \leftarrow membership\_limit - len(S_p)$
      if $next$ and $remaining > 0$ then
         $current \leftarrow next$
         $role \leftarrow POUNCER$
      else
         $next \leftarrow NONE$
      end if
   else if $role = POUNCER$ then
      $T_L[current].state \leftarrow COMPLETE$
      $current \leftarrow NONE$
   end if
```

A new transition task is created for any reported contact using that contact's ID and geographic information. Upon adding the new resource, a new auction is initiated. After completion of the auction, the awarded task is copied to the `next` variable if the UAV is currently in a searcher role (there is no need to update UAVs in a pouncer role since the awarded task represents a potential transition to the pouncer role). If the auction resulted in the award of a transition task, this queues the tactic for a role transition.

When called for, a UAV transitions to the pouncer role as soon as the UAV finishes its current search task. This notification is made by the observer member added to component plays during role initialization. The observer invokes the tactic's `notify_task_complete` method which verifies the `next` task for UAVs in the searcher role. If the `next` value is a valid transition task, it then checks the current size of the role set against the role size limit. If the limit has not been reached the method initiates the role change. If the UAV's role was already a pouncer, the current task resource is updated to the `COMPLETE` state.

Algorithm 1 functionality is implemented across several methods including the previously mentioned `notify_task_complete`, `process_behavior_data`, and `compute_maneuver`. One additional method, `notify_new_task`, was implemented to maintain task synchronization between the role-assignment auction and play auction. The `notify_new_task` method is the handler method for another event-driven addition that informs the tactic when the play has started a new task. This method updates the corresponding transition resource to `ACTIVE` and assigns the tactic's `current` task to the new resource. This method also accounts for instances where null tasks are assigned in either role. In this case, pouncers will immediately transition back to searchers and searchers will transition to pouncers if the pounce limit has not been reached.

The role-assignment algorithm was chosen to minimize the number of auctions occurring at the tactic level so that the number of auctions executed equals the number of contacts discovered. That diverges slightly from the `CuedSearch` play auction approach, but was considered a reasonable concession to reduce overhead.

3.4.4 Immediate Role Assignment

As implemented in the `SearchTacticDynamic` tactic and `CuedSearch` play, role transitions do not occur until a UAV has completed its most recently assigned task. The `SearchTacticImmediate` tactic was developed to prioritize for contact response by transitioning to the pouncer role immediately. To achieve the immediate transition, the role assignment algorithm is modified as depicted in Algorithm 2 by removing the task completion requirement.

The immediate role-transition algorithm must also to account for the fact that multiple UAVs may transition immediately after an auction if more than one transition task is available.

Algorithm 2 Immediate Role Assignment

```
Require: $new\_task$
Require: $T_L = [...]$
  $T_L = T_L \cup new\_task$
  $Auction.state \leftarrow RESET$
  while $assignment = NULL$ do
      $assignment \leftarrow auction.update()$
  end while
  if $current\_role = SEARCHER$ then
      if $\neg assignment.isnull()$ then
          $remaining \leftarrow membership\_limit - len(S_p)$
          if $remaining > 0$ then
              $available \leftarrow len(T_L.available\_ids())$
              $available \leftarrow \min(available, remaining)$
              if $assignment.id \in T_L.available\_ids()[0...available]$ then
                  $current \leftarrow assignment$
                  $role \leftarrow POUNCER$
              end if
          end if
      end if
  end if
```

This could lead to a violation of the role membership ratio if too many UAVs transition. To account for the immediate transition Algorithm 2 compares number of available pouncer roles to the number of remaining available transition tasks and selects the smaller of the two. It uses this number to identify a subset of the oldest available tasks. If the UAV's assigned task is an element of this subset, the UAV transitions to the pouncer role. Otherwise the transition is discarded. This approach effectively prioritizes older contacts over newer ones when the number of contacts exceeds the maximum number of pouncers.

The logic for Algorithm 2 is primarily primarily implemented in a `process_auction_win` method of the tactic. The event callback, `notify_task_complete`, is still utilized to provide synchronization support to the `SearchTacticImmediate` tactic, but the original dynamic transition logic is no longer required.

One additional factor must be considered if tasks are to be aborted in lieu of a higher priority task: effort already invested into a task becomes more relevant than effort required to complete the task. To account for this, the utility function for the immediate transition auctions was modified to account for time already spent on the current task, t_{s_i} instead of time remaining in a task t_{r_i}. This results in the following formula for the time component of the utility function:

$$t_{s_i} + t_{t_r} + t_r \tag{3.6}$$

Incorporating time spent on the current task into the equation amounts to a bias towards searchers that have completed less of their search assignment than other bidders. The intent is to decrease the time eventually spent researching relinquished cells.

3.4.5 Summary

This chapter described the development of three UAV swarm tactics for performing a search-and-pounce mission in which a geographic area is searched and contacts that are identified over the course of the search are investigated. All three tactics are composed of the ARSENL `SwarmSearch` and `ContactInvestigation` plays. Each tactic utilizes a different approach to the assignment of investigators to contact investigation (i.e., pounce) tasks: static assignment the pounce role, dynamic assignment with delayed transition, and dynamic assignment with immediate transition. The compositions were achieved with a common set of supporting functions requiring only role allocation methods be modified to achieve specific goals. This supports the MASC goals of reusing simple behaviors for creation of more complex tactics. The intent of all three tactics is to produce a behavior that is similar to a play implemented from monolithic code. Chapter 4 describes experimentation with these behaviors to include a description of the metrics of interest and performance comparisons to the preexisting ARSENL `CuedSearch` play.

CHAPTER 4: Results

This chapter presents results of simulation and live-flight experimentation for the four behaviors that were described in Chapter 3. The three tactics, `SearchTacticDynamic`, `SearchTacticImmediate`, `SearchTacticStatic`, and one play `CuedSearch` were each tested on heterogeneous multi-UAV swarms within the ARSENL SITL environment and during live-flight testing at McMillan Airfield. Further reference to the tactics will be abbreviated to `Dynamic`, `Immediate`, and `Static`. Tests directed swarms to search a predetermined geographic area, and investigate a predetermined number of dynamically generated contacts. Live-flight testing was conducted to demonstrate real-world applicability and verify simulation results.

The performance of the three tactics composed of `Contact Investigation` and `SwarmSearch` plays are compared to the analogous `CuedSearch` play. The chapter is presented in three sections: experimentation methods, data preparation, and experimental results. The experimentation methods section details the testing environment including search area descriptions and contact generation. The data preparation section describes the data post-processing that was conducted to account for observed anomalies and discusses the causes of those anomalies. Finally, the results section presents an analysis of the experimental outcomes with emphasis on timed metrics, vehicle utilization, and task distribution.

4.1 Experimentation Methods

Experimentation consisted of events conducted in the SITL simulation environment and live flights conducted at McMillan Airfield, Camp Roberts, CA. Both simulation and live-flight events utilized heterogeneous UAV swarms composed of Zephyr II fixed-wing and Mosquito Hawk quadcopter vehicles. The characteristics of these vehicles is provided in Table 4.1

Experiments utilized two geographic search areas, both anchored on an initial geographic location at McMillan Airfield but varying in size. The live-flight search area was defined as a

Table 4.1. ARSENL UAV Platform Characteristics. Adapted from [26]

	Dimensions	**Maximum Endurance**	**Cruise Speed**	**Autopilot Computer**	**Companion Computer**
Zephyr II (blended wing)	1.45m (wingspan)	50min	18m/s	Pixhawk	Odroid U3
Mosquito Hawk (quadcopter)	0.29m (motor axis to motor axis)	20min	15m/s	PixRacer	Odroid C0

575 meter by 750 meter rectangle located as depicted in Figure 4.1. Both live-flight and SITL environment experiments were conducted with this area. SITL simulation experiments were also conducted in a large search area defined by 1200 by 1950 meter rectangle as depicted in Figure 4.2. Safety-of-flight requirements and airspace limitations precluded live-flight experimentation in the large area.

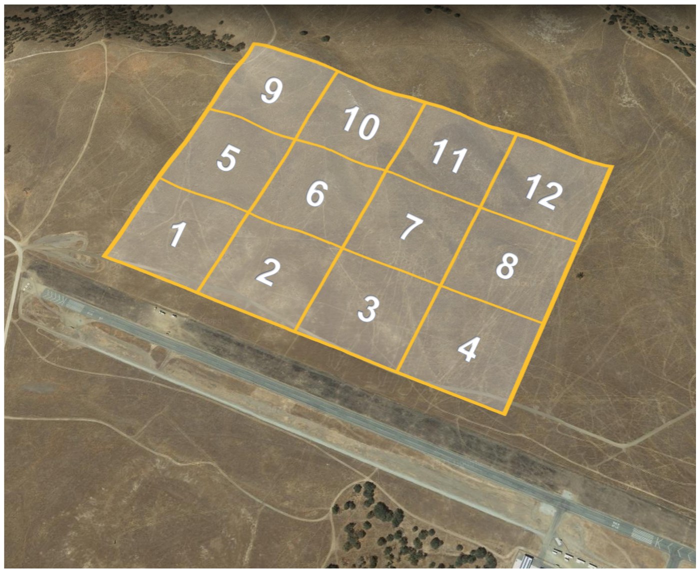

Figure 4.1. Live-Flight Search Area Cells.

Search cells were generated at runtime to partition the specified search areas using functions from the `arsenl_behavior_tools` package's `search_planners` subpackage. These functions divide an area into equally sized cells according to maximum length and width parameters. Plays in this experiment used maximum length and width of 200 and 225 me-

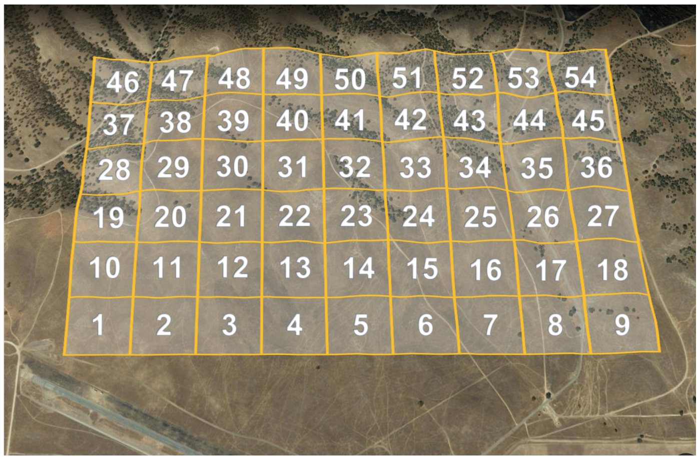

Figure 4.2. Large Search Area Cells.

ters respectively. This resulted in 12 search cells arranged in a three-by-four grid for the live-flight search area and 54 search cells arranged in a six-by-nine grid for the large search area. Note that the size of the search cells in the live-flight area were 191 meters by 185.7 meters, which was slightly smaller than the 200 meter by 216.6 meter search cells in the large search area. This is due to approach used by the `RectangleSearchArea`'s algorithm described in Section 3.1.3 to create equally sized cells.

Simulated contacts were associated with specific search cells to force consistency of discovery events across multiple tests. The live-flight search area was seeded with six contacts located in cells 1, 4, 5, 7, 8, 10 of Figure 4.1, and the large search area contained 18 contacts located in cells 1, 4, 5, 8, 11, 13, 16, 17, 19, 21, 24, 26, 27, 30, 31, 32, 35, 39 of Figure 4.2. Contacts were generated within the specified cells by the vehicles searching those cells and reported to other members of the swarm accordingly. The discovery event for a contact occurred after a vehicle completed searching 35 percent of the respective area. The subsequent report contained the contact's location and a unique contact identification (integer).

SITL environment and live-flight tests were conducted for swarm sizes of three to 10 UAVs for the live-flight search area, and SITL environment tests with swarm sizes of six to 10 UAVs were conducted in the large search area. All swarms consisted of a near-even mix of fixed-wing and quadcopter vehicles. For swarms of *n* UAVs, the number of quadcopters,

n_q, was $\lfloor n/2 \rfloor$, and the number of fixed-wing UAVs, n_f, was $n - n_q$. In the SITL simulation, each swarm size was tested a minimum of five times, with additional tests conducted if permitted by simulated vehicle battery life. A single live-flight experiment was conducted for each presented swarm configuration to validate the SITL environment results.

4.1.1 Test Adjustments

Two versions of the code base were utilized during testing, both of which are represented in the analyzed results. Version one testing was conducted using both the SITL environment and during live-flight events at McMillan Airfield. Implementation errors that did not manifest in simulation were identified during testing at McMillan Airfield. Code corrections applied in the field allowed the successful completion of two `Immediate` behavior tests of swarm size eight, but other behaviors and swarm sizes were not completed due to time constraints. Version two is the version described in Chapter 3. It was utilized in the SITL environment for both search area sizes and is the exclusive data set used for the large area analysis. The `CuedSearch` play was not affected by these changes.

4.2 Data Preparation

The data gathered from testing was verified to ensure the expected number of search cell and contact tasks were assigned and completed for each test. During the course of verification a number of anomalies including repeated tasks and incomplete tasks were noted. This section analyzes these events and proposes possible causes along with mitigation steps used to continue analysis.

Results from 390 total simulations were included in the analysis presented in Section 4.3. Preliminary verification of test performance revealed reports of both search and pounce tasks completed more than once, in some cases by multiple vehicles. This was not expected since the task allocation mechanism was implemented with the goal of tasks being assigned and completed only one time. The set of simulations with repeated tasks consisted of 158 unique tests with a total of 191 repeated search and pounce tasks. The breakout is displayed in Table 4.2.

There was also one case noted where a contact was reported twice, but this duplicate report

Table 4.2. Number of Simulations with Repeated Tasks.

Behavior	**Area**	**Searches**	**Pounces**	**Total**
`CuedSearch`	Large	23	19	42
	Live-flight	6	16	22
`SearchTacticDynamic`	Large	11	22	33
	Live-flight	2	14	16
`SearchTacticImmediate`	Large	9	23	32
	Live-flight	11	16	27
`SearchTacticStatic`	Large	4	10	14
	Live-flight	0	5	5
total/event		**76**	**125**	**191**

did not result in a duplicate pounce. The second detection was determined to stem from a searcher that relinquished a cell to pounce on a contact (i.e., when executing the `Immediate` tactic). The cell was subsequently searched again, resulting in the contact being generated a second time.

Finally, three tests were discarded after determining that the behavior was terminated prior to the last task being finished.

Investigation of the duplicates found two primary causes. The first was a race condition, and the second was an incomplete auction participation. The race condition occurred when an auction was conducted when at least one vehicle already had a next task identified. If the vehicle finished its current task and began the next task before the auction completed, the auction would effectively include a task that had already commenced. The change in task status from one of the available statuses to `Active` is checked as part of the new bid submission process; however, when a vehicle is already the high bidder for that task, the same bid is resubmitted without conducting the check. When this occurred, the task was often awarded to the same vehicle. In this case, after completing the task initially, the task is again assigned as current task. The task was noted as complete upon the first control loop iteration, however, and the task was immediately relinquished. If necessary, a new auction was then initiated. Overall search performance was not significantly affected in this case, with time to execute an additional auction being the most significant cost. If, on the other hand, the auction assigned the task to another vehicle, the assigned vehicle would complete the task despite its having already been completed by another vehicle. This occurred because the control loop checked the resource record's driver for the complete state rather than the resource record's state (i.e., it determined when it had physically completed the task as opposed to whether the task had been completed in general).

The root cause of inconsistent auction participation could not be definitively determined from the data collected. In this implementation, a new task-assignment auction is requested immediately by a newly assigned vehicle. The plays, however, ignore auction restart requests during an active auction. It was hypothesized that the possibility existed for vehicles to transition to a new role while an auction was in progress. Regardless, the errors do not coincide with the expected behavior for this case which would be a hung auction in any instance where a vehicle is added after the second round bidding. Evidence suggests that issues with the auction participant list or another issue with the transitioned vehicle's internal auction state prevented it from participating in the ongoing auction correctly. In either case a task is repeated.

Analysis of the duplicate task anomalies showed that a significant number were caused by the race condition and won by the vehicle that already transitioned to the same task. These duplicate completions were removed from the data set since they did not result in a vehicle actually completing an additional task. Removal of these tasks prevented skewing of the task completion and distribution analysis. Removal eliminated approximately 41 percent of the duplicate tasks from the original set as illustrated in Table 4.3. Overall, this left 84 duplicates out of 12,904 total tasks included in the analysis.

Duplicate tasks that were completed multiple times were retained in the data set for analysis because of their small proportion relative to the overall number of tasks completed. The duplicates account for less than one percent of the tasks performed and occurred in only 73 of 378 tests (19 percent), which was considered acceptable for the purposes of this analysis.

Table 4.3. Number of Tests with Repeated Tasks After Filtering.

Behavior	**Area**	**Searches**	**Pounces**	**Total**	**Change**
`CuedSearch`	Large	13	11	24	42%
	Live-flight	6	8	14	36%
`SearchTacticDynamic`	Large	4	16	20	39%
	Live-flight	1	9	10	37%
`SearchTacticImmediate`	Large	6	19	25	22%
	Live-flight	2	11	13	52%
`SearchTacticStatic`	Large	1	4	6	57%
	Live-flight	0	1	1	80%
total/event		**33**	**80**	**113**	**41%**

4.3 Results

Data was captured and analyzed to assess three general aspects of swarm performance: completion times, task distribution, and vehicle utilization. Completion times were examined with respect to time to complete the overall search, the individual search cells, and pounce tasks. Time lost for dropping active searches for pounces during `Immediate` tactic execution is also measured to determine whether it affected pounce and overall search times. Task distribution is presented as a *pounce ratio* that quantifies the number of pounces completed relative to the total number of tasks for each UAV. Finally, utilization provides a measure of time a UAV is actively engaged in completing a task compared to the overall test time.

Together these metrics provide mechanisms to compare performance of the newly developed tactics to the baseline `CuedSearch` play to verify that the composition of plays provide comparable task allocation and overall performance. The hypothesized expectation is that `Dynamic` will most closely approximate `CuedSearch` across all measured metrics examined in this section.

4.3.1 Completion Times

This section reviews various time metrics, including time to complete a test, time to complete the search area, and timed components related to pouncing. These measurements provide comparable performance metrics across tactics; however, they do not totally capture the effectiveness of the vehicle-specific utilities used for task assignment in the heterogeneous swarm. Chapter 3 defined utility functions and parameters used to influence vehicles' preferences for specific task types aligned to their capabilities. With these preferences, we generally expect that the difference between mean completion times will be within overlapping confidence intervals across the three tactics if the ratio of assigned task types to vehicle types is similar. Specific expectations for each behavioral variation are described in the following sub-sections.

Test Completion Time

Total time to completion measures the time the swarm takes to complete all search and pounce tasks in a test (i.e., from behavior initiation to completion of the last task). In general, the overall times to completion for the three tactics were comparable to the total time performance of the baseline `CuedSearch` play. As depicted in Figures 4.3 and 4.4, all behaviors displayed a downward trend in total time to completion as swarm size increased. The live-flight area results indicated diminishing returns as the number of vehicles approached the total number of search cells available. The large area test did not reach a large enough swarm size to display this characteristic.

The displayed 83 percent confidence intervals in Figures 4.3 and 4.4 depict the difference between behavior test means in accordance with [72] with non-overlapping intervals representing a significant difference between means. Non-overlapping mean intervals appear in a few instances throughout testing, however there is only one instance of the mean difference of the baseline `CuedSearch` play outside the confidence intervals of all three of the other tactics which occurred in the live-flight search area with a swarm size of five. The most notable trend is the consistently narrower confidence intervals of `Static` in large search area and non-overlapping intervals in swarm sizes 8-9. This provides a visible behavioral distinction between the behaviors with auction assigned roles and `Static` which is assigned by vehicle type. The remaining paragraphs in this subsection describe expectations and results from each behavior with respect to completion times.

As mentioned in Section 4.3, the `Dynamic` behavior was expected to most closely match the behavioral characteristics of the baseline `CuedSearch`, including completion times. However, `Dynamic`'s average completion time was faster than `CuedSearch` in all but three

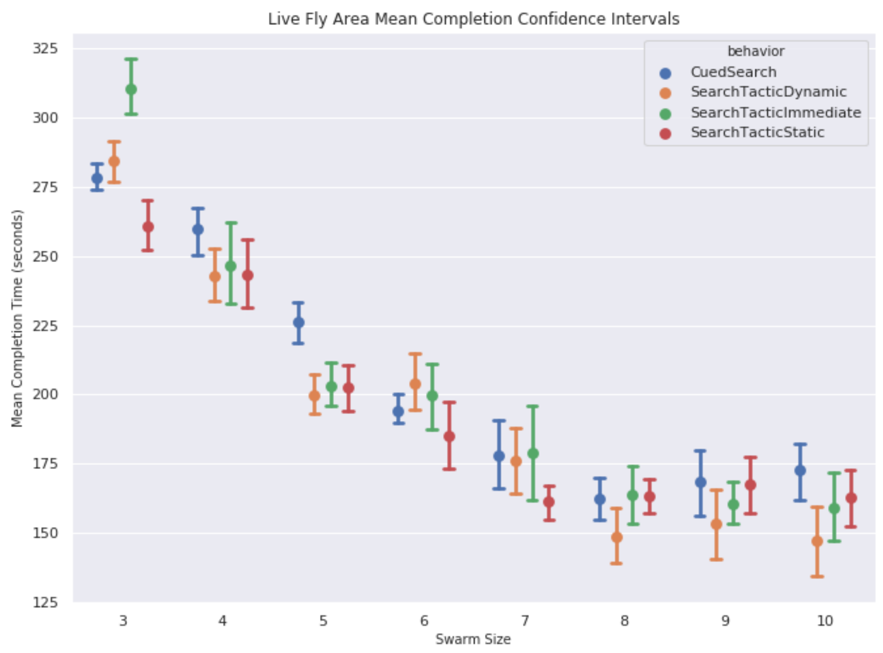

Figure 4.3. Live-Flight Area Test Mean Completion Time Confidence Intervals (83%).

of the 13 test groups. There were also non-overlapping confidence intervals in three of the test groups. Sections 4.3.2 and 4.3.3 investigate the possible reasons for test completion time difference.

The expectation for `Immediate` was that it would perform similarly to `Dynamic` but that it would incur additional time costs due to active assigned searches being interrupted in lieu of prioritized pounce tasks. A relinquished search task required a complete re-search of the search cell when it was eventually reassigned. The predicted relative performance was observed in both search areas as displayed in Figures 4.3 and 4.4. The mean time of `Immediate` was higher than `Dynamic` in all but one instance. Section 4.3.4 discusses the measured completion time increase in comparison to the pounce delay as a design trade-off.

The `Static` tactic has two opposing assumptions for prediction of average test times. One is that because task assignment is segregated by vehicle type, vehicle utilization will be low

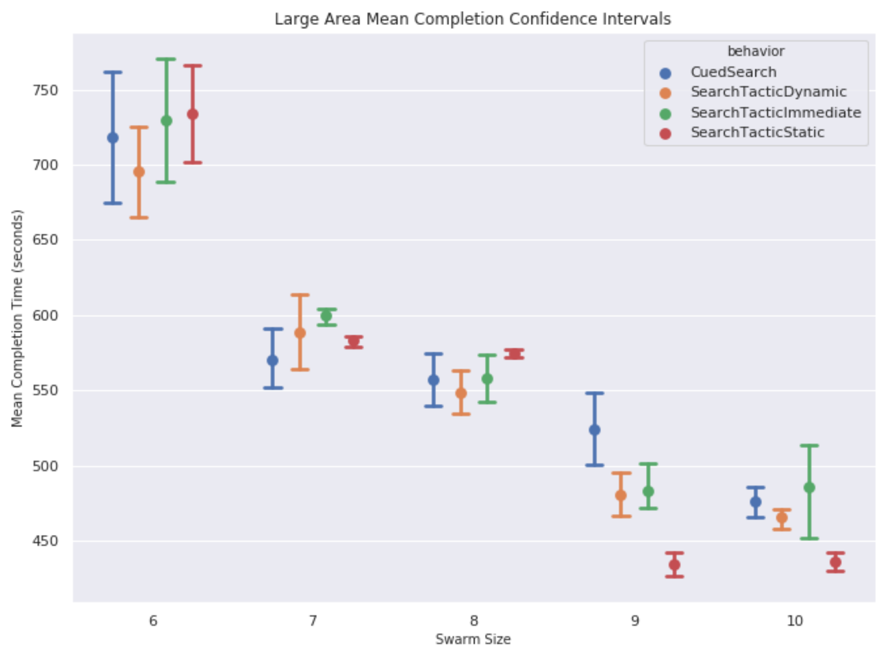

Figure 4.4. Large Area Test Mean Completion time Confidence Intervals (83%).

which will increase test times due to the fact that not all vehicles participate in search tasks and remain idle until a pounce task is available. The other is that test times will be faster due to the longer search tasks always being completed by the faster fixed-wing vehicles. There is merit for consideration of each assumption. Application of the latter assumption can be used to reasonably explain `Static`'s overall faster test completion times than other behaviors in the large search area for swarm sizes of nine to ten UAVs as depicted in Figure 4.4 due to all fixed-wing vehicles dedicated to searching the larger number of search cells. Note that for swarms of 9 to 10 vehicles in the large area, the `Static` test completion times were an average of 31 seconds faster than the `CuedSearch` baseline with only one other test from remaining behaviors within the range of completion times. This advantage is not apparent in the live-flight search area or with the smaller swarms sizes in the large search area. `Static` maintains a competitive average completion time within the range of live-flight search area tests, indicating the utilization handicap is negligible within these parameters or partially offset by the search speed from all fixed-wing vehicles.

Search Completion Time

The search area completion measures the time to complete all search tasks within the test. The predictions stated in Section 4.3.1 can be extended to the analysis of search completion times. However, there is no expectation of a strict positive or negative correlation to search area completion time and test completion time due to the variation in task distribution and execution. In other words, an increase in search completion time could have no effect because there are still pounce tasks being completed, or it could have a negative correlation because an increased search time allowed a pounce task to finish sooner and decrease the overall test time. However, there is an expectation that search area completion times for `Immediate` will be increased when compared to the `Dynamic` tactic due to the prioritization of pounce tasks. Figure 4.5 shows an exaggerated example of this increase with swarm sizes of six to nine UAVs in the live-flight search area. The large area average search times depicted in Figure 4.6 also concur with the increased time over the `Dynamic` tactic similar to the performance observed during total test completion times. Detailed analysis on time lost is presented in Section 4.3.1.

To the expand on the point of varying correlation of search area completion times to test completion times, analysis shows that in only 16 percent of the tests that a search task was the last task completed during the test. Of those tests 72 percent occurred during the basic large search area. 36 percent of the search tasks that were completed last were from the `Immediate` tactic while `Dynamic` and `Static` split 51 percent and `CuedSearch` accounted for the remaining 13 percent. Figures 4.7 and 4.8 depict the difference between the test completion time and search area completion time. The 0-second difference indicates tests for which completion of the last search task concluded the overall test. All positive values indicate tests for which a contact investigation task was the final task completed. There is a wide time distribution between search area completions and test completions represented in these two figures. However, within the large area there are more search completion times equivalent to test completion times suggesting more opportunity for positive correlation. This suggests that the number of available tasks vs the swarm size is more significant variable towards test completion than the completion of any subset of tasks.

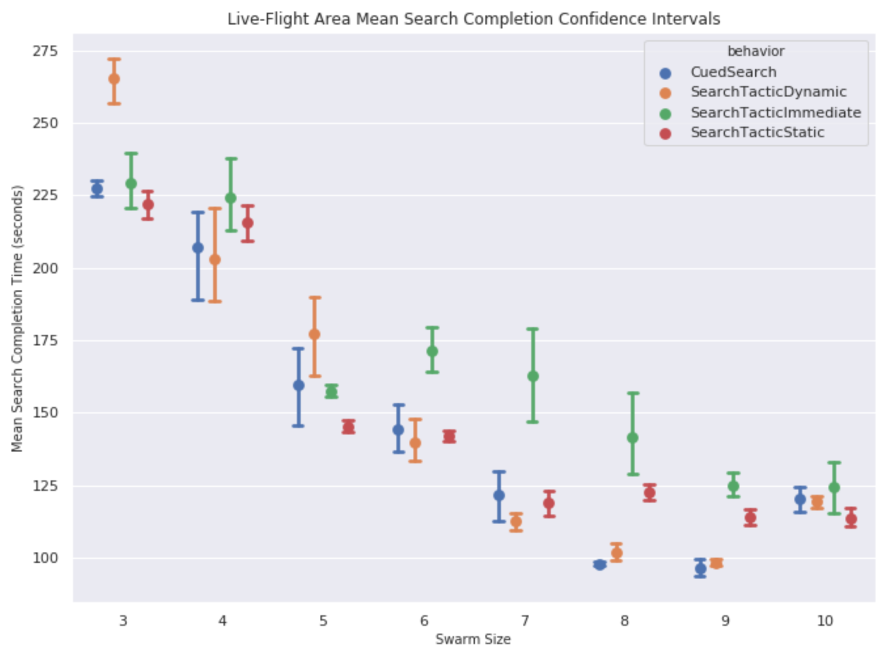

Figure 4.5. Live-Flight Area Mean Search Completion Confidence Intervals.

`SearchTacticImmediate` Time Lost due to Dropping Searches

Figure 4.9 displays the average time spent on `Immediate` search tasks that were discontinued in order to transition to pounce tasks. This immediate transition is the key difference between the `Dynamic` and `Immediate` tactics. The behavior is intended to achieve faster contact response times but incurs the obvious cost of time wasted in discontinued search tasks. As the figure indicates, this loss is overwhelmingly borne by quadcopters. Sections 4.3.3 and 4.3.2 complement this observation with characterization of low utilization and high pounce rates in the `Immediate` behavior. The live-flight area losses can be correlated to the number of dropped tasks described in Section 4.3.4 where the increases time lost between swarm sizes five and six, and seven and eight are due to a single additional dropped task. Similarly, the large search area reflects this increase in average searches dropped at swarm size ten.

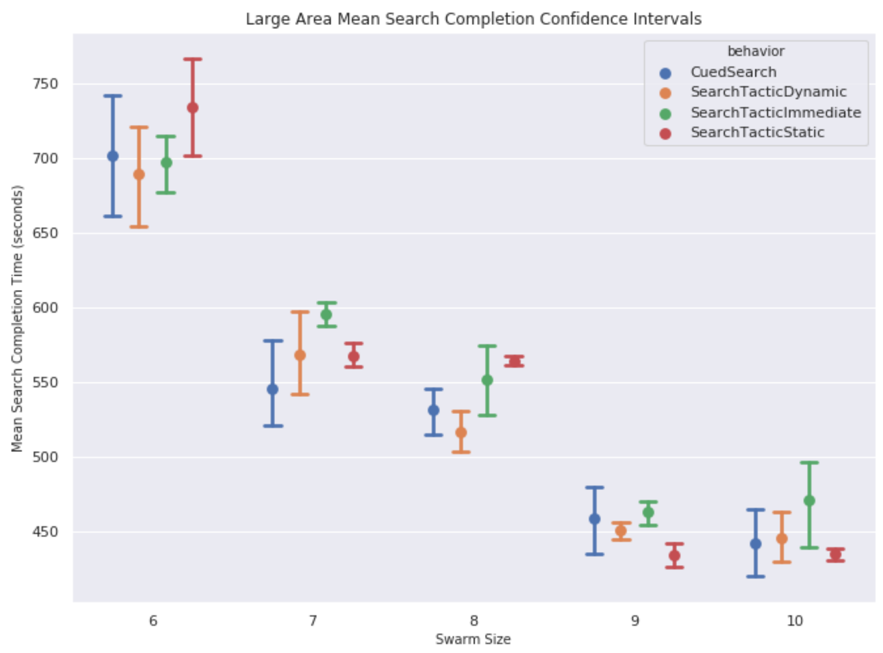

Figure 4.6. Large Area Mean Search Completion Confidence Intervals.

4.3.2 Task Distribution

Task allocation within the heterogeneous swarm is a key performance metric for the `Dynamic` and `Immediate` role assignment auctions. Distributing tasks efficiently according to vehicle capability is a key feature of the baseline play `CuedSearch` as presented in [5]. Thus, comparison of the developed tactics' performances to that of the baseline `CuedSearch` play demonstrates whether assignment of search and pounce tasks to specific vehicles approximates the desired task allocation and utilization metrics.

Task allocation is assessed using a *pounce ratio* that measures the percentage of pounce tasks completed relative to the total number tasks completed for a specific vehicle type. The expectation is that the vehicle subsets assigned to search and pounce tasks by the `Dynamic` tactic will have a similar makeup to those of the `CuedSearch` play. The expectation for both of the tactics is that quadcopters will complete the majority of pounce tasks due to the airframe specific utility used when computing task values for bidding.

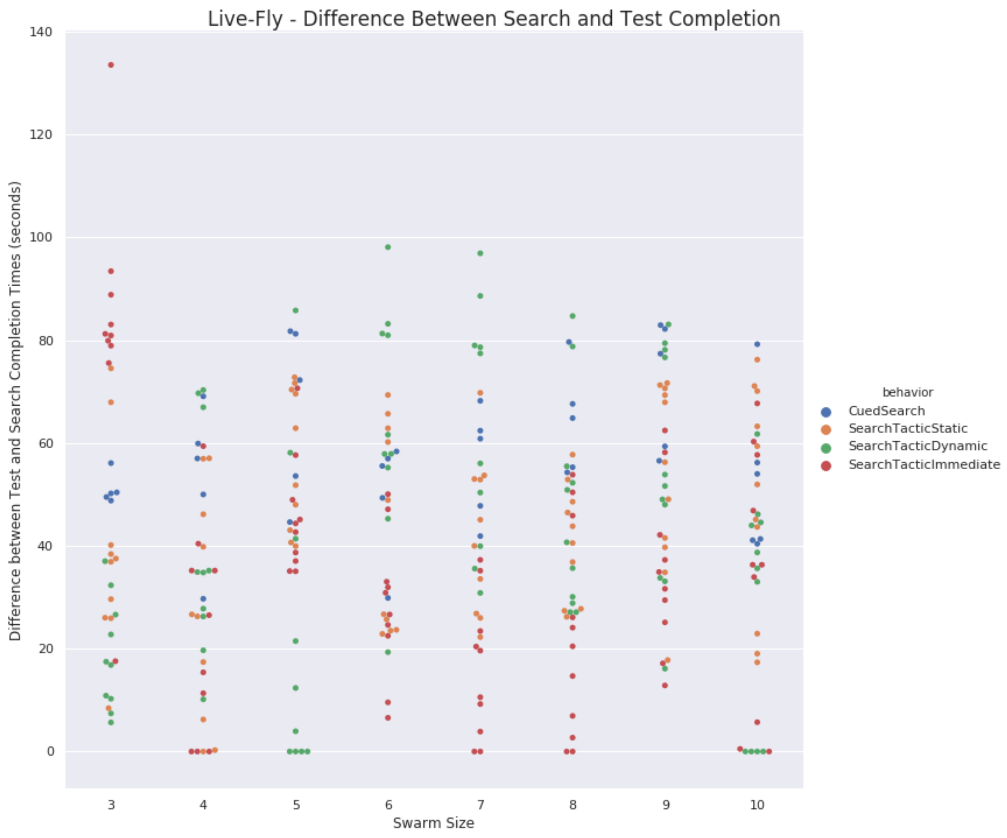

Figure 4.7. Live-Flight Difference Between Search and Test Completion Times

All vehicles start in a searcher role since there are no pounce tasks available at the beginning of a test. After the initial auction the `CuedSearch` and `Dynamic` tactics both require a vehicle to complete a task before changing roles, so each quadcopter will complete at least one search task. In most cases, few additional search tasks will be completed by quadcopters due to the auction utility function's prioritization of pounce tasks. `Immediate`, on the other hand, is not required to complete an in-progress task before changing roles. As a result, the pounce ratio for quadcopters is expected to be higher because of search tasks that are dropped in favor of pounce tasks.

Finally, the expectation for `Static` is for quadcopters to complete almost all pounces and fixed wing UAVs to complete all searches. The `Static` logic does allow fixed wing vehicles to compete for pounce tasks if all search tasks are completed before all pounce tasks have

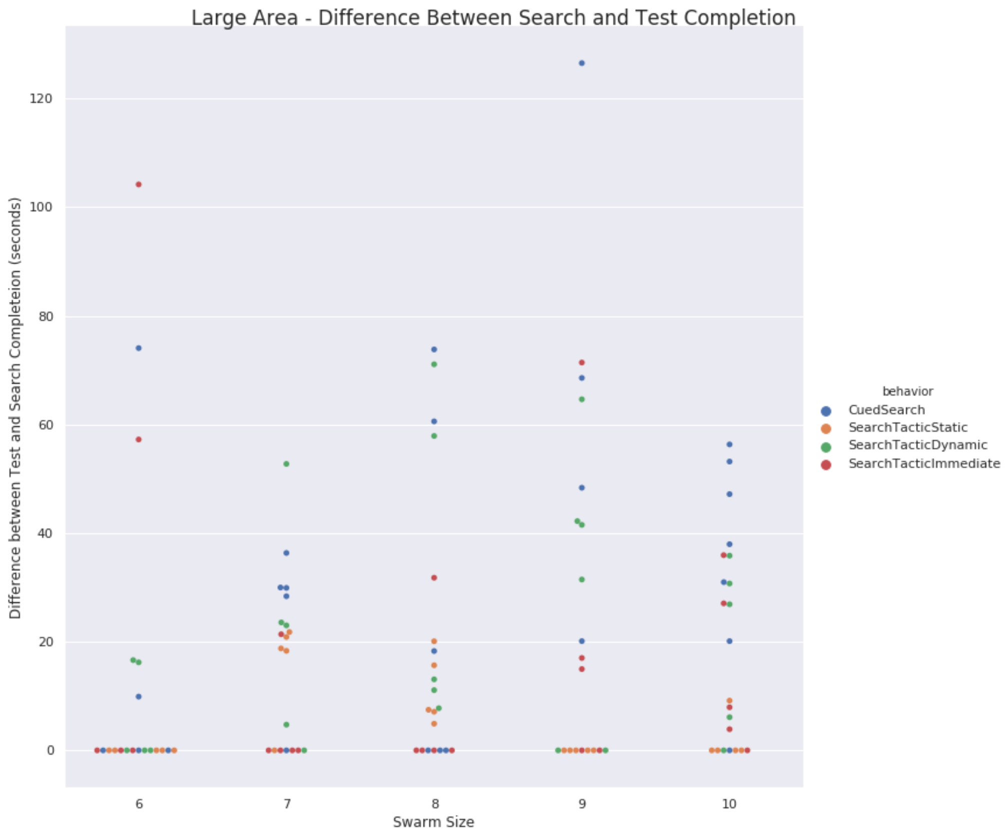

Figure 4.8. Large Area Difference Between Search and Test Completion Times

been accounted for. This only occurred in the live-flight area with a swarm size of three where a total of nine pounce tasks were completed by fixed wing vehicles over 10 tests. Given this, further task allocation analysis of this tactic will not be presented.

Large Area Task Distribution

Raw task completion statistics for the assessed behaviors in the large area are presented in Figure 4.10. This figure contains a chart for each behavior with the total count of each type of task completed binned by swarm size then airframe.

Quadcopter pounce ratios for the large search area are provided in Table 4.4. The pounce ratio for the `Dynamic` tactic was six percent higher than for the baseline `CuedSearch`. This result supports the experiment expectation that the `Dynamic` tactic results would align with

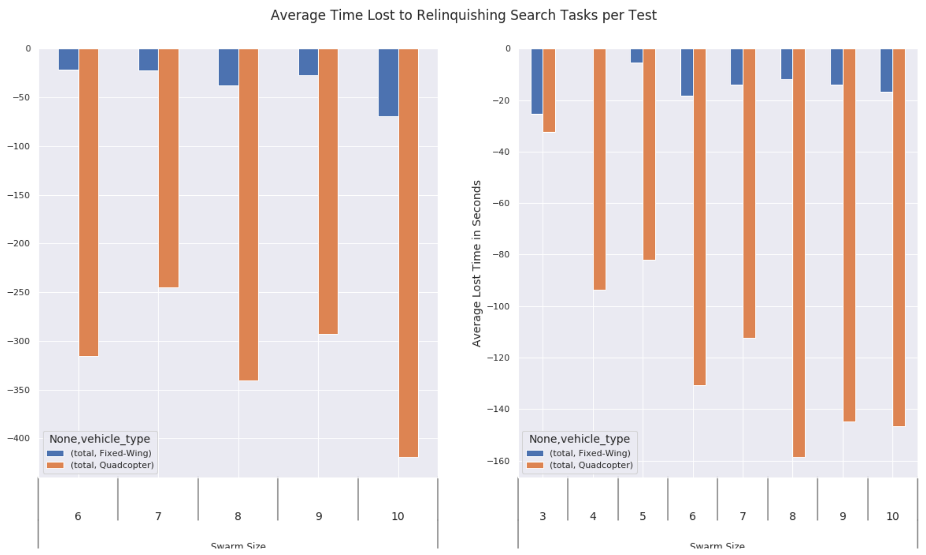

Figure 4.9. Average Time Lost due to Relinquishing Search Tasks.

those of the `CuedSearch` play.

Table 4.4. Quadcopter Pounce Ratio for the Large Area.

Swarm Size	**CuedSearch**	**SearchTacticDynamic**	**SearchTacticImmediate**
6	57%	67%	72%
7	60%	69%	87%
8	62%	65%	76%
9	61%	67%	84%
10	58%	61%	72%
Total	**60%**	**66%**	**78%**

The `Immediate` tactic performed as predicted as well, allowing quadcopters to drop search tasks and transition to pounce tasks as required. This resulted in the number of search tasks completed by quadcopters being significantly lower than with the other behaviors. Fixed wing vehicles, on the other hand, were rarely called upon to execute pounce tasks (see Figure 4.10). Overall this behavior's quadcopters exhibited a pounce rate of 78 percent, well below the 100 percent of `Static` but well above that of the other two behaviors.

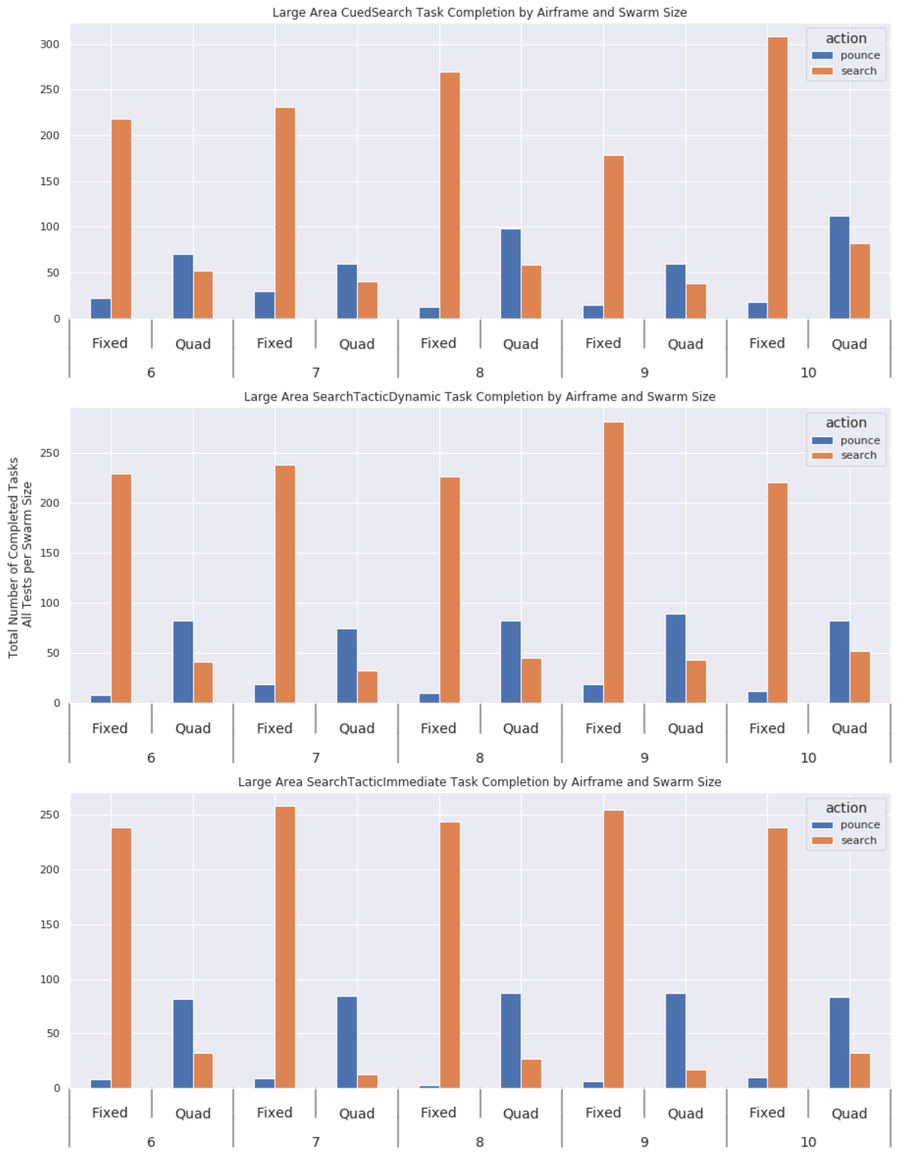

Figure 4.10. Large Search Area Raw Task Distribution.

Live-Flight Area Task Distribution

While results from the large search area yielded expected metrics for task distribution, results from behaviors executed in the live-flight search area exhibited some counterintuitive results. As indicated in Figure 4.11, for swarm sizes 8 - 10 the `Dynamic` tactic saw the number of pounce tasks completed by quadcopters decrease and the number of searches increase. This can be reasoned, however, as there are only 12 search cells available in this search area. This leads to 75 percent or more of the search cells being completed as a result of the first auction with such a high number of starting vehicles. This increases competition for any generated pounce tasks, resulting in available fixed-wing overcoming the airframe utility modifier due to proximity and availability across vehicles. Any fixed-wing reassigned to a pouncer role would then remain a pouncer skewing the task assignment as described.

Quadcopter pounce ratios for the live-flight area tests are provided in Table 4.5. For the baseline `CuedSearch` play, the pounce ratio was 53 percent or lower for swarms of at least eight UAVs and 61 percent or higher for smaller swarms. Despite this discontinuity, the overall pounce ratio for the behavior only differed by one percent compared to the larger area results. As the table indicates, the small-swarm `CuedSearch` results in the live-flight area aligned well with the results from the larger area, while vehicle saturation (i.e., vehicles to tasks) skewed the results for larger swarms.

Table 4.5. Quadcopter Pounce Ratio for the Live-Flight Area.

Swarm Size	**CuedSearch**	**SearchTacticDynamic**	**SearchTacticImmediate**
3	60%	0%	96%
4	61%	59%	97%
5	70%	44%	100%
6	59%	66%	91%
7	61%	59%	84%
8	53%	39%	89%
9	53%	34%	88%
10	51%	30%	82%
Total	**59%**	**41%**	**91%**

Quadcopter pounce ratios from the `Dynamic` tests differed significantly between search areas. Overall, the `Dynamic` pounce ratio in the live-flight area was 22 percent lower than

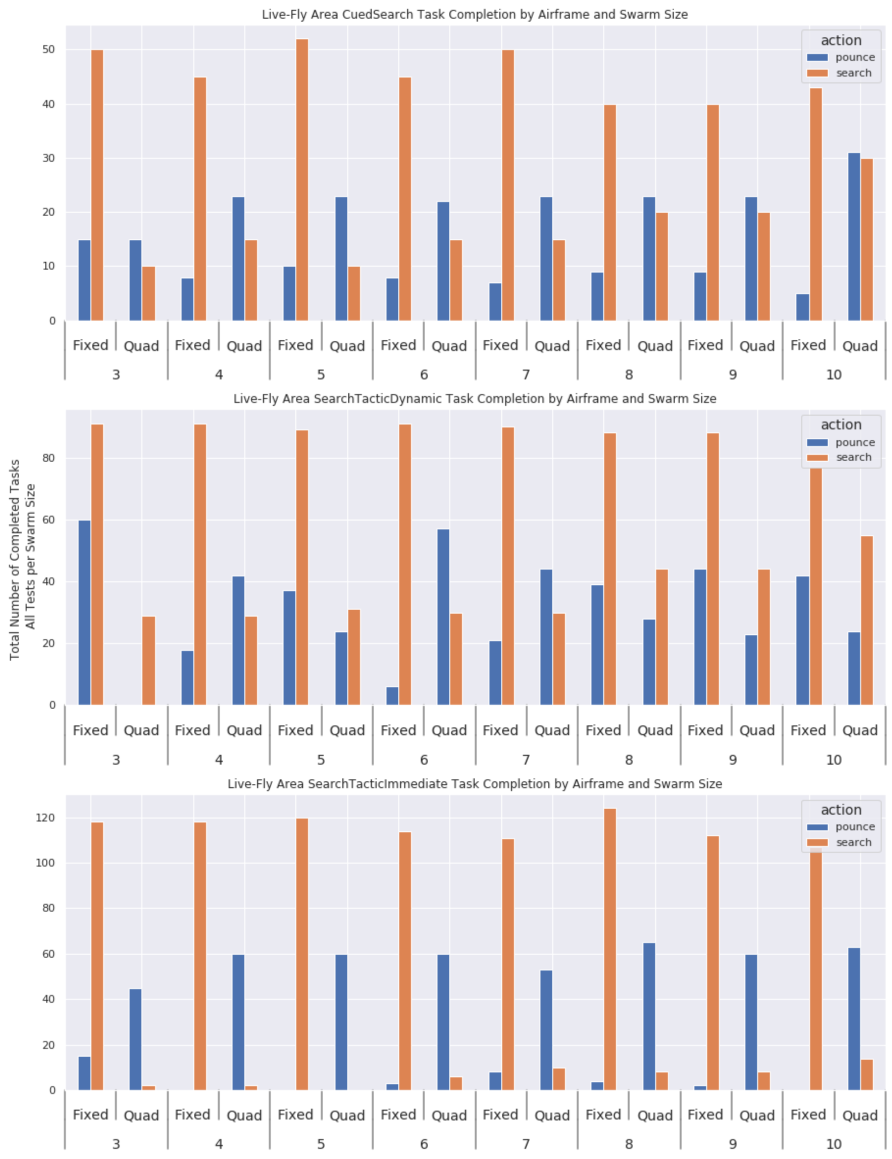

Figure 4.11. Live-Flight Search Area Raw Task Distribution

in the large area. This was largely due to the average quadcopter pounce rate of 35 percent for swarm sizes of between eight and 10 UAVs as explained in the first paragraph of this section. Pounce ratios for swarms of four to seven UAVs were only slightly lower in the live-flight area. Interestingly, no pounce tasks were completed by quadcopters for tests with swarm sizes of three. This can be attributed to the swarm composition and tactic algorithm since there was only one quadcopter in that size swarm. Depending on when the contact was reported during the quadcopter's ongoing search task a fixed wing may have been better positioned in time and space to win the pounce task, and then remain the sole pouncer if more pounce tasks were made available.

Quadcopter pounce ratios for the `Immediate` behavior were minimally affected by the saturation of vehicles. The overall pounce ratio of 91 percent was 13 percent higher than the large area pounce ratio. This can be attributed to the rate at which contacts were generated by fast-moving fixed-wing UAVs relative to the speed with which they were investigated by the slower quadcopters (i.e., quadcopters were likely to relinquish a search area and move from contact to contact rather than resume a searcher role). In general, the high pounce ratios indicate that quadcopters relinquished searches as intended in order to immediately take on higher priority pounce tasks.

4.3.3 Vehicle Utilization

Analysis of vehicle utilization further illustrates the similarities between the `CuedSearch` play and `Dynamic` tactic and highlights the performance differences between these behaviors and the `Immediate` and `Static` tactics. This section discusses two metrics: *total utilization* and *task utilization*. Total utilization measures the time a vehicle is completing a task or transiting to complete a task relative to the total test time. Task utilization is similar but does not include transit time in the measurement. Together, the two measurements are used to differentiate idle time, time spent transiting, and time actively spent searching or pouncing.

Baseline `CuedSearch` Play and `SearchTacticDynamic` Tactic Total Utilization

Figures 4.12 and 4.13 depict total utilization with individual charts representing specific behaviors (columns) for each search area (rows). Each chart plots the calculated utilization of each vehicle across test for each swarm size. The colors for each airframe illustrate distinct patterns of the utilization metric across the behaviors.

The baseline CuedSearch total utilization shows distinct groupings of the fixed-wing and quadcopter vehicles for both search areas with quadcopters maintaining a higher total utilization ratio than fixed wing vehicles. The live-flight search area results display a gradual decrease in total utilization accompanied by an increase in the range of utilization values as the swarm size increases. This aligns with the previous observation regarding vehicle saturation. For CuedSearch the fixed wing vehicles have a low total utilization in the large swarms as they will idle after completing all the search cells.

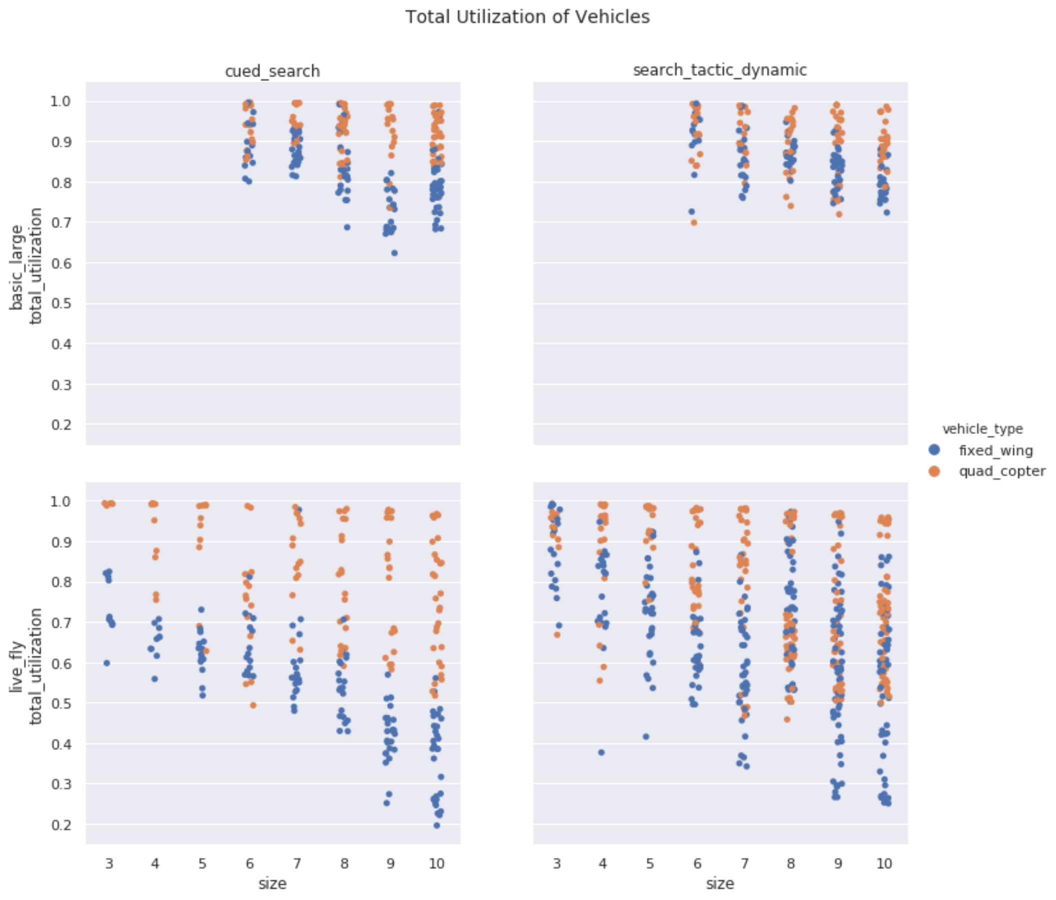

Figure 4.12. Vehicle Total Utilization.

A key distinction visible between the CuedSearch and Dynamic tactics in the figure 4.12 live-flight row is the disappearance of distinct vehicle groupings with the range of fixed wings total utilization measurements increased and interspersed among the higher utilization quadcopter vehicles. The difference appears to be facilitated entirely by fixed wing variation in utilization, with the quadcopter's maintaining a similar total utilization range between the two tactics. When coupled with the known decrease in pounce rate for the quadcopters in the `Dynamic` this observation suggests that the increase in utilization for the fixed wing vehicles comes at the cost of completing more pounces.

`SearchTacticImmediate` and `SearchTacticStatic` Tactic Total Utilization

Closer examination of `Immediate` in both search areas reveals distinct differences between the `Dynamic` behavior that accounts for the difference in pounce rate observed in section 4.3.2. Figure 4.13 shows that in the large search the fixed-wing and quadcopter utilization distributions are inverted when compared to `CuedSearch` and `SearchTacticDynamic`. There is still a distinct separation of vehicle groups but the quadcopters maintain the lower total utilization rate in this case. This reversal is also evident in the live-flight area where there is downward shift of 0.17 in average quadcopter total utilization. The lowest quadcopter total utilization rate dips to approximately 0.1 in the worst case occurring in swarm sizes seven and above for this search area. In contrast The lowest rate observed for `Dynamic` quadcopters in this search was 0.46. This drop in total utilization can be attributed to quadcopters dropping searches to pursue pounces. This data suggests that the high pounce rate in `Immediate` comes at a real cost of lower utilization rate for quadcopters.

This utilization cost for increased pounce rates is supported by analysis of the task utilization metric among quadcopters. Figures 4.14 and 4.14 depict the time spent engaged in completing tasks for quadcopters across all behaviors. It is notable that in the live-flight search area that quadcopters executing the `Immediate` behavior have a similar utilization profile to the `Static` behavior which was expected to have low utilization rates due to inactive quadcopters at the start of tests. This similarity is not pronounced in the large search area but the `Immediate` behavior's task utilization rate is 0.13 lower than `Dynamic`. For the live-flight area this difference is a very significant 0.4 and is not confined to the larger swarm sizes, indicating that this cost is unavoidable.

4.3.4 Pounce Delay and Relinquished Search Cost

Measuring pounce delay and time spent on search tasks that were not completed provides a basis for comparison of the developed tactics with regards to contact investigation re-

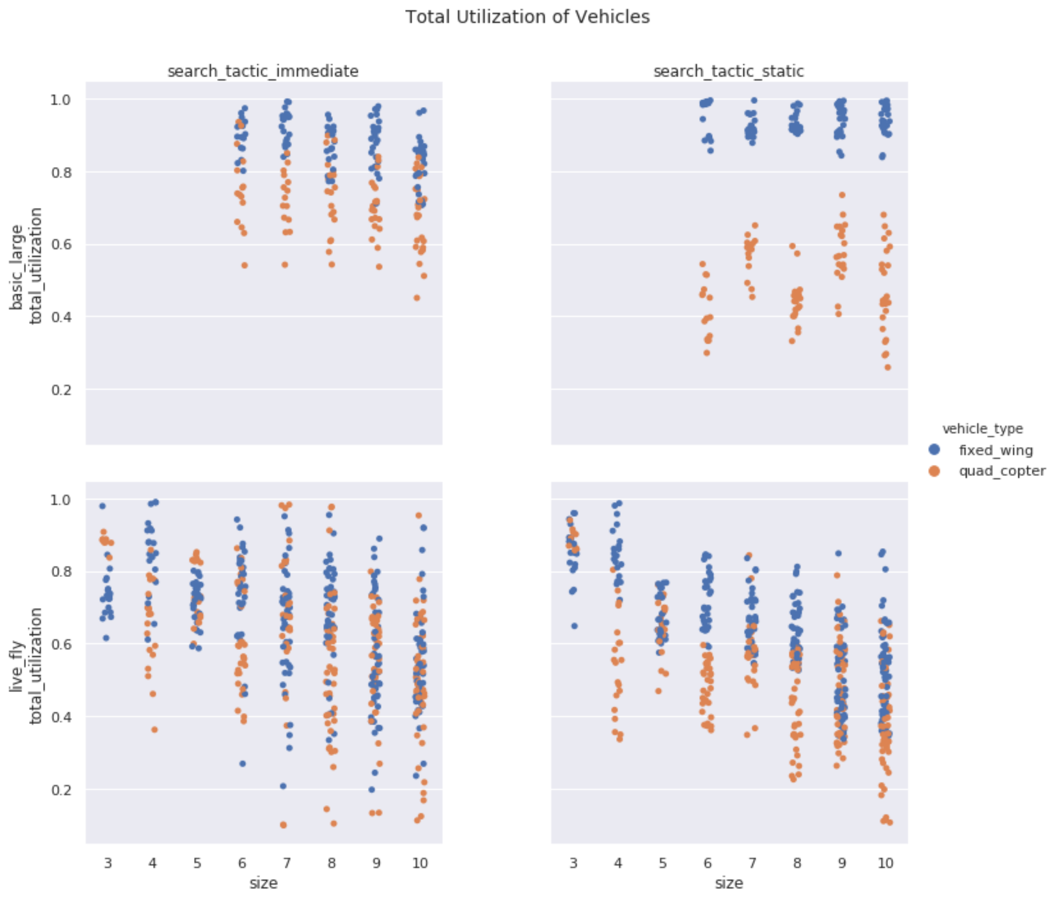

Figure 4.13. Vehicle Total Utilization.

sponsiveness and the costs associated with discontinuing partially completed search tasks. When combined with utilization, this facilitates the assessment of various design choices such as whether or not to relinquish active searches in order to complete pounce tasks more efficiently. Section 4.3.1 detailed the time lost as a result of relinquished searches and assessed its effect on search completion times. This section provides an analysis of the number relinquished tasks to add to the behavioral assessment.

Examination of the number of dropped search tasks in the `Immediate` live-flight search area tests did not reveal any unexpected trends with respect to swarm size or search area. Table 4.6 show the average number of relinquished searches per swarm size. The tests had a small number of dropped searches per test along with slight increases as the swarm size

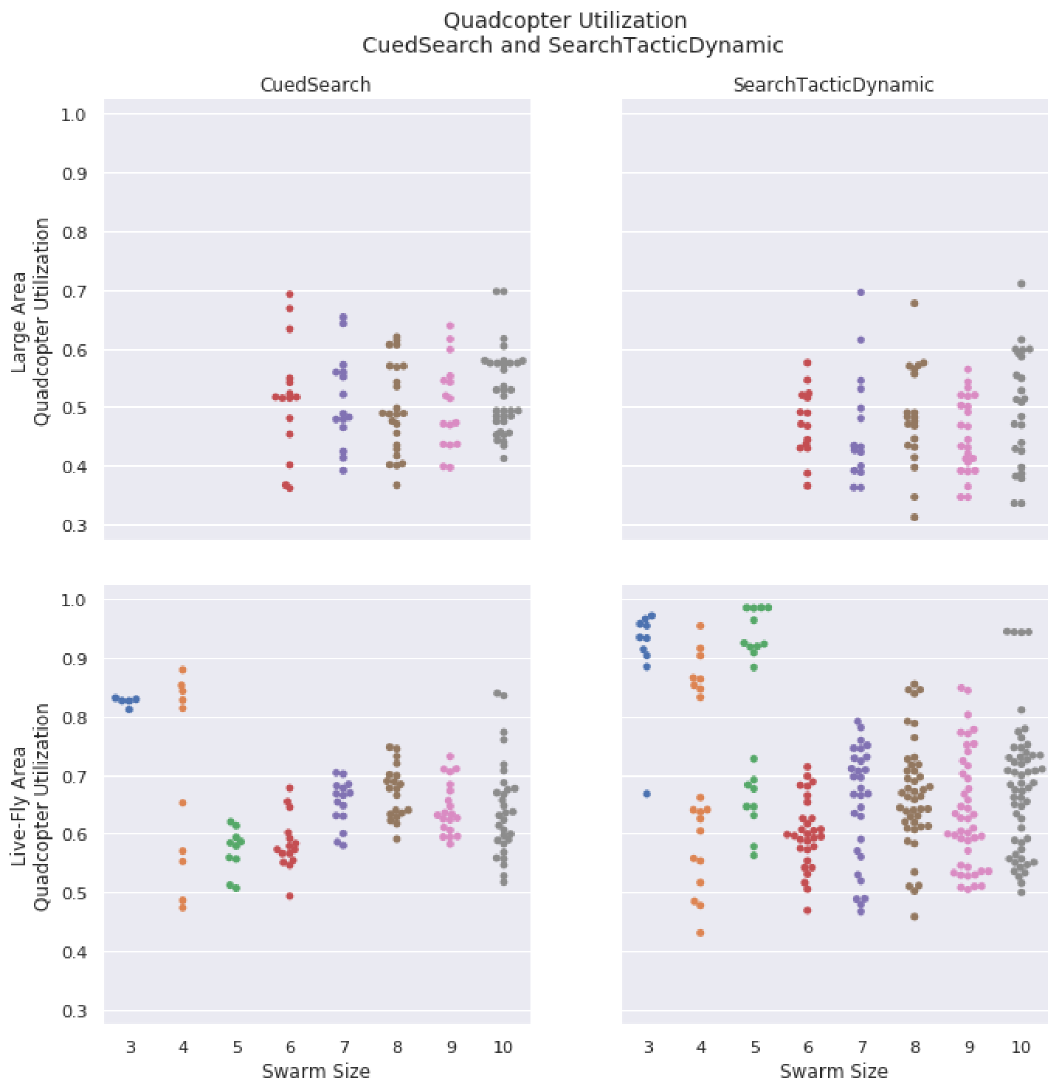

Figure 4.14. `CuedSearch` play and `SearchTacticDynamic` tactic Quadcopter Task Utilization.

increased. This indicates that once a vehicle was assigned to the pouncer role, it rarely transitioned back to a searcher role. This was often because all search tasks were in progress or completed prior to the first assigned pounce tasks being completed. In other cases, additional contacts were generated prior to the completion of the first pounce tasks, so pouncer vehicles stayed in that role.

Notably among live-flight area tests with swarms of three to five UAVs only two searches

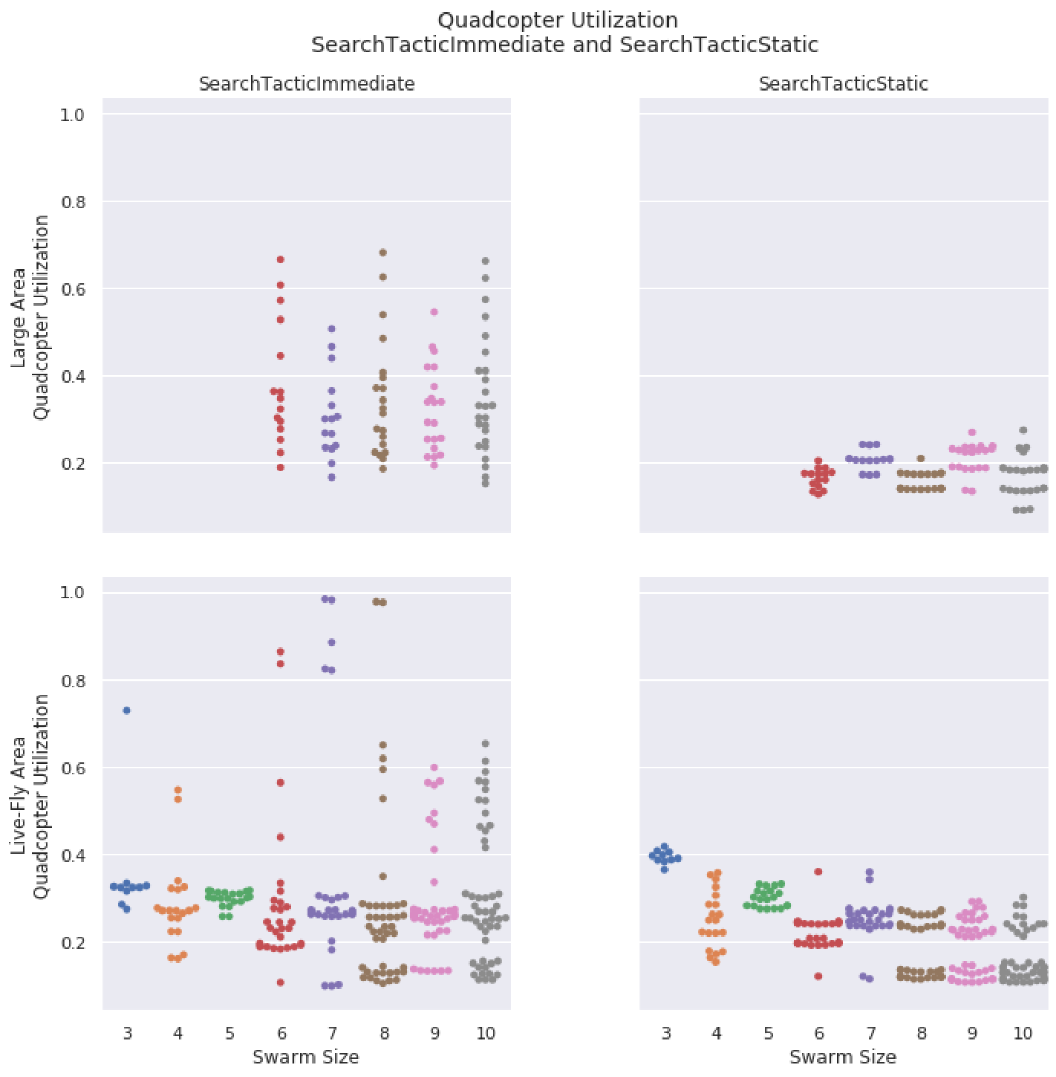

Figure 4.15. SearchTacticImmediate and SearchTacticStatic Quadcopter Task Utilization.

were dropped by fixed-wing UAVs over the course of 30 tests, whereas in swarms larger than six 50 percent of the tests had at least one fixed-wing UAV drop a search task. This result empirically indicates that the task utility function was appropriately prioritizing the quadcopters for pounce tasks.

Tests in the large search area generated contacts at a slower pace resulting in a smaller number of investigation tasks available at any one time. This increased the likelihood that pounce vehicles would transition back to a searcher role upon completion of a pounce task. This would often result in another search task being aborted later when a new contact was eventually generated. The distribution of dropped searches is more varied than the live-flight area with no discernible pattern. It best viewed in Table 4.6 below. The key takeaway is that the quadcopters were much more likely to abandon in-progress searches than fixed-wing UAVs, which is exactly the behavior that the utility function was designed to induce.

Table 4.6. `SearchTacticImmediate` Mean Number of Searches Dropped per Event.

Live-Flight Search Area

Swarm Size	Fixed-Wing	Quadcopter	Total	Standard Deviation
3	0.1	1.9	2.0	0.0
4	0.0	2.9	2.9	0.7
5	0.1	2.0	2.1	0.3
6	0.4	3.0	3.4	0.7
7	0.7	2.5	3.2	0.4
8	0.5	3.4	4.0	0.4
9	0.6	3.2	3.8	0.8
10	0.6	3.6	4.2	0.8

Large Search Area

6	1.8	9.6	11.4	2.9
7	1.4	5.8	7.2	3.1
8	2.6	8.6	11.2	2.9
9	2.6	7.8	10.4	3.2
10	4.4	11.0	15.4	0.9

4.3.5 Contact Response Time

This section investigates the elapsed time from initial contact reporting until completion of the investigation. Expedient contact investigation is a desired outcome for all behaviors and is a key comparison of the baseline `CuedSearch` play to the `Dynamic` tactic. Additionally, the `Immediate` tactic was specifically designed to achieve a faster pounce completion time than `Dynamic` tactic. This improved contact-response time comes with a known cost of lost time from dropping in-progress searches (the average lost time and the negative effect on overall performance was examined in Section 4.3.1).

The average time from the initial contact report to the completion of the pounce for all four behaviors in both test areas is depicted in Figure 4.16. The response time accounts for the time from the contact report until the commencement of the pounce task by the assigned investigator, the transit time to the contact, and a 20 second loiter on top of the target (i.e., the investigation time). As predicted the `Immediate` tactic had lower response times than the baseline `CuedSearch` play and `Dynamic` tactic; however, the fastest response times were observed in the `Static` behavior tests. This was somewhat unexpected since it was hypothesized that the dispersion of the `Immediate` vehicles executing search tasks would result in shorter transits than for the loitering `Static` quadcopters. The results, however, did not bear this out.

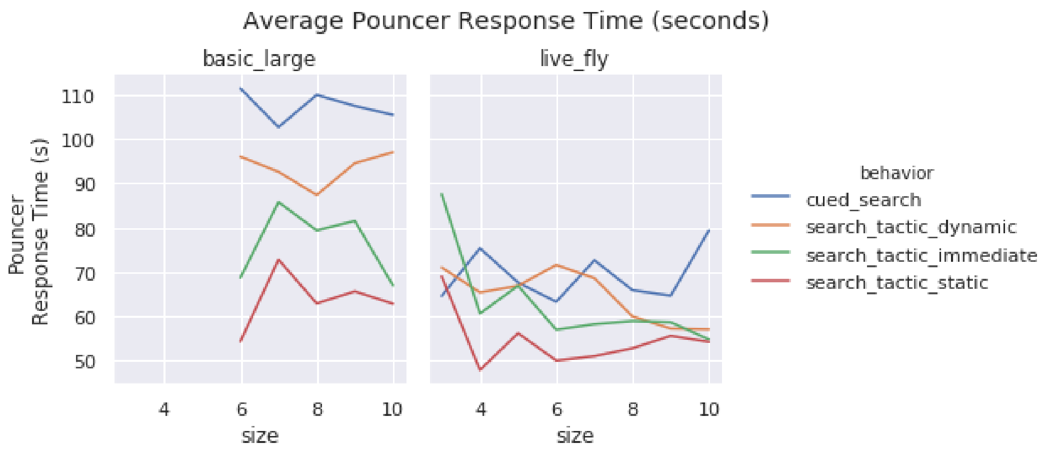

Figure 4.16. Average Pounce Response Time.

4.4 Summary

The experimentation conducted affirms that composed behaviors with auction-based role assignment is a reasonable approach to behavior development that achieves performance characteristics similar to monolithic behaviors with the same functionality. Utilizing the MASC framework to compose behaviors in this fashion provided flexibility in role assignment, composition of plays, and task allocation methods.

Results characterized the performance of the `SearchTacticStatic`, `SearchTactic`

Dynamic, and SearchTacticImmediate tactics and compared them to the baseline CuedSearch play. In general the two auction based tactics, SearchTacticDynamic and SearchTacticImmediate were shown to be comparable to the baseline behavior but enabled the consideration of trade-offs associated with specific design objectives. The SearchTacticDynamic tactic performed similarly to the CuedSearch play and the SearchTacticImmediate tactic achieved the desired decrease in pounce response time at the cost of less efficient search. The SearchTacticStatic tactic exceeded expectations with search times competitive with the baseline behavior and pounce response on par with or better than those of the SearchTacticImmediate tactic despite lower overall total UAV utilization.

As expected, the SearchTacticDynamic tactic behaved most similarly to the CuedSearch play with regards to total test time, search time, total utilization, and pounce rates. Tactic pounce response times were somewhat lower than those of the play but were closer than for either of the other tactics. This is overall performance was expected since the behavioral semantics of the SearchTacticDynamic tactic were modeled on those of the CuedSearch play.

The SearchTacticImmediate tactic prioritized pounces by relinquishing searches to immediately commence pounces. This resulted in an increase in the time required to complete the search area when compared to the other tactics; however, this did not always equate to a higher total test times. The desired improvement in pounce response time was achieved when compared to the SearchTacticDynamic tactic and CuedSearch; however, it did not have faster pounce times than SearchTacticStatic tactic. With respect to pounce rates, the SearchTacticImmediate tactic was more comparable to the SearchTacticStatic tactic than the other two to behaviors and also displayed traits resembling those of the SearchTacticStatic tactic with regards to utilization by vehicle type. Overall the SearchTacticImmediate tactic achieved its primary goal of improving pounce times over the SearchTacticDynamic tactic and the CuedSearch play.

Notable observations outside the primary predictions included probable sensitivity to search area size relative to the ratio of swarm size and available tasks as observed in Sections 4.3.2 and 4.3.4. It was also observed that the SearchTacticStatic tactic test completion times were substantially faster than any other behavior for nine- and 10-UAV swarms and also displayed significantly less variance than other tests.

Given these observations the behavior of choice for prioritizing pounce times in this test environment turned out to be the `SearchTacticStatic` behavior, which was also the simplest implementation within the MASC framework. However, the results presented in this chapter confirm the utility and flexibility of the MASC framework in developing robust swarm behaviors that can meet desired characteristics such as favoring utilization over speed and prioritizing UAV types for specific tasks through selection of role assignment, and task allocation methods to achieve specific goals.

Overall conclusions and suggestions for follow-on research are provided in Chapter 5.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusion

This research demonstrated the viability of auction-based task allocation and behavior composition within the hierarchical ARSENL framework to successfully direct a large UAV swarm in the completion of a complex mission. Our work expanded on recent efforts documented in [37] and [5] that demonstrated the use of single-item auctions to conduct MRTA within a heterogeneous swarm. We surmise that this approach to composability will lead to flexibility in swarm behavior design and allow for creation of complex behaviors that meet the unique requirements of military mission planning, C2, and execution.

Three tactics were developed with different role allocation methods. Experimentation compared these behaviors to one another and to a more monolithic behavior. Two of the tactics utilized an additional single-item auction to effect role assignment, while the third used static assignments determined upon initiation of the tactic. All three tactics were tested in live-flight and virtual environments with various swarm sizes and compositions within two geographic search areas to observe key performance characteristics. The remainder of this chapter summarizes the research findings and recommends future work.

5.1 Summary of Findings

The composition of plays into tactics with integrated task allocation techniques was shown to be a viable alternative to monolithic implementation. Development of behaviors in this compositional style shows excellent potential for reuse and rapid integration with the addition of standardized supporting classes and methods.

5.1.1 Performance Measures

This research used four key measurements for assessment of behavioral performance: completion time, response time, utilization, and pounce ratio. Of those, the utilization and pounce ratio were the most useful in identifying performance differences among tactics. Completion times provided confirmation that there were no adverse performance differences but did not provide enough differentiation to identify superior or inferior performance.

In the case of utilization, examination by airframe provided a distinct differentiation in how efficiently the tactics employed each type of vehicle. In particular, this metric provided the first metric confirming that the `SearchTacticDynamic` tactic was comparable to the `CuedSearch` play.

Pounce ratio was the most important metric in distinguishing the performance of the tactics' role allocation methods. As discussed in Section 3.3.2, this is arguably the most important portion of a tactic's control implementation. Effective comparison of various approaches, then, will be an important aspect of any future work in this area. One specific finding of note was that quadcopter pounce ratios turned out to vary significantly between geographic areas. This was seen as an indication that role allocation can be sensitive to the ratio of swarm size to task size.

Pounce response times exhibited a surprising distinction between the `SearchTactic Dynamic` tactic and the `CuedSearch` play it was designed to mirror. Specifically, the tactic exhibited lower response times in the large geographic area. This demonstrates an inverse correlation with the pounce ratio observations: the higher `SearchTacticDynamic` pounce ratios equate to lower response times. This is a desired characteristic but did diverge somewhat from the `CuedSearch` play results.

5.1.2 Behavior Composition

We showed that a composition of multiple auction-based plays in a single tactic can be designed to have comparable performance to a single auction-based play conducting the same tasks. The solution treated the assignment of swarm UAVs to the composed behaviors as an additional task allocation problem.

Three tactics, `SearchTacticStatic`, `SearchTacticDynamic`, and `SearchTactic Immediate`, were developed, and each demonstrated comparable performance to the baseline `CuedSearch` play. The `SearchTacticDynamic` tactic was the most similar across all observed metrics. This was the desired outcome as the `SearchTacticDynamic` tactic semantics were designed to most closely resemble those of the `CuedSearch` play.

Other measures, including completion, response times, and utilization showed acceptable performance in accordance with the intended design. One surprising result was the overall performance of the `SearchTacticStatic` tactic. In several cases it performed on par with or better than the auction-based tactics. This is likely not a result of the composition or tactic design, but more likely that the perceived relation between utilization and completion time is not strongly correlated for this behavior as initially hypothesized.

Overall, the performance of the developed tactics met the goal of the research to confirm that composition of behaviors and MRTA techniques is a viable alternative to monolithic behavior implementations.

5.1.3 Composition Benefits

The expectation for composed behaviors is that the benefits of reusability lead to faster integration and an ability to develop increasingly complex behaviors. This research supports that assumption with the caveat that key support mechanisms must be implemented and available to enable interaction between the composed plays and the tactic. In particular, well-defined mechanisms for passing play-specific information between hierarchical levels are required.

The current ARSENL code design using the `PlugInBehavior` class supported the exogenic style composition without issue. Chapter 3 noted that the `SearchTacticStatic` tactic implementation was surprisingly straightforward after the supporting classes and methods were developed. The remaining two tactics required adjustment to the role assignment methods that was not burdensome, even with the addition of the role-assignment auction mechanism. Given this and the acceptable level of performance noted in Section 5.1.2, the composition does support usability with the benefit of rapid reuse and integration. With careful design of supporting framework methods, the benefit of creating even more complex behaviors is likely achievable as well.

5.2 Future Work

The classes and methods developed to support the composition for the created tactics were specific to the plays and auction components used. However, the key issues that they solve can be abstracted and addressed more comprehensively to enable a wider set of interactions that promote the goals of component reuse and ease of integration. The MASC framework provides a viable hierarchy, and the ARSENL codebase provides the tools for implementing swarm behaviors. An updated solution stemming from this research will provide the tools necessary for repeatable, consistent composition.

5.2.1 Supporting Framework

There are four themes that the implemented classes and methods supported for the composition: standard tactic and play communication mechanisms, decoupling and encapsulation of state data, data-driven event generation, and event and data notification requirements. Ultimately, this is purely an implementation detail to support the goals of usability and rapid integration.

Principle among the identified needs is the synchronization of tactic and play knowledge onboard the individual UAV. Chapter 2 discussed several examples of supporting shared knowledge within compositional models. Ultimately this implementation used container classes and event-based methods to enable the required synchronization. The container classes added required members and methods to enable state manipulation and provide observer members to enable callbacks to the tactic.

The event-observer mechanism was chosen because of the need for a tactic to intercept task state changes within the play in order to initiate immediate role transfers (i.e., before the play actually transitions to a new state). However, immediacy is not a general requirement, and changes to the codebase could eliminate most situations that currently require event call backs to effect tactic-play interaction. For general synchronization between tactic and play, standard programming patterns relying on method arguments and return values might provide a less complex and decoupled option.

Within the ARSENL architecture, utilizing the `maneuver_command` control loop is arguably the most direct way to implement composition. Extending this method to provide arguments to the play and return play-specific information can be accomplished in multiple ways. Standardizing the data-passing mechanism among plays with disparate functionality will be the greater challenge.

The most important information that was coordinated in this research was was a byproduct of states and events associated with tasks, auctions, and maneuver. Chapter 3 described the components utilizing the state information. The `Role` and `Auctioneer` classes were developed to provide access to and manipulation of state data. They were, however, implementation specific with the `Role` class in particular having only moderate potential for reusability. Abstracting the requirements that the `Role` and `Auctioneer` classes satisfied into virtual classes defining state, tasks, and task allocation would provide the foundation for more generalizable reusability. Designing these classes with cooperation specifically in mind might also align required members and methods to implement the chosen information synchronization mechanism.

5.2.2 Role Assignment Methods

Further research into enhanced or alternate role assignment methods within a tactic presents the greatest opportunity for reuse of the technique. The implementation in this research replicates the functionality of a single auction. This basic implementation can be modified relatively easily to support a wider range of auction scenarios. Specifically, this would involve refining the transition task definition to account for all roles and implementing a more robust function to calculate the optimal number of each role assignment. This model would support the ability to initiate an auction for any role at any time it determines that additional support is needed. Defining a standard for this role assignment class would also support integrating other MRTA methods.

5.2.3 Decoupling Auctions and Plays

The composition of plays containing their own MRTA functions, such as an auction, is advantageous in that plays can be built with the ideal MRTA solution for the play-specific objectives. One drawback that was encountered during this research was the need to synchronize tasks at multiple hierarchical levels. This synchronization might be alleviated by defining only the requirement for an auction within the play but having the tactic actually conduct the auction to service the requirement.

Conducting auctions at the tactic level would potentially enable more granular design of the tactic. Implementing this design choice, for instance, might allow for an exact duplication of the `CuedSearch` play semantics by a tactic rather than the approximation of the `SearchTacticDynamic` tactic. The `ContactInvestigation` play provides a good option for implementing this design since tasks are received from another play or another UAV; that is, the auctions are already dependent on external events. The `SwarmSearch` play would require exposing its geographic decomposition methods (i.e., its means of dividing the search area into search cells). Standardizing the task generation would be essential to implementing this approach. Nevertheless, it might provide a generalizable approach to composition that has broad benefits.

5.2.4 Comprehensive Tactic Control

Developing a tactic control framework to manage compositions incorporating multiple role assignment strategies and task allocation methods that support the military mission requirements is a recommended long-term step to allow tactics to achieve robust C2 and usability. Developing a tactic control framework is distinct from role assignment and MRTA methods in that it aggregates distinct requirements for mission completion based on parameters relevant to specific mission time and space. A tactic may need to allocate an exact number of vehicles for an enduring task, form a coalition for a ST-MR-IA task, and then apportion the remainder of vehicles between roles via a particular MRTA mechanism. Thus, this functionality forms an additional layer of aggregation where the control framework assigns plays to a list of role assignment methods for which it specifies the priority, vehicle types, and allocation constraints for each role assignment method. This description could arguably be the thought of as the mission level in the MASC framework, but it does not correlate directly to an object in the ARSENL codebase.

5.3 Summary

This work demonstrated that the creation of tactics as a composition of plays and the integration of task allocation techniques is a viable alternative to monolithic UAV swarm behavior implementation. While the recreation of performance metrics was not perfect, there is ample opportunity to refine role allocation method to improve desired performance characteristics and implement more complex behaviors. Development of behaviors in this compositional style shows excellent potential for reuse and rapid integration with the addition of standardized supporting classes and methods.

List of References

[1] J. Arquilla and D. F. Ronfeldt, *Swarming & the Future of Conflict*. Santa Monica, CA: Rand Corporation National Defense Research Institute (U.S.), 2000.

[2] P. Scharre, "Robotics on the Battlefield Part II: The Coming Swarm," *Center for a New American Security*, p. 68, Oct. 2014.

[3] J. Hurst, "Robotic Swarms in Offensive Maneuver," *Joint Forces Quarterly*, vol. 87, no. 4, pp. 105–111, 2017.

[4] K. Giles, "Mission-based Architecture for Swarm Composability," Ph.D. dissertation, Naval Postgraduate School, Monterey, CA, 2018.

[5] B. J. Campbell, "Autonomous Cueing within Heterogeneous Robot Swarms," Master's thesis, Naval Postgraduate School, Monterey, CA, 2019.

[6] "Unmanned Systems Integrated Roadmap FY2017-2042," Aug. 2018.

[7] D. T. H. Chung, "OFFensive Swarm-Enabled Tactics (OFFSET)," Mar. 2021.

[8] C. Korpela, M., "Swarms in the Third Offset," United States Army War College, Carlisle, PA, Strategy Research Project, Jan. 2017.

[9] "The Low-Cost Unmanned aerial vehicle Swarming Technology (LOCUST)," 2015.

[10] "Department of Defense Announces Successful Micro-Drone Demonstration," https://www.defense.gov/News/Releases/Release/Article/1044811/department-of-defense-announces-successful-micro-drone-demonstration/, Jan. 2017.

[11] "In the Sky and on the Ground, Collaboration Vital to DARPA's CODE for Success," https://www.darpa.mil/news-events/2019-03-22, Mar. 2019.

[12] A. R. Smith, "Simple Computation-Universal Cellular Spaces," *Journal of the ACM*, vol. 18, no. 3, pp. 339–353, July 1971.

[13] G. Beni, "The concept of cellular robotic system," in *Proceedings IEEE International Symposium on Intelligent Control 1988*. Arlington, VA, USA: IEEE Comput. Soc. Press, 1989, pp. 57–62.

[14] G. Beni and J. Wang, "Swarm Intelligence in Cellular Robotic Systems," in *Robots and Biological Systems: Towards a New Bionics?*, P. Dario, G. Sandini, and P. Aebischer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 703–712.

[15] G. Beni, "From Swarm Intelligence to Swarm Robotics," in *Swarm Robotics*, vol. 3342, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, E. Şahin, and W. M. Spears, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–9.

[16] A. F. T. Winfield, C. J. Harper, and J. Nembrini, "Towards Dependable Swarms and a New Discipline of Swarm Engineering," in *Swarm Robotics* (Lecture Notes in Computer Science), E. Şahin and W. M. Spears, Eds. Berlin, Heidelberg: Springer, 2005, pp. 126–142.

[17] L. B. Rainey and M. Jamshidi, Eds., *Engineering Emergence: A Modeling and Simulation Approach*. Boca Raton: CRC Press, Aug. 2018.

[18] C. W. Reynolds, "Flocks, Herds and Schools: A Distributed Behavioral Model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, p. 10, July 1987.

[19] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[20] J. Kennedy and R. Eberhart, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Nagoya, Japan: IEEE, 1995, pp. 39–43.

[21] M. Senanayake, I. Senthooran, J. C. Barca, H. Chung, J. Kamruzzaman, and M. Murshed, "Search and tracking algorithms for swarms of robots: A survey," *Robotics and Autonomous Systems*, vol. 75, pp. 422–434, Jan. 2016.

[22] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm Robotic Behaviors and Current Applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, Apr. 2020.

[23] B. Khaldi and F. Cherif, "An Overview of Swarm Robotics: Swarm Intelligence Applied to Multi-robotics," *International Journal of Computer Applications*, vol. 126, no. 2, pp. 31–37, Sep. 2015.

[24] J. Harwell and M. Gini, "Improved Swarm Engineering: Aligning Intuition and Analysis," *arXiv:2012.04144 [cs]*, Oct. 2021.

[25] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE International Conference on Robotics and Automation*. St Paul, MN, USA: IEEE, May 2012, pp. 3293–3298.

[26] K. B. Giles, D. T. Davis, K. D. Jones, and M. J. Jones, "Expanding domains for multi-vehicle unmanned systems," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2021, pp. 1400–1409.

[27] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, Dec. 2012.

[28] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2004, pp. 2149–2154.

[29] "Using Gazebo Simulator with SITL — Dev documentation," https://ardupilot.org/dev/docs/using-gazebo-simulator-with-sitl.html?highlight=swarm.

[30] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. Suárez Fernández, and P. Campoy, "A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 683–709, Dec. 2017.

[31] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human Interaction With Robot Swarms: A Survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, Feb. 2016.

[32] P. Beautement, D. Allsopp, M. Greaves, S. Goldsmith, S. Spires, S. G. Thompson, and H. Janicke, "Autonomous Agents and Multi –agent Systems (AAMAS) for the Military – Issues and Challenges," in *Defence Applications of Multi-Agent Systems*, vol. 3890, S. G. Thompson and R. Ghanea-Hercock, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–13.

[33] B. J. Strawser, Ed., *Killing by Remote Control: The Ethics of an Unmanned Military*. Oxford ; New York: Oxford University Press, 2013.

[34] R. Arnold, K. Carey, B. Abruzzo, and C. Korpela, "What is A Robot Swarm: A Definition for Swarming Robotics," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, Oct. 2019, pp. 0074–0081.

[35] L. Vig and J. A. Adams, "Coalition Formation: From Software Agents to Robots," *Journal of Intelligent and Robotic Systems*, vol. 50, no. 1, pp. 85–118, Sep. 2007.

[36] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Cooperative Robots and Sensor Networks 2015*, vol. 604, A. Koubâa and J. Martínez-de Dios, Eds. Cham: Springer International Publishing, 2015, pp. 31–51.

[37] M. Hopchak, S., “Autonomous Decision in Multi-Robot Systems,” Master’s thesis, Naval Postgraduate School, Monterey, CA, 2018.

[38] T. Yasuda and K. Ohkura, “Collective Behavior Acquisition of Real Robotic Swarms Using Deep Reinforcement Learning,” in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, Jan. 2018, pp. 179–180.

[39] M. Brambilla, E. Ferrante, M. BIRATTARI, and M. Dorigo, “Swarm Intell Swarm robotics: A review from the swarm engineering perspective,” *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[40] R. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[41] L. Parker, “ALLIANCE: An architecture for fault tolerant multirobot cooperation,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, Apr. 1998.

[42] J. McLurkin, D., “Stupid Robot Tricks A Bahavior-Based Distributed Algorithm Library for Programming Swarms of Robots,” Master’s thesis, MIT, Boston, MA, 2004.

[43] C. A. Miller, R. P. Goldman, H. B. Funk, P. Wu, and B. B. Pate, “A Playbook Approach to Variable Autonomy Control: Application for Control of Multiple, Heterogeneous Unmanned Air Vehicles,” in *FORUM 60, the Annual Meeting of the American Helicopter Society*. Baltimore, MD: American Helicopter Society International, Inc, June 2004, vol. 3, pp. 2146–2157.

[44] B. Browning, J. Bruce, M. Bowling, and M. Veloso, “STP: Skills, tactics, and plays for multi-robot control in adversarial environments,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 219, no. 1, pp. 33–52, Feb. 2005.

[45] P. K. Davis and R. H. Anderson, “Improving the Composability of DoD Models and Simulations,” *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 1, no. 1, pp. 5–17, Apr. 2004.

[46] R. G. Bartholet, D. C. Brogan, P. F. Reynolds Jr., and J. C. Carnahan, “In Search of the Philosopher’s Stone: Simulation Composability Versus Component-Based Software Design,” in *2004 Fall Simulation Interoperability Workshop*. Orlando, FL: University of Virginia, Department of Computer Science, Sep. 2004.

[47] K.-K. Lau and T. Rana, “A Taxonomy of Software Composition Mechanisms,” in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. Lille, TBD, France: IEEE, Sep. 2010, pp. 102–110.

[48] K.-K. Lau and M. Ornaghi, “Control encapsulation: A calculus for exogenous composition of software components,” in *International Symposium on Component-based Software Engineering* (Lecture Notes in Computer Science), G. A. Lewis, I. Poernomo, and C. Hofmeister, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5582, pp. 121–139.

[49] F. Zhu, Y. Yao, J. Li, and W. Tang, “Reusability and composability analysis for an agent-based hierarchical modelling and simulation framework,” *Simulation Modelling Practice and Theory*, vol. 90, pp. 81–97, Jan. 2019.

[50] E. W. Weisel, M. D. Petty, R. R. Mielke, V. Modeling, and N. Va, “Validity of Models and Classes of Models in Semantic Composability,” in *Proceedings of the Fall 2003 Simulation Interoperability Workshop*. Orlando, FL: Simulation Interoperability Standards Organization (SISO), Sep. 2003, vol. 9, p. 68.

[51] H. S. Sarjoughian, “Model Composability,” in *2006 Winter Simulation Conference*. Los Alamitos, CA: IEEE Computer Society, Dec. 2006, pp. 149–158.

[52] T. H. Chung, “Advancing autonomous swarm capabilities: From simulation to experimentation,” in *2015 Winter Simulation Conference (WSC)*, Dec. 2015, pp. 2341–2341.

[53] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, Oct. 2013.

[54] B. P. Gerkey and M. J. Matarić, “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems,” *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, Sep. 2004.

[55] O. Shehory and S. Kraus, “Task allocation via coalition formation among autonomous agents,” in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canado, 1995, vol. 1, p. 7.

[56] M. Dias, R. Zlot, N. Kalra, and A. Stentz, “Market-Based Multirobot Coordination: A Survey and Analysis,” *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006.

[57] L. B. Johnson, H. L. Choi, and J. P. How, “The role of information assumptions in decentralized task allocation: A tutorial,” *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, Aug. 2016.

[58] D. P. Bertsekas, “Auction Algorithms,” in *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds. Boston, MA: Springer US, 2009, pp. 128–132.

[59] D. P. Bertsekas, “The auction algorithm: A distributed relaxation method for the assignment problem,” *Annals of Operations Research*, vol. 14, no. 1, pp. 105–123, Dec. 1988.

[60] A. Stentz and M. B. Dias, “A Free Market Architecture for Coordinating Multiple Robots:,” Robotics Institute, Carnegie Mellon University,, Pittsburgh, PA, Technical CMU-RI -TR-99-42, Dec. 1999.

[61] A. Viguria and A. M. Howard, “Probabilistic Analysis of Market-based Algorithms for Initial Robotic Formations,” *The International Journal of Robotics Research*, vol. 29, no. 9, pp. 1154–1172, Aug. 2010.

[62] Smith, “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver,” *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.

[63] C. Daskalakis, “Multi-item auctions defying intuition?” *ACM SIGecom Exchanges*, vol. 14, no. 1, pp. 41–75, Nov. 2015.

[64] L. Vig and J. A. Adams, “Market-Based Multi-robot Coalition Formation,” in *Distributed Autonomous Robotic Systems 7*. Tokyo: Springer Japan, 2006, pp. 227–236.

[65] L. Liu and D. A. Shell, “Multi-Level Partitioning and Distribution of the Assignment Problem for Large-Scale Multi-Robot Task Allocation,” in *Robotics: Science and Systems 2011*, Los Angeles, CA, 2011, p. 8.

[66] J. Guerrero and G. Oliver, “Multi-robot coalition formation in real-time scenarios,” *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1295–1307, Oct. 2012.

[67] M. Irfan and A. Farooq, “Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities,” in *2016 International Conference on Intelligent Systems Engineering (ICISE)*. Islamabad, Pakistan: IEEE, Jan. 2016, pp. 210–215.

[68] R. M. de Mendonça, N. Nedjah, and L. de Macedo Mourelle, “Efficient distributed algorithm of dynamic task assignment for swarm robotics,” *Neurocomputing*, vol. 172, pp. 345–355, Jan. 2016.

[69] H. Kose, U. Tatlıdede, K. Kaplan, and H. L. Akın, “Q-Learning based Market-Driven Multi-Agent Collaboration in Robot Soccer,” in *Proceedings of the Turkish Symposium on Artificial Intelligence and Neural Networks*, June 2004, pp. 219–228.

[70] T. H. Chung, M. R. Clement, M. A. Day, K. D. Jones, D. Davis, and M. Jones, “Live-fly, large-scale field experimentation for large numbers of fixed-wing UAVs,”

in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden: IEEE, May 2016, pp. 1255–1262.

[71] D. T. Davis, T. H. Chung, M. R. Clement, and M. A. Day, "Consensus-based data sharing for large-scale aerial swarm coordination in lossy communications environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea: IEEE, Oct. 2016, pp. 3801–3808.

[72] H. Goldstein and M. J. R. Healy, "The Graphical Presentation of a Collection of Means," *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, vol. 158, no. 1, pp. 175–177, 1995.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California