

# UAV Swarm Intelligence in CEMA

Enhancing Urban Communication Line Management

AE5310: Thesis Control and Operations

S.H.S. (Storm) Holman



Delft University of Technology



# UAV Swarm Intelligence in CEMA

Enhancing Urban Communication Line Management

by

## Storm Holman

to obtain the degree of Master of Science  
at the Delft University of Technology,  
to be defended publicly on Friday, 12 January at 09:30h.

Student number:	4645715	
Project duration:	March 2023 – January 2024	
Thesis committee:	Dr. O.A. Sharpans'kykh	Supervisor
	Dr. B.F. Lopes Dos Santos	Chair
	Dr. G. la Rocca	Examiner
Company supervisor:	Dr. O. den Ouden	Royal NLR

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.





# Acknowledgements

Writing this thesis has felt much like a really long hike in the mountains. There are days when you feel great, having cracked a piece of your case, and other days when things don't go as planned, making it pretty tough. The process of researching in unexplored terrain can be compared to summiting a mountain, only to find out that the mountain is actually a range of many peaks. It's the researcher's decision to choose which peak to climb based on what he or his department finds most relevant and feasible. It is inherent in humans to keep innovating, driven by our boundless ambition. This relentless drive is what propels science forward. However, I've learned that it's crucial for this to happen in an orchestrated manner. What really helps is having awesome people around you – friends, family, and advisors. They are the ones who keep you going when things get rough and are there to celebrate with you when things go right.

I want to extend a special note of gratitude to my supervisors, Dr. A.O. Sharpans'kykh from TU Delft and Dr. O. den Ouden from Royal NLR, for their unwavering support and insightful guidance. Their valuable feedback and the collaborative brainstorming sessions have been fundamental in enhancing the quality of this work. My thanks also go to my colleagues at Royal NLR for their brainstorming sessions, support, and the enjoyable distractions they provided, which were more appreciated than they might realize.

Further, my sincere appreciation is extended to both the institutions of Royal NLR and TU Delft, whose resources and environments have been indispensable in facilitating my research.

Above all, my deepest gratitude is reserved for my family, friends, and especially Daphne. Their constant support and readiness to listen to my discussions about aviation and aircraft have played a crucial role in my journey. I recognize that my current achievements and who I am today are largely due to their unwavering backing and for this, I am profoundly thankful.

Storm H.S. Holman  
Amsterdam, January 2024



# Contents

List of Figures	vii
List of Tables	ix
List of Abbreviations	xi
Introduction	xiii
I Scientific Paper	1
II Literature Study	
previously graded under AE4020	35
III Supporting Work	95
1 Problem Setup	97
1.1 Mission Tasks	97
1.1.1 Transmitter Search	97
1.1.2 Communication Line Search	98
1.1.3 Communication Line Mapping	99
1.1.4 Interference Point Finding	99
1.2 Research Questions	99
1.3 Scope and Assumptions	99
2 Model Elaboration	101
2.1 Agent-Based Model	101
2.2 Model Implementation	102
2.3 Technical Setup	103
2.3.1 Specialized Software: NLR - EW Toolbox	103
2.3.2 Python Libraries	103
2.3.3 Hardware Infrastructure	104
3 Elaboration on Methods	105
3.1 Transmitter Search (Single)	105
3.1.1 Particle Swarm Optimisation Search	105
3.1.2 Pair-based Gradient Search	107
3.2 Communication Line Search & Other Specialized Algorithms	109
3.2.1 Communication Line Search	109
3.2.2 Sobol Sequence	110
3.2.3 One Point Departure Strategy	110
3.3 Communication Line Mapping	111
3.3.1 Frontier Boustrophedon Mapping	111
3.3.2 Frontier Spiral Mapping	112
3.3.3 Multi-agent Q-learning	114
3.4 Interference Point Finding	116
4 Verification & Validation	119
4.1 Verification	119
4.1.1 EW Environment Functions Testing	119
4.1.2 Verification of PSO Algorithm Convergence	120
4.1.3 Verification of Gradient Determination Algorithm	120
4.1.4 Verification of Mapping in Mapping Area	121

4.1.5	Verification of Updates in the MCTS Tree	.121
4.2	Discussion on Validation	.122
4.3	Discussion on Validation	.122
4.3.1	Environment	.122
4.3.2	Operations	.122
4.3.3	Simulation	.123
4.3.4	Self-confidence	.123
5	Specification of Experimental Setups & KPIs	125
5.1	Single Transmitter Search	.125
5.1.1	Experimental Setup: E1	.125
5.1.2	Key Performance Indicators	.126
5.2	Multi Transmitter Search	.127
5.2.1	Experimental Setup: E2, E5	.127
5.2.2	Key Performance Indicators (KPIs)	.128
5.3	Communication Line Mapping & Interference Point Finding	.128
5.3.1	Experimental Setup: E6, E7	.128
5.3.2	Key Performance Indicators (KPIs)	.129
6	Statistical Elaboration	131
6.1	Single Target Search: E1	.131
6.2	Multi Target Search: E5	.134
6.3	Communication Line Mapping: E6	.135
7	Supplementary Experiments & Analysis	139
7.1	Single Transmitter Search	.139
7.2	Multi Transmitter Search	.141
7.2.1	Ma1: Refinement of Utility-function	.141
7.2.2	Ma2: Further Metric Exploration	.142
7.2.3	Comparative Results: E5 - exclusion unsuccessful runs	.143
7.3	Communication Line Mapping & Interference Point Finding	.143
8	Pseudocode	147
	Bibliography	159

# List of Figures

1.1	Visualisation of Mission Flow for UAV swarm in CEMA	97
1.2	Schematic Representation - Global Setup	98
1.3	Schematic Representation - Base Setup	98
2.1	High-level Model Architecture Flow	103
3.1	Visualization of the key PSO agent components: current velocity, social influence, cognitive drive, and random factors $r_1$ and $r_2$ .	106
3.2	The figure illustrates the triangulation method used in the pair-based gradient search. As shown, the follower agent (P2, depicted in green) is positioned at a $90^\circ$ offset from the trajectory of the leader agent (from $P1_{last}$ to $P1$ , depicted in red). This setup ensures non-colinear positioning of sample points, facilitating accurate gradient determination.	107
3.3	Trajectory of the agent using historical positions to circle towards the mapping point.	109
3.4	Comparison between Sobol sequence (left) and Scipy's random number generation (right). Adapted from [8].	110
3.5	One Point Departure Strategy with Intermediate Destinations.	110
3.6	Visualization of the Frontier Boustrophedon Mapping. Highlighting offset distance, turn angles, and the agent trajectory following boustrophedon pattern.	111
3.7	Visualization of the Frontier Spiral Mapping. Showing the inward spiral trajectory and distinct legs of the agent's movement.	112
3.8	Illustration of the initialized grid by the first mapping agent, starting from the mapping point and extending opposite to its historical point. The green area indicates the communication line being mapped.	114
3.9	Illustration showing local mapping statuses (left) and potential actions (right).	114
3.10	Triangular environment used for pre-training.	115
3.11	Communication Line Mapping.	117
3.12	Reinforcement Learning Method	117
4.1	MCTS Decision Nodes $u(\text{value})$ , $v(\text{visits})$ - (U_seed: 23, Agents: 20, Steps per layer: 200000, C: 1)	121
4.2	95% Confidence Interval	123
5.1	20x20 grid cells (Frontier Boustrophedon)	129
6.1	Quantile-Quantile Plots for each KPI	132
6.2	Coefficient of Variation Against Number of Simulations for each KPI	133
6.3	QQ Plot for value score with initiation variation.	134
6.4	QQ Plot for value score with transmitter variation.	134
6.5	CV Plot for varying agent initialization.	135
6.6	CV Plot for varying transmitter positioning.	135
6.7	Quantile-Quantile Plots for each KPI	136
6.8	Coefficient of Variation Against Number of Simulations for each KPI	137
7.1	A qualitative analysis of potential valuable metrics for subswarming with PSO	143





# List of Tables

4.1	Signal Strength and Wavelength Measurements	119
6.1	Results of Shapiro-Wilk Test and QQ Plot References	132
6.2	Wilcoxon Signed-Rank Test on PSO (1) against Pair-Gradient (2)	134
6.3	Shapiro-Wilk Test Results and QQ Plot References	134
6.4	Analysis of agent initialization and Transmitter Positioning	135
6.5	Results of Shapiro-Wilk Test and QQ Plot References	136
6.6	Wilcoxon Signed-Rank Test on Boustrophedon (1) and Spiral mapping (2)	137
6.7	Wilcoxon Signed-Rank Test on Boustrophedon (1) and tab-QL mapping (2)	137
6.8	Wilcoxon Signed-Rank Test on Spiral (1) and tab-QL mapping (2)	138
7.1	Overview of Supplementary Experiments and Analyses	139
7.2	Mean Value KPIs for Sensitivity Analysis in Particle Swarm Optimisation	140
7.3	Mean Value KPIs for Sensitivity Analysis in Pair-based Gradient Search	141
7.4	Experimental Results with Revised Utility Function	142
7.5	Mean Performance Overview of Maximum Utility Configuration (MUC) and Monte Carlo Tree Search (MCTS) for various agent initializations and transmitter positioning (successful runs: 29 & 44)	143
7.6	Mean Value KPIs for Sensitivity Analysis in Boustrophedon Mapping	144
7.7	Mean Value KPIs for Sensitivity Analysis in Spiral Mapping	145
7.8	Mean Value KPIs for Sensitivity Analysis in Tabular Q-Learning	145



# List of Abbreviations

ABM	Agent-Based Model
ACO	Ant Colony Optimization
AM	Amplitude Modulation
ASK	Amplitude Shift Keying
BCO	Bee Colony Optimization
CEMA	Cyber Electromagnetic Activities
COMA	Counterfactual Multi-Agent Policy Gradients
CRL	Cumulative RSS Line
DFS	Dynamic Frequency Selection
DoA	Direction of Arrival
EA	Evolutionary Algorithms
EM	Electromagnetic
FM	Frequency Modulation
FSK	Frequency Shift Keying
FSPL	Free Space Path Loss
HURMS	Heuristic-driven Utility by Regression-based Metrics Synthesis
IPF	Interference Point Finding
KPI	Key Performance Indicator
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MUC	Maximum Utility Configuration
NLR	Royal Netherland Aerospace Centre
NTRD	Number of TRDx
NTRO	Number of TROx
NUAV	Number of UAVs
PM	Phase Modulation
PPO	Proximal Policy Optimization
PSK	Phase Shift Keying
PSO	Particle Swarm Optimization
QL	Q-learning
RF	Radio Frequency
RL	Reinforcement Learning
RSS	Received Signal Strength

RSSI	Received Signal Strength Indication
RT	RSS Threshold
SI	Swarm Intelligence
SNR	Signal-to-Noise Ratio
TRD	Directional transmitter/receiver
TRO	Omni-directional transmitter/receiver
UAV	Unmanned Aerial Vehicle
UAVs	Unmanned Aerial Vehicles

# Introduction

My master's program at TU Delft offered me an invaluable platform to cultivate my fascination and proficiency in operations research and Artificial Intelligence. Recognizing AI's potential to significantly influence the future, I was eager to delve into its complexities. I actively pursued a comprehensive range of courses in this area, which not only enhanced my understanding but also sparked the inspiration to initiate my own AI-driven projects.

In my journey to apply AI and operations research, I embarked on a series of significant projects. My internship at Lockheed Martin in Texas provided the opportunity to work on AI inspection processes for the F-35, where I integrated AI with manufacturing engineering. In the medical imaging sector, I engaged in organ segmentation during my Deep Learning course, demonstrating AI's impressive capabilities. Further, I explored Reinforcement Learning for autonomous taxiing in the bio-inspired intelligence course and tackled autonomous obstacle avoidance with UAVs in the 'Autonomous Flight of Microvehicles' course. An additional fascinating experience was in the agent-based modeling and simulations (ABMS) course, where I developed airport agent path planning. Moreover, in the Airline Planning and Optimization course, I gained experience working with MILP optimizers, concentrating on the optimization of airline planning and operations. Each of these projects not only expanded my technical skills but also solidified my conviction in the transformative impact of AI and operations research across various sectors.

The connection with the ASDO department at the Royal NLR was established through Michelle Jagtenberg, whom I met in Texas. Collaborating with Olivier den Ouden and Joris Stronkman from the Royal NLR, along with my supervisor Alexei Sharpans'kykh, we merged the concepts of Agent-based Modeling, Swarm Intelligence, and Reinforcement Learning into a comprehensive project within the CEMA domain. This project effectively integrates the knowledge and experience I have gained during my MSc in Aerospace Engineering. My fascination with science has been a constant since childhood, and it is with gratitude that I reflect on the opportunity to contribute to a field that I deeply believe in.

The structure of this thesis report is as follows: Part I presents the scientific paper, which includes the key literature underpinning this research, the formulation of the problem, the methods applied in addressing it, and the findings. Part II encompasses the Literature Study, detailing the preparatory research conducted prior to the commencement of this project. Part III offers additional background context and supplementary information important to the understanding and support of the paper's content.





# I

Scientific Paper



# UAV Swarm Intelligence in CEMA

Storm Holman,\*

Delft University of Technology, Delft, The Netherlands

## Abstract

In response to the increasing challenges of Cyber Electromagnetic Activities (CEMA) in urban settings, characterized by dense electromagnetic (EM) signals and rising data traffic, this research introduces an Agent-Based Model (ABM) aimed at prioritizing critical signals. The primary goal of this research is to deploy a Unmanned Aerial Vehicle (UAV) swarm operating to selectively interfere with communication lines in a CEMA environment. The research goal is segmented into a multi-stage approach, focusing on the following mission tasks for the UAV swarm: i) Transmitter Search, ii) Communication Line Search, iii) Communication Line Mapping, and iv) Interference Point Finding. This research proposes and evaluates various methodologies for these tasks. A methodological contribution is the development of the Heuristic-driven Utility by Regression-based Metrics Synthesis (HURMS) framework. This framework addresses the subswarming coalition formation problem in Multi Transmitter Search. The HURMS framework utilizes the benchmarking Monte Carlo Tree Search (MCTS), a heuristic search method, to enhance the Maximum Utility Configuration (MUC), a transparent utility-based method, overcoming heuristic search limitations and complexities in creating utility functions. While the HURMS-enhanced MUC method effectively located all transmitters in the task, comparative analysis showed MCTS to be about 45% faster and 56-71% more successful in transmitter detection. This highlights potential areas for enhancing the MUC algorithm further under the guidance of the HURMS framework. Furthermore, the Particle Swarm Optimization (PSO) Search was utilized as a velocity controller in the context of Multi Transmitter Search methods, guiding the speed and direction of UAVs. Regarding Single Transmitter Search, a significant observation was that the PSO Search was approximately 38% faster than the novel Pair-gradient Search method in locating a single transmitter. The study also examines Communication Line Mapping by comparing two frontier-based methods with the Multi-agent Tabular Q-Learning method. The frontier-based methods provided better coverage, while the Tabular Q-Learning excelled in precision and adaptability for multi-agent mapping. Data from these methods were applied to find the interference point. Additionally, a Reinforcement Learning (RL)-trained agent was used for Interference Point Finding, proving to be faster but less accurate.

## 1 Introduction

Activities in the cyberspace and electromagnetic spectrum, collectively known as Cyber Electromagnetic Activities (CEMA), have become increasingly important in our society. CEMA encompasses a range of activities including cyber defense, management of electromagnetic (EM) signals, and overseeing the flow of information across various wireless networks. A significant challenge in this domain arises in urban areas, where EM activities are particularly dense and are often crowded with overlapping signals, each varying in importance. Global internet traffic in these areas reached 238 exabytes per month in 2020 and is projected to quadruple by 2025, further complicating the task of managing these communication lines [Utilities One, 2023]. In such environments, the primary focus is on locating the signals causing congestion. This presents a challenge in managing and prioritizing communication channels, especially prioritizing critical EM signals like government networks and key organizational communications. To prioritize these signals, effective strategies to detect and selectively interfere with disrupting communication lines must be deployed. The complexity of this task in urban environments underscores the need for innovative and effective solutions to manage the ever-increasing demands on communication networks and maintain the integrity and security of these important networks.

Unmanned Aerial Vehicles (UAVs), or drones, equipped with specialized payloads, are utilized for tasks involving wireless data communication in various network systems. In the network systems signals from e.g., 3G/4G/5G signals, are present. Existing research by [Ouden et al., 2023], has primarily focused on using a single UAV for these CEMA tasks. However, the use of a single UAV can be challenging for mission tasks in urban environments. This is due to the limitations in coverage, data processing, and operational duration of a single UAV. **This emphasizes the need for more advanced and collective strategies.**

---

\*Msc Student, Sustainable Air Transport, Faculty of Aerospace Engineering, Delft University of Technology

Swarm intelligence, inspired by natural systems, offers a promising solution to this challenge by utilizing multiple UAVs. It draws on the principles observed in the behavior of social insects, bird flocking, and fish schooling, advocating for the efficacy of collective actions over individual efforts in problem-solving, as noted by [Di Caro, 2014]. This decentralized approach leads to the emergence of patterns and structures that help a swarm achieve its objectives [Abd, 2021]. Implementing this concept in UAV operations enables them to collaborate, enhancing essential communications while effectively managing secondary transmissions. This collaborative approach of UAVs allows for comprehensive data collection, redundancy for increased reliability, and scalable responses to varying environmental conditions. The effectiveness of UAV swarm intelligence in fields like search and rescue and flood monitoring has been demonstrated by [Arnold et al., 2020] and [Baldazo et al., 2019]. However, the concept of swarm intelligence in the realm of CEMA remains unexplored. **This leaves a research gap in applying swarm intelligence within CEMA.**

Swarm intelligence shows potential for UAVs to act in urban environments, handling varying signal strengths, frequency bands, and moving targets. The swarms' robustness ensures resilience under diverse conditions, including signal interference and UAV fall-out. Their scalability allows for managing tasks of varying scales, enhancing coverage and communication reliability in dense urban areas. The collective intelligence of UAV swarms, characterized by shared data and joint decisions, potentially exceeds individual UAV capabilities. Another bio-inspired approach is Reinforcement Learning (RL), a branch of artificial intelligence that has achieved super-human performance in various sequential games, like playing the complex game of Go in [Silver et al., 2016]. RL enables agents to learn optimal behaviors through environmental interactions, enhancing decision-making and efficiency in complex scenarios.

**The primary goal of this research is to deploy a UAV swarm operating to selectively interfere with communication lines in a CEMA environment.** These communication lines are identified as areas where signals from two transmitters intersect. The technique of interference involves the strategic placement of a UAV, carrying a transmitter, within the defined communication line area. The technique utilizing this payload has been proven to work by [Ouden et al., 2023], and is not further investigated in this research. The aim is to position it at a strategic point for interference. Once in place, the UAV transmits counter signals, seeking to disrupt or modify the signals emitted by both transmitters. This approach is essential for effectively impacting the communication line, allowing the prioritization and clearance of pathways for more critical signals.

**The mission objective exists of four main mission tasks:** i) Transmitter Search, ii) Communication Line Search, iii) Communication Line Mapping, and iv) Interference Point Finding. Organizing the research goal into these mission tasks not only ensures a systematic approach but also promotes convergence to a solution, enhancing the likelihood of meeting the mission objective of finding, mapping, and interfering with targeted communication lines. The complete mission flow is visualized in Figure 1. The first mission task is the Transmitter Search, where the primary objective is to locate transmitters. This involves starting with Single Transmitter Searches, aiming to effectively locate a single transmitter in terms of detection time and success rate. Alongside this mission task contributes to advancing Multi Transmitter Searches that require coordinated agent efforts and adaptability in environments with multiple communication lines. Within this context, each UAV can only tune into one specific frequency at a time. This limitation necessitates dividing the swarm into smaller units, or subswarms, creating a coalition formation problem to effectively manage and target diverse frequencies. Following this, the second mission task enters the Communication Line Search, focusing on the interaction between different transmitters to identify communication lines. The third subsequent mission task, Communication Line Mapping, involves agents mapping these communication lines. The communication line mapping contributes to strategic planning and situational awareness. The final mission task, Interference Point Finding, focuses on identifying optimal points for applying interference to these communication lines. In this mission task signal strength differences between transmitters are utilized for maximize effectiveness, and efficiently locating these points with minimal localization error.

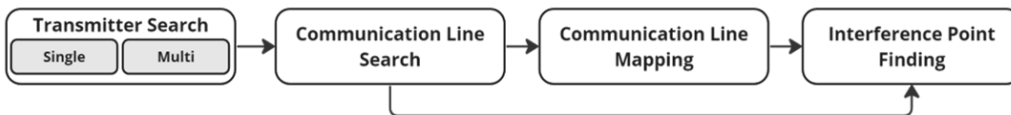


Figure 1: Mission Flow for UAV swarm in CEMA

To accomplish the primary goal of this research, the following research question is posed: *Which methods are most effective in developing a swarm intelligence model for the mission tasks in a specific CEMA environment?*

This research contributes to developing and assessing a comprehensive framework for UAV swarm operations in CEMA environments, targeting interference with communication lines. **The main contribution is the development and evaluation of methods for mission tasks:** i) Transmitter Search, ii) Communication Line Mapping, and iii) Interference Point Finding. More specifically:

- i In Multi Transmitter Search, a core methodological contribution is the development of the Heuristic-driven Utility by Regression-based Metrics Synthesis (HURMS) framework, for the subswarming coalition formation problem. This problem appeared to have sparse and delayed rewards, making it unsuitable for traditional algorithmic or Reinforcement Learning (RL) approaches. The HURMS effectively combines the strengths of utility-based and heuristic-driven methods by integrating the Monte Carlo Tree Search (MCTS) [Silver et al., 2016] method with the Maximum Utility Configuration (MUC) method, complemented by regression-based metrics synthesis. This approach addresses the limitations of each method, which are: the non-real-time nature of heuristic search, and the complexities of creating utility functions in utility-based methods. Furthermore, the Particle Swarm Optimization (PSO) Search [Kennedy and Eberhart, 1995] was employed as a velocity controller in the Multi Transmitter Search, notably enhancing the effectiveness of the HURMS framework. In the Single Transmitter Search task, PSO Search was compared with the Pair-gradient Search, a novel method developed in this research.
- ii Regarding the Communication Line Mapping, this research compared two frontier-based mapping methods with a Reinforcement Learning approach - Multi-agent Tabular Q-Learning mapping.
- iii In the final mission task, Interference Point Finding, data was collected with these mapping methods to determine this interference point. Additionally, a more direct approach to locating the interference point was explored using a RL agent trained with the Proximal Policy Optimization (PPO) [Schulman et al., 2017] algorithm.

This paper is structured as follows: the Related Work (section 2) gives an overview of existing research on UAV coordination and promising methods for the mission tasks, with a section on RL. Following this, the Problem setup for the research is detailed (section 3). Subsequently, the Methodology used for this research is described (section 4). It elaborates on the Agent-Based Model (ABM) and methods used for each mission task. This includes MCTS and MUC utilized in the HURMS framework, PSO Search, Pair-gradient Search, Frontier-based mapping, Tabular Q-learning mapping, and RL-Interference Point Finding. Thereafter, the Model Verification (section 5) explains how the reliability of the environment and methods are assured through tests and analyses. Following that, the Results (section 6) presents an analysis of various experiments, deploying and evaluating the methods. The subsequent section, Supplementary Experiments & Analyses (section 7), includes further experiments, and analyses for the methods used. The Discussion (section 8) highlights limitations and considerations derived from the results and analyses, and puts the research in a broader context. Finally, the Conclusions and Future Work (section 9) provides a summary of the key findings, and suggests directions for future research.

## 2 Related Work

To identify suitable methods for the four mission tasks, this research explores existing work across various domains. The intention is to draw inspiration from work done to develop methods and algorithms tailored for the mission tasks in CEMA. The related work applicable to the scope of this research will be explained in the followings sections: Coordination and Coalition Formation in UAV Swarms (section 2.1), Search and Mapping Methods for the mission tasks (section 2.2), and Application of Reinforcement Learning in mission tasks (section 2.3).

### 2.1 Coordination and Coalition Formation in UAV Swarms

Coordination in UAV swarms, as defined by [Vlassis, 2003], is the process by which the individual decisions of the agents result in good overall performance of the entire system. This concept can be implemented through centralized, decentralized, or distributed systems. Centralized systems, as discussed by [Den, 2023], involve a single control unit that directs the entire swarm, offering robust optimization. However, these systems can become complex and vulnerable as the swarm size increases. Decentralized systems, explored by [June, 2021] and [King and Peterson, 2019], distribute decision-making across individual UAVs, enhancing control and adaptability, making them suitable for tasks like search and mapping. Distributed systems, on the other hand, rely on extensive information sharing among all agents for collective decision-making, improving overall swarm performance but at the cost of increased communication overhead and complexity. For the CEMA mission, a decentralized coordination system is favored due to its balanced approach. This allows for centralized coordination within individual branches or units, ensuring effective control, while also providing resilience by not compromising the entire system if one unit fails. In this mission context, especially for Multi Transmitter

Search tasks, coordination among agents within a decentralized system can be challenging, as the focus is on optimizing for a global outcome, not just success of an individual agent or subgroup of the total swarm.

Subswarming, a method of dividing a large swarm into smaller, independent subgroups for exploring different regions or frequencies, enhances efficiency and coverage in Multi Transmitter Searches, which is part of the mission defined in Figure 1. Each subgroup adapts to specific targets, a process highlighted by [Zhou et al., 2021], although the formation of effective coalitions within these subswarms is less explored. In multi-agent systems, agents can form coalitions, working as unified entities for common goals, and merge into larger coalitions as objectives align. The formation of these coalitions involves evaluating potential values and structures to maximize overall success. Utility-based approaches, like the Shapley value assessments in [Weiss, 2013], further extend this concept, but with an emphasis on individual contribution instead of the effectiveness of the coalition as a whole. Defining the utility function for multi-transmitter search problems can be complex and not trivial, as focus is on collective success rather than individual rewards. The AbACaD methodology, as proposed by [Stef Janssen, 2019], standing for Agent-based Analysis using Causal Discovery, offers a comprehensive framework for analysis in agent-based models and might help here. It combines causal discovery with machine learning and sensitivity analysis techniques to thoroughly investigate emergent behaviors within these systems. This approach could be relevant for gaining insights into effective decision-making, particularly in the complex dynamics of UAV swarm operations.

The complex process of creating utility functions in utility-based methods, as highlighted earlier, calls for innovative solutions. This complexity, particularly in balancing collective and individual goals, might be effectively addressed by generating high-performing solutions with strategic search algorithms. These algorithms, despite the challenges of their non-real-time nature and non-transparent aspects, offer a promising avenue for exploring and optimizing utility functions in dynamic environments like UAV swarm operations.

In addressing complex problems with dynamic, spatial, and behavioral factors, strategic search algorithms like MCTS and Evolutionary Algorithms (EA) emerge as suitable methods. Traditional methods like dynamic programming, previously used in coalition formation [Changder et al., 2019], fall short due to numerous unknowns and the vastness of potential solution spaces. While EA focuses on immediate reward assessment and is less effective in environments with indirect reward structures [Mousavi et al., 2019], MCTS, known for its success in complex games like Go [Silver et al., 2016] where the search algorithm is combined with RL, excels in scenarios with delayed rewards, connecting present actions to future outcomes effectively. The MCTS' efficiency in handling large and intricate search spaces, as shown by [Larson and Sandholm, 1999] in coalition structure graphs, and its adaptability, demonstrated by [Wu and Ramchurn, 2020] in coalition structure generation, make it a promising choice for developing utility functions in challenging scenarios. However, the application of MCTS in optimizing subswarming strategies with PSO Search is yet unexplored, presenting an intriguing research opportunity.

## 2.2 Search and Mapping Methods

As outlined in the preceding section, coordination strategies for Multi Transmitter Searches have been discussed. However, the approach still requires a velocity control algorithm for the agents' speed and direction, beyond the formation of coalitions. This necessity for a velocity control algorithm is also present to tasks such as Single Transmitter Search, Communication Line Mapping, and Interference Point Finding, which are described in Figure 1.

In UAV swarm operations for search missions, various strategies have been investigated. Exhaustive grid search methods divide the search area into grids, ensuring comprehensive coverage [Cho et al., 2022]. Probabilistic search approaches, in contrast, prioritize areas more likely to contain the target based on statistical likelihood, optimizing the search efficiency [Ha, 2018]. Furthermore, bio-inspired algorithms like Ant Colony Optimization (ACO), PSO, and Bee Colony Optimization (BCO) have gained attention. These methods emulate natural behaviors – ants' foraging, birds' flocking, and bees' foraging strategies – to develop efficient search algorithms [Iba, 2013]. PSO, in particular, is recognized for its scalability and ability to explore [Kennedy and Eberhart, 1995]. Also it has proven to work in UAV multi transmitter search setup in [Zhou et al., 2021]. Another approach for transmitter search is shown by [P and Ghosh, 2022]. The study deploys a gradient determination approach for search, offering a more directed and efficient search process in CEMA. However, the method relies on local gradient determination which could be vulnerable to signal disruptions. For the mission task of Transmitter Search, detection and time efficiency are of importance. Methods should be effective in EM environments for signal source search and capable of adapting to searches involving multiple transmitters across various frequency bands. This research implements two methods that meet these criteria. PSO Search is chosen for its scalability and established effectiveness in complex environments. Additionally, a gradient-based approach is employed to provide a search strategy with enhanced precision.



In environmental mapping, UAVs can utilize a variety of techniques to effectively map their surroundings. Cellular Automata-based methods, for instance, model UAVs as cells in a grid, facilitating local communication and decentralized decision-making, enhancing adaptability [Ioannidis et al., 2011]. Additionally, adaptive mapping, highlighted by [Hollinger and Sukhatme, 2014], employs self-awareness in agents, allowing them to evaluate the value of collected data for more efficient path planning, a strategy particularly useful in areas of low data confidence. For comprehensive area coverage, Exhaustive Grid Search is effective in simpler environments. However, in scenarios where the full extent of the area is unknown, Frontier-based Exploration, which focuses on the boundaries between known and unknown regions, proves more suitable [Lu et al., 2020]. RL further extends the capabilities of UAVs in Area Mapping by enabling them to adjust their strategies to various scenarios, like disaster area mapping [Nakanishi et al., 2021]. In the realm of 3D mapping, the Counterfactual Multi-Agent Policy Gradients (COMA) approach stands out for its effectiveness in assigning credit among agents in multi-dimensional spaces, enhancing coordination and efficiency [Westheider et al., 2023]. In the mission task of identifying interference points, speed and precision are of importance. The UAV swarm can leverage mapping data to pinpoint the most effective interference location, where the swarm can most effectively interfere with the communication line. This strategic positioning allows the swarm to optimize the effectiveness of interference, quickly adapt to situational changes, and assess the overall mission’s success. Another approach suitable for Interference Point Finding is discussed in section 2.3.

### 2.3 Application of Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning which is inspired by natural processes. RL is increasingly applied in UAVs for complex tasks, and looks promising for CEMA tasks like Interference Point Finding and multi-agent mapping. In RL, UAVs iteratively learn and refine actions based on environmental interactions, potentially enhancing their performance in diverse scenarios [Kaelbling et al., 1996, Sutton and Barto, 2018].

In the realm of multi-agent mapping, the adaptability offered by RL could lead to more effective coordination and execution of mapping strategies. The iterative learning process inherent in RL allows for the gradual refinement of strategies, potentially uncovering more efficient mapping techniques. Similarly, for Interference Point Finding tasks, RL could offer enhanced capabilities in identifying optimal interference points, especially in environments with prevalent noise and signal disturbances. This aspect of RL, while still under investigation, shows promise for increasing the resilience and effectiveness of UAV operations in complex CEMA tasks.

Prominent methods in RL include value-based methods, which focus on learning an optimal value function using techniques like Q-learning or Deep Q-Networks (DQN), policy-gradient methods that optimize the policy directly through parameter adjustments, and actor-critic methods like PPO. Value-based methods update the value function iteratively to reach the optimal function [Mnih et al., 2013], while policy-gradient methods, such as the REINFORCE algorithm, use gradient ascent to update policies by maximizing expected rewards [Williams, 1992]. Actor-critic methods like PPO combine these approaches, using separate networks for policy (actor) and value function (critic), with PPO introducing a surrogate objective function for more efficient and robust learning [Schulman et al., 2017]. These RL methods can be implemented in both centralized and distributed multi-agent systems, with each approach offering distinct benefits and challenges [Mugan, 2022]. The choice of the most effective RL algorithm is crucial for enhancing UAV efficiency and adaptability and is a subject of ongoing exploration.

A notable application of RL is demonstrated in a study by the New York University (NYU) [Nakanishi et al., 2021], where tabular Q-learning was used to achieve significant coverage in a square environment. This study utilized the agents’ mapping status within their Field of View as the state for decision-making, rather than their positions, effectively operating within the Markov Decision Process (MDP) framework. This framework, common to RL methods, bases state transitions on current states and actions, embodying the Markov property.

## 3 Problem Setup

In this section the problem setup to deploy the methods will be defined. The primary goal of the methods, discussed in section 4, is to interfere with specific communication lines. As depicted in Figure 2, a communication line is established at the intersection area (colored green) where signals from two distinct transmitter types converge. The orange circles represent thresholds for the UAVs, where the Received Signal Strength (RSS) is too high to distinguish signals and be properly operational, thus acting as a so called RSS Threshold (RT) barrier for the UAVs. The first transmitter is a directional transmitter, represented here as a telephone or telecom device (shown as the left black cone), and the second is an omni-directional transmitter, symbolized as a stationary antenna (illustrated as the right black cone). For the purposes of this research, communication lines are specifically defined as existing between these two transmitter types to focus the research scope. However, it

is acknowledged that in real-world scenarios, communication lines can also form between various combinations of multi-directional and omni-directional transmitters.

To facilitate the deployment of the methodologies for each mission task, the environment is set up for two distinct scenarios: the Base Setup and the Global Setup. Both setups are presented with a top-view in a  $10 \times 10 \text{ km}^2$  search area, as depicted in Figure 3 and Figure 4. The tables show a detailed overview of the Base and Global setup. In both setups, the color gradient is used to represent the RSS in decibels-milliwatts (dBm), where warmer colors indicate stronger signals and cooler colors denote weaker ones. The  $10 \times 10 \text{ km}^2$  search area balances scope and manageability for UAV operations. The Base Setup allows straightforward testing of a common communication line set up, while the Global Setup presents an environment to assess the swarm’s adaptability in a more real-world multi-transmitter scenarios.

In the Base Setup, shown in Figure 3, the focus is on a communication line established between two transmitter/receiver units. The omni-directional transmitter/receiver ( $TR_o$ ) is located at coordinates  $x,y = 0,0$  and operates at a frequency of 3.8 GHz, similar to Zigbee device [Everything RF, 2021]. The directional transmitter/receiver ( $TR_d$ ), positioned at coordinates  $x,y = -4000, 0$  functions at a frequency of 4.2 GHz and is directed  $15^\circ$  East. Both  $TR_o$  and  $TR_d$  transmit a continuous pulse with 0 phase. The Global Setup, depicted in Figure 4, features six omni-directional transmitters distributed within the same search area. These transmitters operate in a frequency range of 3.7 GHz to 4.3 GHz and also emit a continuous pulse with 0 phase. An overview of the specifications is provided in Table 1, including Figure 3 and Figure 4.

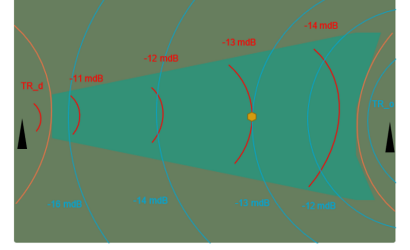


Figure 2: Illustration of a communication line formed by the intersection (green area) of signals from a directional transmitter (left black cone) and an omni-directional transmitter (right black cone).

Table 1: Detailed Overview of Base and Global Setups

Scenario Overview	
Base Setup	Global Setup
Focuses on a communication line between an omni-directional transmitter/receiver ( $TR_o$ ) and a directional transmitter/receiver ( $TR_d$ ), as seen in Figure 3.	Agents detect multiple omni-directional transmitters/receivers ( $TR_o$ ), depicted in Figure 4.
<p>Figure 3: EW Representation - Base Setup</p>	<p>Figure 4: EW Representation - Global Setup</p>
Technical Specifications	
<ul style="list-style-type: none"> <li>• <math>10 \times 10 \text{ km}^2</math> search area</li> <li>• <math>TR_o</math> at xyz coordinates 0,0,-100</li> <li>• <math>TR_o</math> frequency: 3.8 GHz</li> <li>• <math>TR_d</math> at xyz coordinates -4000, 0, -100</li> <li>• <math>TR_d</math> frequency: 4.2 GHz</li> <li>• <math>TR_d</math> lobe <math>15^\circ</math> East</li> <li>• 8 UAVs</li> </ul>	<ul style="list-style-type: none"> <li>• <math>10 \times 10 \text{ km}^2</math> search area</li> <li>• 6x <math>TR_o</math> Sobol distributed (see Appendix 3.2.2) over search area (seed: 0)</li> <li>• Frequency range: 3.7 GHz to 4.3 GHz</li> <li>• 20 UAVs</li> </ul>

## 4 Methodology

This section outlines the methodology to deploy a UAV swarm operating to selectively interfere with communication lines in a CEMA environment. First, a description of the Agent-based Model (ABM) in section 4.1, which serves as the foundational framework to capture the CEMA system, is given. Following this, in section 4.2 an overview of the various methods to complete the mission tasks is provided. Finally, section 4.8 discusses the implementation of the model, detailing the practical aspects and the specifications of the computational setup.

### 4.1 Agent-based Model

This research employs an ABM, particularly focused on capturing the complex dynamics between environmental factors and agent attributes in a CEMA context. Utilizing the ABM paradigm provides a comprehensive framework for understanding and analyzing the effectiveness of swarm intelligence. An agent-based model encompasses three essential elements, according to [Weiss, 2013].

Firstly, it specifies the agents and their local properties. In this model, the agents are UAVs, equipped with both behavioral properties, reacting to the signal environment with velocity updates, and cognitive properties, indicative of internal states like searching, mapping, and Interference Point Finding. Secondly, the environment in which these agents operate is a dynamic 2D (x, y) space, encompassing all non-agent elements and significantly influencing the agents' actions and decisions. This environment is fully observable, allowing agents to have complete awareness of their state, and deterministic, where every action leads to a predictable outcome. The third element is the specification of interactions. In this ABM, interactions occur between agents and between agents and the environment. These interactions are crucial for effective task execution and collaboration, particularly in simulating UAV swarm operations within a CEMA framework. Agents share information and adjust their behaviors based on both observed data and their internal states, facilitating cooperative strategies and collective decision-making.

The ABM simulation functions on a time-stepped basis, with updates to the UAV positions, actions, and states made sequentially until certain predefined termination criteria are met. These criteria might be reaching a maximum number of steps or achieving specified performance thresholds. A more detailed description of the system and the ABM is given in Appendix 2.1.

The primary objective of this system is to explore and evaluate various algorithms for UAV swarm operations, focusing on their effectiveness using different Key Performance Indicator (KPI)s, detailed in section 6. To align the simulated environment with real-world scenarios, assumptions, as detailed in Appendix 1.3, were made. These assumptions were important for handling the experimental workload and improving algorithm development. However, they also show the challenges in applying the findings directly to real-world situations, as further discussed in section 8.

### 4.2 Methods Overview

The methods are developed to solve for the main research objective of interfering with communication lines. The found methods are inspired by existing literature described in section 2, and are modified to serve the CEMA mission. These methods will be applied to execute each of the four mission task described in Figure 1, several methods are utilized. This figure relates to the four mission tasks: Transmitter Search, Communication Line Search, Communication Line Mapping, and Interference Point Finding. The Communication Line Search task is not elaborated upon, because the methodology for this task is trivial and performs well, thus not requiring further experiments. In considering the first mission task, Transmitter Search, a distinction is made between Single Transmitter Search and Multi Transmitter Search. This distinction arises as Multi Transmitter Search can be regarded as a coalition formation problem, which necessitates the use of a velocity control method employed in Single Transmitter Search.

Figure 5 presents an overview of the four mission tasks and the designated methods for each mission task. Each mission task is deployed in either a Global or Base setup.

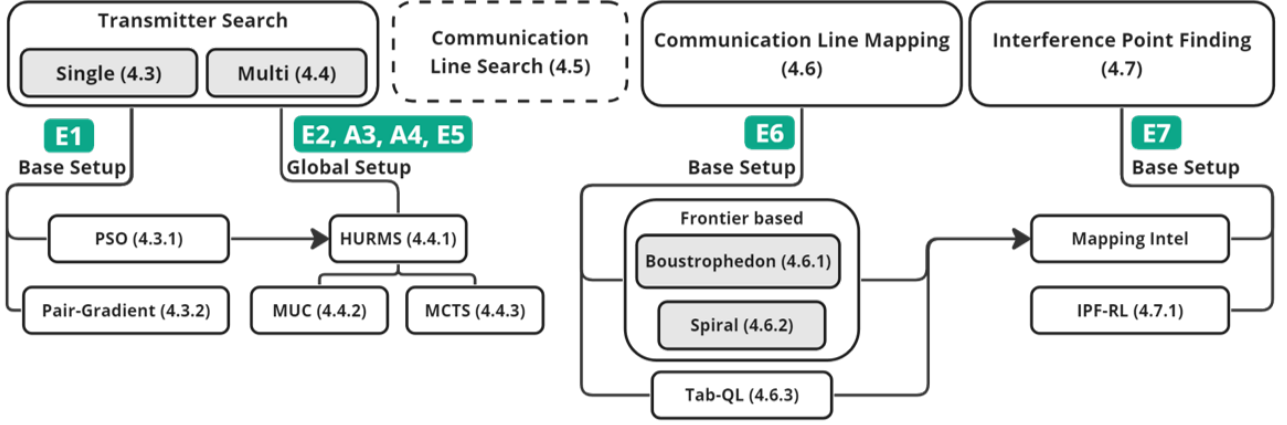


Figure 5: Overview of the methodology, depicting the sequential process from Transmitter Search through Communication Line Mapping to Interference Point Finding, with respective methods and setups for each mission task. The section numbers of where the method is explained can be found within the brackets. The experiments and analyses (A & E) corresponding to these methods are depicted in the green boxes and are detailed in section 6.

### 4.3 Single Transmitter Search

The first mission task described in Figure 1 is the Transmitter Search, consisting of the Single Transmitter Search and the Multi Transmitter Search. In the Single Transmitter Search, two distinct algorithms are evaluated within the Base Setup environment, as illustrated in Figure 3: the PSO Search and the Pair-gradient Search. Research by [Derr and Manic, 2009] has shown that the PSO Search is effective in swarm-based search tasks, and maintaining functionality even with partial UAV failures. However, optimal performance of the swarm hinges on careful calibration of settings and the initial arrangement of UAVs. In contrast to the PSO Search, the Pair-gradient Search is more tailored for scenarios with limited numbers of UAVs. Pair-gradient Search excels in following signal gradients with high accuracy [P and Ghosh, 2022], though it may encounter difficulties in environments where signal spread is irregular.

#### 4.3.1 Particle Swarm Optimization Search

PSO is an optimization method inspired by natural collective behaviors, where each UAV acts as a particle gravitating towards areas of stronger RSS. As visualized in Figure 6, the UAVs update their velocities based on their best known positions and the global best position of the swarm. This process is governed by the velocity update equation  $v_i^{(t+1)} = wv_i^{(t)} + c_1r_1(p_i^{(t)} - x_i^{(t)}) + c_2r_2(g^{(t)} - x_i^{(t)})$ , where  $v_i^{(t)}$  represents the current velocity,  $w$  the inertia weight,  $c_1$  and  $c_2$  the cognitive and social scaling factors, and  $r_1$  and  $r_2$  random factors. To align with the capabilities of real-world top-performing quadcopter drones [Remote Flyer, 2021], the calculated velocity  $v_i^{(t+1)}$  is clipped to a maximum of 30 m/s. The cognitive component, influenced by the UAV's personal best position  $p_i^{(t)}$ , and the social component, related to the swarm's global best position  $g^{(t)}$ , are both integrated into the new velocity calculation. Subsequently, the updated position of each UAV,  $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$ , is determined by adding this newly calculated velocity to its current position, effectively guiding the UAVs to optimized search locations. More details and context of PSO Search implementation are provided in Appendix 3.1.1.

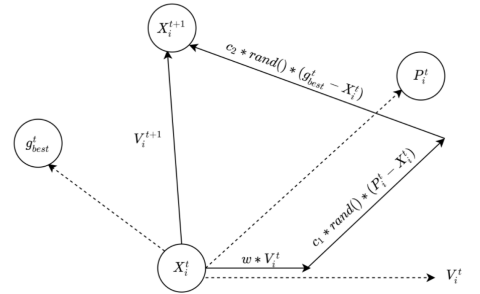


Figure 6: Visualization of the key PSO Search's agent components: current velocity, social component, cognitive component, and random factors  $r_1$  and  $r_2$ .

#### 4.3.2 Pair-gradient Search

Pair-gradient Search involves agents operating in pairs, consisting of a leading agent and a follower agent. The leading agent utilizes a history point and current data from the follower agent, employing triangulation for accurate gradient computation. The gradient at the leader's position is calculated using  $\nabla_{P_1} = f(P_1, RSS_1, P_{1_{last}}, RSS_{1_{last}}, P_2, RSS_2)$ , where  $P_1$  represents the current position of the leader,  $RSS_1$  the signal strength at  $P_1$ ,  $P_{1_{last}}$  the leader's previous position,  $RSS_{1_{last}}$  the signal strength at  $P_{1_{last}}$ ,  $P_2$  the

follower's position, and  $RSS_2$  the signal strength at  $P_2$ . To update the velocity towards the signal source, the average direction pointing towards it is calculated as  $\text{src\_dir} = \frac{\nabla'_1 + \nabla'_2}{2}$ . The normalized direction of movement,  $\text{norm\_dir}$ , is then determined by normalizing this direction vector:  $\text{norm\_dir} = \frac{\text{src\_dir}}{\|\text{src\_dir}\|}$ . This results in the velocity update for the agents being explicitly defined as  $v_{leader} = 30 \cdot \text{norm\_dir}$ , implying a velocity change of 30 m/s in the direction of the signal source. More details and context of this method are provided in Appendix 3.1.2.

#### 4.4 Multi Transmitter Search

Within the Transmitter Search, the second task is the Multi Transmitter Search (depicted in Figure 1). The Multi Transmitter Search is focusing on more than one target or transmitter. For this mission task, methods are deployed in the Global Setup, depicted in Figure 4 (and described in section 3). Regarding Multi Transmitter Searches, it is possible to break a large swarm into smaller groups, known as subswarming, as done by [Zhou et al., 2021]. Each subgroup - in other words a coalition - focuses on a frequency related to one specific transmitter. The challenge lies in adapting these coalitions to efficiently target designated frequencies or areas.

A part of the process of Multi Transmitter Search, is to form these coalitions among the active agents. To form a coalition the agents independently roam to identify frequencies, specifically targeting the frequency with the highest RSS. By setting their receiver antenna bandwidth to the chosen frequency, or transmitter, they proceed to form or join coalitions. These coalitions, which are based on frequency bands, employ a coalition-based PSO Search. The configuration, which is the distribution of the agents amongst the coalitions, is not static; they are periodically reassessed every 5 time steps, a measure introduced to enhance stability within the system. This reassessment facilitates strategic shifts in coalition formations, aimed at enhancing the global success of transmitter detection. The full process flow is illustrated in Figure 7.

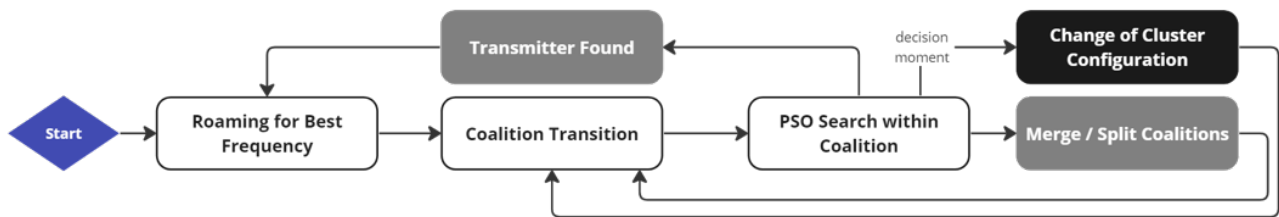


Figure 7: Multi Transmitter Search Process

These reassessments take place within the clusters. Clusters are groups of agents where agents are within a maximum range of 5 km from each other for communication purposes, either directly or indirectly through other agents. In simpler terms, for a coalition to remain intact, each agent should be able to connect with another agent within 5 km, who then connects to another within the same range, and so on. If agents in a coalition cannot maintain this indirect connection within a 5 km range, the coalition will split. On the other hand, if two coalitions operating on the same frequency find themselves indirectly connected within this 5 km range, they are inclined to merge. A cluster-based approach also lays foundation for UAV or communication fall-out, both probable scenarios in real-world. After successfully locating a transmitter, the agents in a coalition will switch back to a roaming state, ready to join new coalitions and continue the search.

To change **cluster configurations**, as depicted in the black box in Figure 7, feasible cluster configurations first have to be generated. In other words, creating sets of how agents within a cluster are distributed over the coalitions.

In the ABM, agents are represented as  $A = \{a_1, a_2, \dots, a_n\}$ , operating within coalitions  $C = \{c_1, c_2, \dots, c_m\}$ , which are formed based on signal frequency. Clusters, denoted as  $G = \{g_1, g_2, \dots, g_k\}$ , are composed of multiple coalitions and are formed based on the proximity of agents. The configurations within these clusters, labeled as  $\Omega = \{\omega_1, \omega_2, \dots, \omega_p\}$ , describe the arrangement of agents with the objective of optimizing collective utility. A visual representation and hypothetical scenario is provided in Figure 8, illustrating how agents form coalitions and clusters, and how configurations evolve within this framework.

During cluster formation, agents with the most connections to other coalitions within their communication range are designated as leaders. These leaders are responsible for initiating cluster formation, often by creating connections with leaders of other coalitions. Notably, these connections can be indirect, facilitated by agents within the coalition, as indicated by dashed squares in diagrams. Within these clusters, leaders collectively decide on potential structural changes or configurations. If an agent is capable of communicating with a member of another coalition, its leader will evaluate whether it should switch coalitions.

The process of forming coalitions begins with agents scanning frequencies to find existing coalitions, either joining them or forming new ones based on the RSS. Coalitions can merge or split, influenced by factors like agent proximity and communication range. For cluster formation, a new cluster  $g'$  is formed for each coalition  $c$  in  $C$ , provided  $c$  is not already part of an existing cluster. Additional coalitions  $c'$  join  $g'$  if their members are within communication range of those in  $c$ . Subsequently,  $g'$  becomes part of  $G$ . When exploring potential configurations within these clusters, each cluster  $g$  is examined. If an agent  $a$  from a coalition  $c$  can communicate with a member from a different coalition  $c'$ , a new configuration  $\omega$  is formed. This involves  $a$  moving from  $c$  to  $c'$ , and the new configuration  $\omega$  is then added to  $\Omega$ .

#### 4.4.1 HURMS: Heuristic-driven Utility by Regression-based Metrics Synthesis Framework

The main problem of the Multi Transmitter Search is selecting the optimal configuration at a given time. Selecting the optimum configuration can be difficult, particularly in complex and dynamic settings with multiple influencing factors. This difficulty is more pronounced when rewards, indicators of how well a system or agent performs, are delayed or sparse. Therefore two approaches can be promising: i) utility-based methods, ii) heuristic search methods.

Employing a utility-based method like the MUC method can be advantageous. This MUC method is designed to provide a clear and transparent structure, as it directly relates the utility to the value of a decision, aiding in real-time decision-making. The development of an appropriate utility function, integral to MUC, can be challenging and necessitates careful consideration. Conversely, the heuristic search methods, such as MCTS, are highly effective in these contexts, especially when dealing with delayed rewards. Reason for this, is that they excel by effectively linking current choices to future outcomes, although they are not typically employed for real-time decision-making. Thus, MCTS is a suitable method for developing a practical utility function in complex scenarios, leveraging its heuristic search capabilities.

To combine the strengths of utility-based and heuristic-driven methods, the HURMS framework is developed in this research. More specifically the MCTS method is combined with the MUC method, along with regression-based metrics synthesis. This integration of the two methods addresses the limitations of each method: the

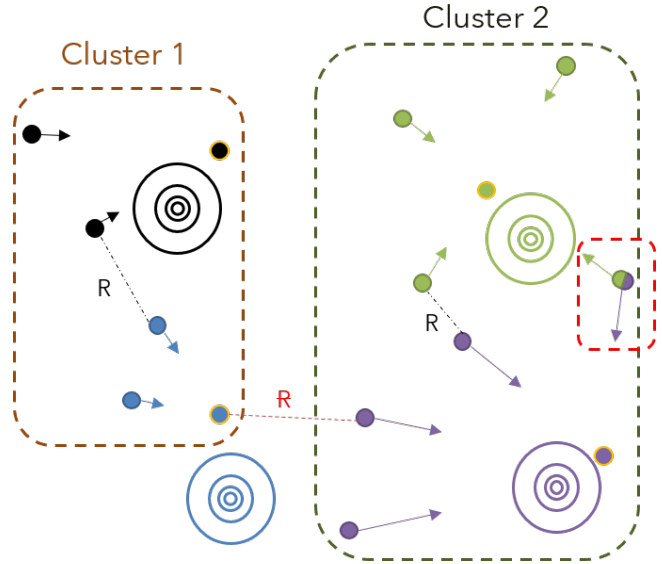


Figure 8: This schematic depicts agents (represented as dots) organized into coalitions (indicated by different colors) performing a PSO Search with their coalition's best agent (yellow circles). Agents within communication range "R" of each other form a cluster. Within these clusters, agents have the flexibility to change coalitions, enhancing global search efficiency. The red box highlights a scenario where an agent's decision to switch coalitions, from purple to green, could significantly impact the search outcome. This potential switch is based on the anticipation that the agent's contribution to the green coalition might prevent its premature convergence, thereby aiding in successfully locating the transmitter.



non-real-time nature of heuristic search and the non-transparent, complex process of creating utility functions in utility-based methods. Drawing inspiration from the AbACaD methodology developed by [Stef Janssen, 2019] and discussed in section 2, the HURMS framework, depicted in Figure 9, employs heuristic search strategies and regression analysis to formulate a utility-based decision-making model. This approach could leverage the solution-search power of MCTS in scenarios with delayed rewards and the clarity and systematic structure of MUC for real-time decision-making. By synthesizing these methods, the HURMS framework provides a comprehensive, transparent, and effective framework for decision-making based on utility and metrics. Therefore, filling the gap between real-time decision-making capabilities and the creation of meaningful, transparent utility functions.

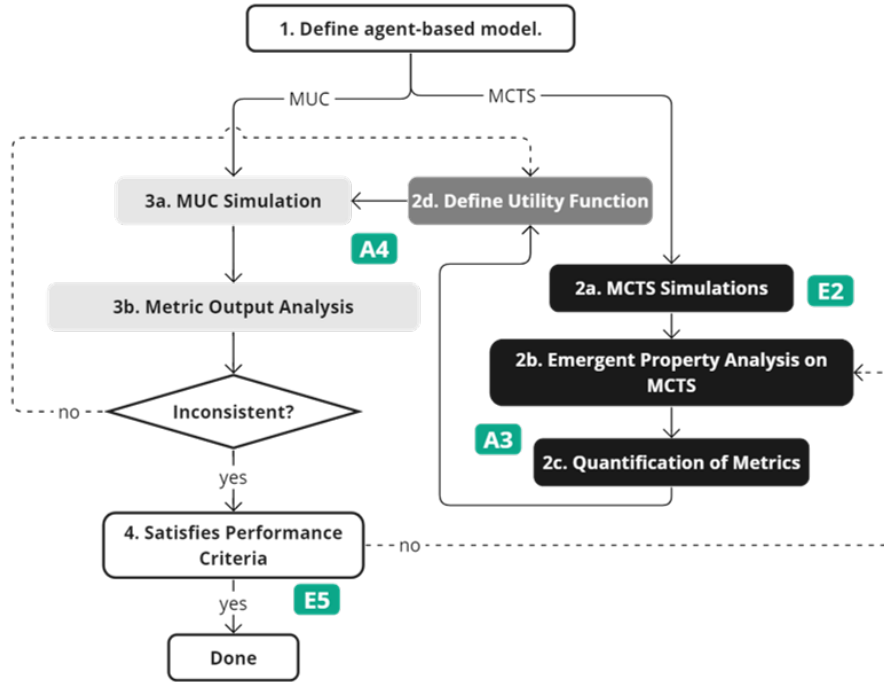


Figure 9: HURMS: Heuristic-driven Utility by Regression-based Metrics Synthesis Framework. This framework facilitates the definition of the utility function for the MUC method, employing the MCTS method. The green boxes illustrate the experiments and analyses (A & E) related to these methods, with detailed descriptions provided in section 6.

As depicted in Figure 9, HURMS framework, starts with defining an agent-based model. This model details the agents and their environment and sets the stage for developing the two methodologies and the analyses. The next steps involve exploring the MCTS parameters during the MCTS Simulations (2a) to generate good performing solutions. These solutions have found maximum amount of transmitters in minimum time. After the MCTS simulations, next step in the framework is to analyze emergent behaviors in these runs (2b). Understanding these behaviors helps in identifying important metrics. These metrics are then quantified (2c) and used to define a utility function (2d). This function is key in guiding the agents’ decisions in the simulation towards desired outcomes.

Further, the framework executions of MUC simulations using this utility function are done (3a). The results of these simulations are checked for consistency with the utility function (3b). If there are differences, the framework examines these to find the reasons. Adjustments are made to the utility function or other parts of the model as needed (2d). The last step is comparative testing. Here, the performance of the MUC approach is compared with the original MCTS approach. This comparison ensures that the utility-based solutions are effective and comply to the criteria of finding maximum transmitters in minimum time. If improvements are needed, additional metrics are added to the utility function (2b). For the sake of completeness and overview, the green boxes in Figure 9 indicate the experiments and analyses done in section 6. The algorithms utilized in the HURMS framework are layed out in detail in section 4.4.2 and section 4.4.3.

#### 4.4.2 MUC: Maximum Utility Configuration

The MUC is a method designed to maximize the effectiveness of a coalition through a utility function. Unlike the approach in [Weiss, 2013], which focuses on individual contributions via Shapley values, MUC emphasizes the

performance of the entire cluster or group. In MUC, the utility of a coalition is not directly measurable, thus a utility function is formed using relevant metrics to estimate the cluster’s performance. This utility is calculated as a weighted sum of both coalition-specific and cluster-specific metrics. The utility  $U(C)$  of a coalition  $C$  is expressed mathematically in Equation 1, considering both coalition-based utility ( $U_C$ ) and cluster-based utility ( $U_G$ ):

$$U(C) = \sum_{i=1}^n w_i^C \cdot m_i^C(C) + \sum_{j=1}^m w_j^G \cdot m_j^G(G) \quad (1)$$

Where  $w_i^C$  and  $w_j^G$  are weights for coalition-based and cluster-based metrics, respectively.  $m_i^C(C)$  represents coalition performance metrics, while  $m_j^G(G)$  indicates metrics for the cluster or group  $G$ .

In the PSO subswarming context, performance metrics are categorized into two types: coalition-based and cluster-based. The coalition metrics assess aspects related to the agents’ arrangement and communication within a coalition, while cluster-level metrics focus on the dynamics and distribution of agents among different coalitions. The normalization of these metrics is essential, given their different scales. This ensures that they can be consistently and meaningfully integrated into the utility function. By adjusting each metric to a uniform scale, the utility calculation is balanced, preventing any metric from having an undue influence.

#### 4.4.3 MCTS: Monte Carlo Tree Search

As mentioned before, MCTS is heuristic-driven search algorithm which can be used in the HURMS framework. MCTS is an algorithm primarily used for decision-making in complex situations, such as in advanced games like Go, as demonstrated by [Silver et al., 2016]. Therefore, it is known for its ability to manage large and intricate search spaces effectively. Building upon this, MCTS has been adapted for the significant search spaces in coalition structure graphs, which [Larson and Sandholm, 1999] initially explored. Their work aimed to identify optimal solutions in extensive combinatorial settings. MCTS uses random sampling in its search tree expansion, enabling a thorough exploration of potential solutions. Another study [Wu and Ramchurn, 2020] further refined MCTS for coalition structure generation, proving its effectiveness and scalability in larger agent-based problems. However, the application of MCTS in subswarming strategies within PSO Search is a new area, presenting opportunities for optimizing subswarm dynamics in PSO Search settings.

In the context of multi-transmitter PSO subswarming, MCTS is employed to determine actions taken at decision moments every 5 steps. An action refers to picking a configuration of coalitions for each cluster. Each possible configuration would be represented by a node within the MCTS framework. An early choice in the process can greatly affect the configuration options available later, as well as the overall distribution of agents, thereby impacting the solution’s final outcome and reward.

MCTS involves a four-stage process: selection, expansion, simulation, and backpropagation. The construction of the MCTS decision tree, shown in Figure 10, is a gradual process, with each layer having a set step limit for expansion. The process continues until reaching an overall stopping condition. Such as finding all transmitters or exceeding a specific step count, which makes further rollouts not possible. This leads to a tree that outlines various decision paths.

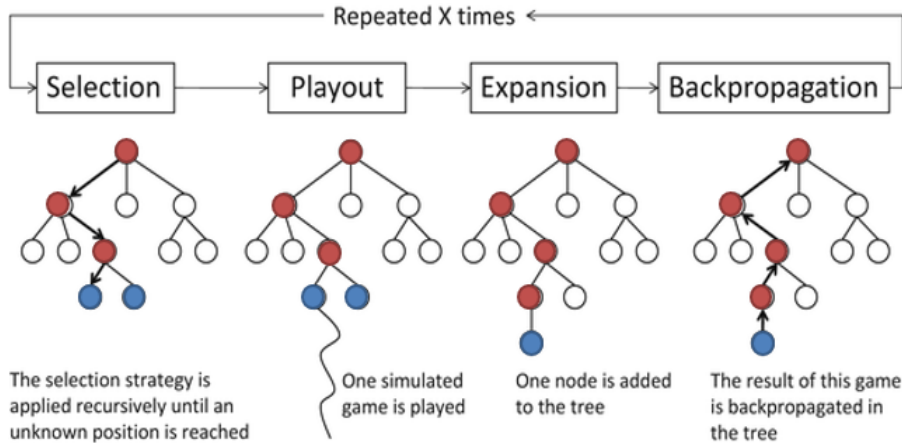


Figure 10: MCTS Node Structure & Decision Path Finding [Soemers et al., 2018]

The selection of child nodes (blue nodes) in the tree is guided by the UCB1 algorithm, which balances between exploiting well-performing nodes and exploring less visited ones. The UCB1 score, which influences this selection, is calculated using the formula in Equation 2:

$$UCB1 = \frac{\text{Average Value}}{\text{Visits at Node}} + C \cdot \sqrt{\frac{2 \cdot \ln(\text{Visits at Parent Node})}{\text{Visits at Node}}} \quad (2)$$

This formula determines the value of each option; it includes the success rate of the node ( $\frac{\text{Average Value}}{\text{Visits at Node}}$ ) and a term that encourages exploration, where ‘C’ is a constant that influences the level of exploration undertaken by the algorithm.

Once the UCB1 scores are calculated, the tree’s selection stage chooses the child node with the highest score, which becomes the decision node (red) for the next rollout or simulation phase. During rollout, sequences of moves are tested to assess their outcomes, and this simulation continues until a predefined condition, such as a maximum number of steps, is met. The rewards from the rollouts are expressed in Equation 3:

$$R = N - \frac{S}{1000} \quad (3)$$

In the equation,  $R$  represents the reward,  $N$  denotes the number of TR\_x found, and  $S$  signifies the total number of steps taken. These rewards are then propagated back through the tree, updating the average value and number of visits of each node and influencing future node selections. To optimize the simulation process, a deep copy of the ‘swarm’ object is utilized.

MCTS results in two primary outcomes: the ‘decision path’ of chosen configuration at each decision moment, and the ‘best path’, which is the most successful sequence found. Recording rollout results can improve future decision-making. MCTS’s ability to handle large and complex domains efficiently makes it a valuable tool for finding quality solutions within practical timeframes, especially in scenarios with numerous potential coalitions. Its systematic approach, based on simulation and selection, is crucial for evaluating potential coalition configurations. For a detailed application of MCTS in cluster-based coalition formation for PSO subswarming, see Algorithm 13.

## 4.5 Communication Line Search

As mentioned in the start of this section, the Communication Line Search is an in-between mission task within the defined CEMA mission framework, depicted in Figure 1. This task activates once the  $TR_o$  is located and involves searching for the intersection of the  $TR_o$  and  $TR_d$ . The Communication Line Search method employs a single agent tasked with establishing the communication line. This process depends on both the current and historical positions of the agent and the corresponding signal strengths recorded. The agent’s position at a given time is marked as  $\mathbf{p}$ , while  $\mathbf{p}_{\text{last}}$  (or  $RSS'$ ) and  $\mathbf{p}_{\text{pre-last}}$  (or  $RSS''$ ) represent its last and second-to-last positions, respectively. These positions are used for determining the agent’s trajectory towards a target, known as the mapping point. The algorithm includes specific procedures for navigating the communication line. Using ‘circle\_velocity’ (refer to Algorithm 4), the agent calculates its next move based on its recent locations and signal strengths. Then, the ‘SignalStop’ procedure (see Algorithm 5) allows the agent to decide whether to continue its movement or stop, guided by the analysis of current and previous signal data. This method ensures the agent effectively searches for and identifies the mapping point located on the border of the communication line. As the algorithm has a bridging role in the overall CEMA mission, further analysis is excluded. For an in-depth explanation of the Communication Line Search algorithm, see Appendix 3.2.

## 4.6 Communication Line Mapping

In the mission task of Communication Line Mapping, two frontier-based mapping methods (section 4.6.1 & section 4.6.2) and one Q-learning method (section 4.6.3) have been implemented in the Base Setup, as shown in Figure 3.

The frontier-based mapping methods employed include a Boustrophedon pattern and a Spiral pattern. In these methods, agents are tasked with dividing the area into partitions to achieve effective coverage. They utilize either area partitioning or frontier-based strategies for this division. The mapping boundary for these methods is defined by the lobe borders of the directional signal.

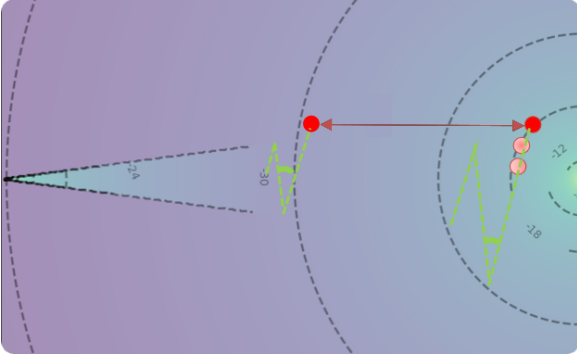


Figure 11: Visualization of the Boustrophedon Mapping. Highlighting offset distance, turn angles, and the agent trajectory following boustrophedon pattern.

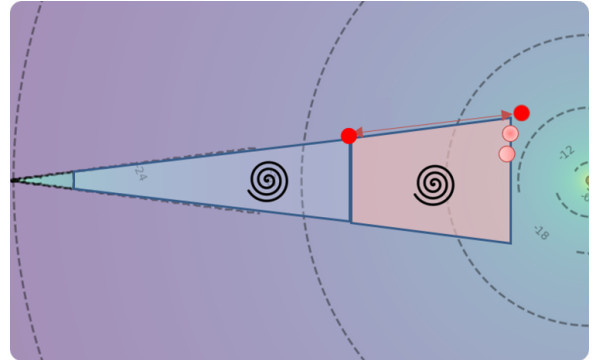


Figure 12: Visualization of the Spiral Mapping. Showing the inward spiral trajectory and distinct legs of the agent’s movement.

#### 4.6.1 Boustrophedon Mapping

This algorithm employs a back-and-forth pattern for mapping areas. As agents enter the area, they follow a systematic boustrophedon pattern, traversing back and forth across the width and making pronounced turns at the boundaries, as depicted in Figure 11. Key components of this method, illustrated in the figure, include the agent offset distance (indicated by the red arrow), turn angles (shown with green arcs), and the relative starting angle. The dotted green trajectories in the figure represent the agents’ path following the boustrophedon pattern. These properties are crucial in defining the agents’ trajectory and systematic coverage pattern. In practice, agents adjust their velocity and update positions in response to environmental and mapping requirements.

#### 4.6.2 Spiral Mapping

Drawing inspiration from search and rescue mission strategies [Arnold et al., 2018], this algorithm uses an inward spiraling pattern. As shown in Figure 12, agents spiral towards a central reference point, with the spiral trajectory defining the mapping boundary. This figure demonstrates the spiral mapping in action, with the agent following the inward pattern. Key elements of this method include the helper agent’s offset, similar to the Boustrophedon Mapping, but with initial velocities based on the position difference of the agents. The spiral borders and the agent’s specific leg within the spiral are critical trajectory components, as highlighted in the figure. Further details on these agent properties, including offset, shrink factor, initial velocities, and actions focused on altering the agent’s leg and adjusting velocity, are provided in Appendix 3.3.2.

The Frontier Boustrophedon and Spiral Mapping algorithms are tailored for different mapping applications within the studied environment. Boustrophedon Mapping is particularly effective in large, open areas, capitalizing on factors such as the agent’s speed and starting orientation. Its key advantage is its straightforward, comprehensive area coverage, especially suitable for spaces without obstacles. On the other hand, Spiral Mapping is geared towards ensuring thorough coverage when the boundaries are clearly defined. This approach, however, presents a limitation as it necessitates prior knowledge of the environment’s borders. For more in-depth operational details, see Appendix 3.3.1 and Appendix 3.3.2.

#### 4.6.3 Multi-agent Tabular Q-learning for Mapping

The Multi-agent Tabular Q-learning algorithm, as detailed in a study from the NYU [Nakanishi et al., 2021], showcases the effective use of tabular Q-learning for deploying multiple agents in a square environment to achieve substantial coverage. This method emphasizes the advantage of utilizing mapping status within an agent’s Field of View (FoV) as a state for enhanced coverage and scalability, as opposed to relying solely on agent positions. The algorithm operates within the MDP framework, wherein the probability of transitioning to a particular state is determined exclusively by the current state and the chosen action, consistent with the Markov property.

In this Tabular Q-learning approach, agent properties such as state and actions are discretized, requiring the initialization of a grid to facilitate agent maneuvers. This grid, established by the first mapping agent, starting from a designated mapping point and extends in the opposite direction of its historical path, with each cell sized at 30m x 30m. The grid’s dimensions are chosen to ensure complete coverage of the communication line, thus providing a discrete framework for state observation and action execution.

In the shared environment of the Q-learning algorithm, each agent’s observations ( $O$ ) and actions ( $A$ ) are interconnected, primarily focused on the local mapping status of adjacent cells. Observations are denoted

by values like 0 (undiscovered), 1 (mapped), -1 (obstacle - not considered), and -2 (boundary), as depicted on the left side of Figure 13. The actions include moving up, down, left, and right, represented as  $a_i \in \{a_{up}, a_{down}, a_{left}, a_{right}\}$  and illustrated on the right side of Figure 13. An agent’s internal state ( $I$ ) is defined by its observation ( $o_i(t)$ ) of these mapping statuses. In this setup, a reward of +10 is awarded to an agent for each move towards an undiscovered cell, encouraging exploration. The Q-learning algorithm thus enables interactions between agents, where one agent’s actions can influence the states and rewards of others, promoting both cooperative and competitive elements in the mapping process.

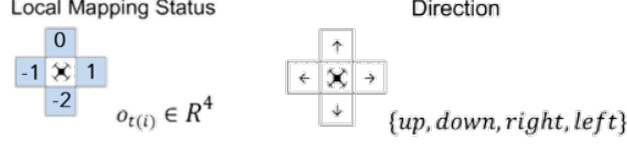


Figure 13: Illustration showing local mapping statuses (left) and potential actions (right).

The training procedure employs a Centralized Training with Decentralized Execution (CTDE) approach. Throughout the training phase, agents share resources, pooling their experiences for collective learning. Post-training, each agent executes its independent policy based on the strategies learned. This process is augmented by curriculum learning, gradually increasing the complexity of tasks to instill foundational behaviors before tackling more challenging environments. Pre-training is conducted in a triangular, non-predefined border environment, as inspired by [Nakanishi et al., 2021], involving 200,000 episodes. The simplified environment is shown in Appendix 3.3.3. Following this, agents undergo an additional 30,000 episodes of training in a specialized NLR-EW environment, as shown in Figure 3. The training algorithm and its specifics are detailed in Algorithm 1.

---

**Algorithm 1** Training Q-Learning for Grid-Based Agent Mapping

---

```

1: procedure TRAINGRIDMAPPING_Q_LEARNING( $Q$ , episodes,  $\alpha$ ,  $\gamma$ ,  $\epsilon$ ,  $t_{terminate}$ )
2:   Initialize Q-table  $Q(o, a)$  for all observations and actions.
3:   for each episode in episodes do
4:     Initialize observation  $o$  based on the agent’s local mapping.
5:     Set  $t_{undiscovered} = 0$  ▷ Time counter for undiscovered cells
6:     while  $t_{undiscovered} < t_{terminate}$  do
7:       Choose action  $a$  from  $o$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy).
8:       Take action  $a$ , observe signal strength  $RSS_{dir}$ , and next observation  $o'$ .
9:       if  $RSS_{dir} = 0$  then
10:        Mark border cells in direction  $a$  as -2.
11:       end if
12:       for each neighboring cell  $c$  of current position do
13:         Observe state of  $c$  and update observation  $o$  accordingly.
14:       end for
15:        $Q(o, a) \leftarrow Q(o, a) + \alpha [s + \gamma \max_{a'} Q(o', a') - Q(o, a)]$ .
16:       if no new cells discovered in action  $a$  then
17:          $t_{undiscovered} \leftarrow t_{undiscovered} + 1$ 
18:       else
19:         Reset  $t_{undiscovered} = 0$ 
20:       end if
21:       Update observation  $o \leftarrow o'$ .
22:     end while
23:   end for
24: end procedure

```

---

The algorithm uses  $Q(o, a)$  for the value of each pair of observation and action. The agent’s current observation ( $o$ ), based on what it sees around it, decides its next action ( $a$ ) from options like up, down, left, or right. After the action, the agent gets a new observation ( $o'$ ) and checks the signal strength ( $RSS_{dir}$ ) in the direction it just moved. The algorithm stops if the agent has not found any new cells for a while, tracked by  $t_{terminate}$ , and  $t_{undiscovered}$  keeps track of how long it’s been since the last new cell was found. This method aims to make sure the agent maps the area well and responds to changes as needed.

The Q-learning algorithm for grid-based agent mapping, detailed in Algorithm 1, involves precise tuning of several key parameters to function effectively, as inspired by [Nakanishi et al., 2021]. The learning rate  $\alpha$ , set at

0.01, determines the impact of new information on what the agent has already learned. The epsilon parameter  $\epsilon$ , which starts at 1 and ends at 0, guides the agent in balancing between trying new actions and using known ones. The discount factor  $\gamma$ , valued at 0.5, helps prioritize future rewards over immediate ones, while the reward decay rate  $r$ , set at 0.9999, controls the rate at which the value of rewards decreases. Additionally, the algorithm employs a reward structure that gives a +1 reward for each new cell discovered, encouraging exploration. The state space, defined as a field of view of 4 cells, and the total number of training episodes, set at 200,000, are also crucial elements. Properly adjusting these parameters, particularly in environments with multiple agents, is key to the algorithm’s success and involves iterative testing and adjustments.

The hypothesis for the multi-agent Q-learning approach is that starting with simpler training tasks will help agents learn basic mapping skills before they move on to more difficult environments. This method is expected to help agents work together in changing environments. There might be issues, like agents getting stuck or not covering some areas because of the limited FoV, but these could be uncovered by allowing agents to sometimes choose actions randomly. The main benefits of this approach are that it teaches agents basic skills, can adapt to different environments in terms of scale and shape, and works with more agents.

## 4.7 Interference Point Finding

The Interference Point Finding task is the concluding mission task in the CEMA mission, as outlined in Figure 1. This task involves pinpointing the most effective location for signal interference within communication lines. The objective is to minimize the difference in RSS levels between two communication signals, essentially finding the point where  $\min(|RSS_o - RSS_d|)$  occurs. This point is deemed optimal because at this location, the UAV can achieve equivalent interference effects on both signals using the same power output, which is advantageous for energy efficiency.

This phase utilizes two distinct methods to achieve its goal of pinpointing the optimum interference point. The first method focuses on mapping the communication line, as detailed in section 4.6. Once the mapping is complete, signal maps for both signals are generated. The point with the lowest RSS difference between these signals is identified as the optimal location for effective interference. This specific point is marked as a yellow hexagon in Figure 14. In this figure, the green area represents the communication line or mapping area. The red square denotes the mapping point. The color gradient in the figure illustrates the RSS difference, with brighter areas indicating higher differences and darker areas signifying lower differences.

### 4.7.1 Reinforcement Learning for Interference Point Finding

The second method applies RL using the PPO algorithm, an actor-critic method, to train an agent in the environment shown in Figure 15. PPO [Schulman et al., 2017] merges value-based and policy-gradient techniques, employing separate networks for the policy (actor) and the value function (critic).

In the RL framework implemented in this research, the agent’s learning and execution processes are informed by specifically defined state representations, action spaces, and a structured reward mechanism. The agent gathers observations ( $o$ ), which include its relative position and the RSS values, formulated as  $s = (d_x, d_y, RSS_o, RSS_d)$ . The set of possible actions ( $a$ ) for the agent includes a range of discrete choices, such as remaining stationary, moving at predefined speeds in the x or y directions, articulated as  $a = \{\text{stay, move } 10 \text{ m/s, move } 30 \text{ m/s}\}$ . The reward function is defined as  $R = -\frac{|RSS_d - RSS_o|}{max\_diff}$ , where  $max\_diff$  is equal to the RSS at the RT (orange circle in Figure 15). This function is designed to efficiently steer the agent’s actions by penalizing the variance between the received signal strengths ( $RSS_d$  and  $RSS_o$ ).

Learning parameters are the learning rate, set at 0.0002, impacts updates to the policy network; the discount factor ( $\gamma$ ), set at 0.95, which balances immediate versus future rewards; the clip range, set at 0.2, which limits

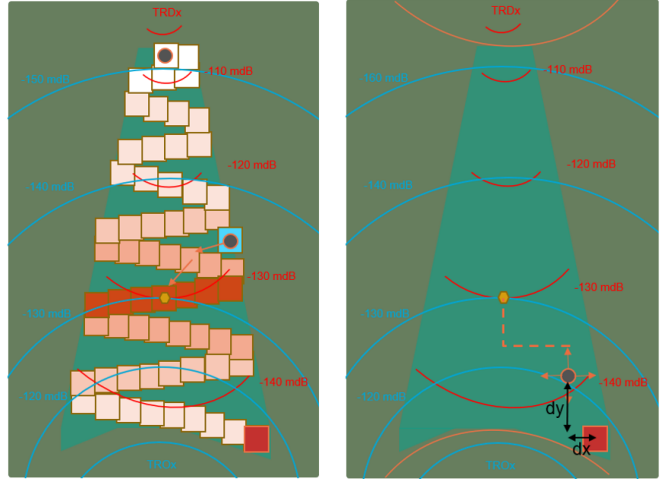


Figure 14: Mapping Intel - Reinforcement Learning - Interference Point Finding



policy update steps; and the batch size, set to 64 experiences, which determines the update frequency. For further details on the method implementation, refer to Appendix 3.4, and for a comprehensive explanation of the PPO algorithm, refer to Algorithm 14.

RL can find the interference point more quickly than the mapping-based method but may be less precise due to dynamic instabilities. Its speed comes from bypassing the initial data collection step that mapping requires. Mapping-based methods, however, provide a more reliable interference point identification through thorough signal analysis. On the other hand, RL’s dependency on a reward function might not effectively account for all environmental complexities, which could affect localization accuracy. Despite being rich in data, mapping approaches are limited by the time and resources necessary for data gathering and analysis.

## 4.8 Model Implementation

The implementation of the model occurs within a 2D ( $x, y$ ) operational space, utilizing the NLR-EW Toolbox. This Toolbox is a comprehensive library designed for the control and manipulation of radio frequency elements such as transmitters, receivers, reflectors, and countermeasures, allowing for the adjustment of various attributes including position, velocity, attitude, and signature lobe features. In addition, signal processing parameters such as bandwidth, frequency, power, and pulse width can be altered to suit specific requirements. This environment is deterministic, ensuring consistent outcomes that are directly dependent on the actions of the agents. For the effective implementation of the ABM, various Python libraries are employed. These libraries serve a range of functions, from numerical operations to data visualization and reinforcement learning. Key libraries used include SciPy/Numpy for data manipulation, Matplotlib for visual representation, Stable Baselines [sta, ] for reinforcement learning algorithms, and [Gym, 2023] for the creation of custom RL environments. For more technical details refer to Appendix 2.3.

## 5 Model Verification

Verification processes were conducted to ensure the reliability of the environment and methods through tests and analyses. This involved conducting both unit and system tests, along with resolving all compilation errors. The verification steps documented in this paper include: i) Testing the NLR-EW environment functions (Appendix 4.1.1), ii) Mission-specific functions such as PSO Search (Appendix 4.1.2), iii) Gradient determination (Appendix 4.1.3), iv) Mapping (Appendix 4.1.4), and v) MCTS (Appendix 4.1.5).

At first, the NLR-EW environment was verified for signal detection and processing capabilities. Signal strength measurements at various locations confirmed the expected operational behavior of both directional and omnidirectional transmitters. Secondly, the verification of the PSO Search assessed its convergence within a 10 km x 10 km area, focusing on the agents’ navigation towards a collective target. Each agent, limited to 200 steps and a maximum velocity of 30 meters per second, had a potential travel distance of up to 6000 meters. In 100 independent runs, the framework consistently showed agents converging within the 50-meter threshold. Then, the gradient determination algorithm was tested for its accuracy in identifying signal source directions. The algorithm appeared to successfully calculate gradients and ascertain source directions within a set tolerance range. Then, the mapping activities were verified to ensure they predominantly occurred within a designated triangular area. Most mapping activities were contained within this area, with minimal deviations due to agent dynamics.

Finally, the verification of the MCTS tree, as depicted in Appendix 4.1.5, involved analyzing decision node visits and values within the tree structure. This analysis showed an increasing trend in the number of nodes and their visits, coupled with a stabilization of node values ( $u$ ). Notably, there was an increase in the number of visits ( $v$ ) and average values for selected nodes in each search layer. This pattern, particularly the rising visit counts and stable values at certain nodes, suggests that the MCTS algorithm is effectively focusing on more promising decision paths. As the algorithm repeatedly visits and evaluates these nodes, it refines its choices, leading to a more targeted and efficient exploration of the decision space. The consistent improvement in decision quality across successive layers indicates that the algorithm is systematically refining its strategy, a clear sign of improved decision-making. This process, underpinned by the backpropagation mechanism in MCTS, demonstrates how the algorithm adapts and enhances its decision-making approach as it navigates through the decision tree.

In summary, the verification procedures employed in this research comprehensively evaluated the performance of each system component, confirming their functionality and effectiveness.

## 6 Results

This section will discuss the experimental setup and the resulting outcomes from applying the methods and algorithms in a practical context. The experiments aim to address the research question: Which methods are most effective in developing a swarm intelligence model for Transmitter Search, Communication Line Mapping, and Interference Point Finding in a specific CEMA environment? Table 2 outlines each experiment including the ID, mission task, method utilized, and setup. For a better understanding of context, refer to the green blocks in Figure 5 and Figure 9. For each mission task, the experiments are detailed in the following sections including: the goal, experimental setup and KPIs, followed by the results. For methods specific hypotheses refer to Appendix 3.

Table 2: Overview of Experiments

ID	Mission Task	Method	Setup
E1	Single Transmitter Search	KPI Performance: PSO Search & Pair-based Gradient	Base Setup
E2	Multi Transmitter Search	MCTS - Parameter Analysis	Global Setup
A3	Multi Transmitter Search	Visual Analysis of Emergent Properties and Metric Quantification via MCTS	Global Setup
A4	Multi Transmitter Search	MUC - Utility Defining and Analysis	Global Setup
E5	Multi Transmitter Search	KPI Performance: MCTS & MUC Comparative Tests	Global Setup
E6	Communication Line Mapping	KPI Performance: Boustrophedon, Spiral, Tabular QL	Base Setup
E7	Interference Point Finding	KPI Performance: RL-Interference Point Finding & Mapping Intel	Base Setup

### 6.1 Single Transmitter Search (E1)

Regarding the Single Transmitter Search task, *KPI Performance: PSO Search & Pair-based Gradient* in E1, was designed to test the methods for the mission task of Single Transmitter Search. It aims to determine which method, PSO Search or Pair-gradient Search, is most effective and efficient in locating a single transmitter.

#### Experimental Setup

The first experiment was deployed within a  $10km^2$  Base Setup, as illustrated in Figure 3. The effectiveness is mainly measured in terms of time to detect and success rate. The experiment deployed PSO Search and Pair-gradient Search algorithms, each initiating from a uniform and random distribution across the search area. The setup used 8 agents per run, balancing efficiency and computational load. The PSO Search was configured with specific coefficients (cognitive: 0.1, social: 0.05, inertia weight: 0.8, maximum velocity:  $30 m/s$ ) [Tam, 2021], while Pair-gradient Search utilized a  $5 m$  follower offset and  $20 m/s$  velocity.

Performance was measured using KPIs (detailed in Appendix 5) such as Total Time Steps in Steps, Swarm Mean Distance to RT in meters, Trajectory Coverage in  $m^2$ , Energy Usage in  $(m/s)^2$ , Compute Time in seconds, and Failure Rate in percentage. The experimental procedure involved 100 runs for each method. Runs were terminated upon locating the target (TR\_o) or after reaching 10,000 steps. To ensure accurate performance assessment, runs that did not locate TR\_o were excluded from the primary analysis but provided insights into algorithm robustness.

#### Results

In the Single Transmitter Search experiment, PSO Search and Pair-gradient Search algorithms were compared using various performance metrics. The comparison, detailed in Table 3, highlights PSO Search’s faster performance in reaching the RT, taking an average of 751 steps (or 94 seconds: divide by 8 agents), compared to Pair-gradient’s 1218 steps (or 153 seconds). The longer time for Pair-gradient Search is attributed to its initial pairing phase, suggesting a potential area for optimization.

PSO Search, however, exhibited a 12% failure rate in converging to a solution across 100 runs, often occurring when agents were unevenly distributed at initialization. A more spatial initialization strategy could potentially mitigate this issue. In terms of energy efficiency, Pair-gradient Search outperformed PSO Search, using only  $658970 (m/s)^2$  against PSO Search’s  $936643 (m/s)^2$ , likely due to the stationary nature of Pair-gradient’s leaders.

Regarding proximity to the RT, Pair-gradient Search agents had a higher mean distance (2055 meters) compared to PSO Search’s 1187 meters. Additionally, PSO Search demonstrated denser clustering near the target with a trajectory density of  $5.93 \times 10^{-8} m^{-1}$ , surpassing Pair-gradient’s  $5.75 \times 10^{-8} m^{-1}$ . This could be a result of the limitations imposed by Pair-gradient’s pairing system versus PSO Search’s social dynamics that encourage closer proximity to the target.



Table 3: Mean Value KPIs for Comparison Between PSO Search and Pair-gradient Search for Single Transmitter Search (88 runs)

Algorithm	Steps to RT	Mean distance RT [m]	Coverage [ $m^2$ ]	Density [ $m^{-1}$ ]	Energy usage [ $(m/s)^2$ ]	Compute Time [s]	Unsuccessful searches
PSO Search	751*	1187	35661863	$5.93 \times 10^{-8}$	936643	0.069*	0.12
Pair-gradient Search	1218	2055*	34678424	$5.75 \times 10^{-8}$	658970*	0.135	0

\* denotes  $p < 0.05$ , indicating statistical significance.

Overall, the evaluation of both algorithms reveals distinct strengths and weaknesses. While PSO outperforms Pair-gradient Search in speed, being approximately 38% faster in locating a single transmitter, it faces challenges in convergence reliability. On the other hand, Pair-gradient Search shows more energy efficiency and initially higher success rates, though its pairing mechanism and search initiation could be optimized for speed.

## Statistical Analysis

The study used statistical methods to analyze the data. The Shapiro-Wilk test [SHAPIRO and WILK, 1965], which is generally more effective than the Kolmogorov-Smirnov test [Saculinggan and Balase, 2013], was used to assess normality, along with QQ-plots. These indicated to not assume normality and to take a conservative approach. For each simulation, the coefficient of variation was calculated. This helped in deciding the number of simulations required, based on the data distribution. Because the data was assumed to be not normally distributed, the Wilcoxon Signed-Rank Test [Rey and Neuhauser, 2011] was chosen. This paired non-parametric test was used for identifying differences in important measures, such as the total steps and Swarm Energy Usage.

The Wilcoxon Signed-Rank Test, using an alpha level of 0.05 for statistical significance, revealed key findings. The step count, a driving metric, displayed a significant difference with a p-value of  $5.81 \times 10^{-13}$ , indicating PSO’s superiority over Pair-Gradient in this aspect. Additionally, Total Energy was also significant, with a p-value of  $9.05 \times 10^{-13}$ , leading to the rejection of the null hypothesis that the performances of the two algorithms are identical in this regard. These results, coupled with significant differences observed in Mean Distance to Origin and Computational Time, reinforce the conclusion that PSO outperforms Pair-Gradient in terms of speed, but not for all KPIs. Although Coverage and Density did not reach statistical significance, their moderate effect sizes suggest notable differences between the algorithms, although not strong enough to be deemed statistically significant at the chosen alpha level. A complete overview of the statistical analysis can be found in Appendix 6.1.

## 6.2 Multi Transmitter Search (E2, A3, A4, E5)

The objective of the experiments in the Multi Transmitter Search is to apply the HURMS framework in creating a utility function for the MUC approach, and to test the defined utility function against the MCTS as benchmark. Solutions are evaluated based on detection speed and the number of transmitters found. The HURMS framework involves both qualitative and quantitative analyses to be performed in this research. To deploy and test the HURMS model, the following experiments and analyses (E2, A3, A4, E5) were done:

The *MCTS Parameter Analysis* in E2 aimed to explore effective and less effective MCTS solutions and understand the impact of various MCTS parameters on the search strategy’s thoroughness and efficacy. In *Visual Analysis of Emergent Properties and Metric Quantification via MCTS* in A3, the goal was to qualitatively assess emergent behaviors in MCTS and to determine quantified metrics. These metrics are utilized in the development of the utility function and provide insights into the characteristics of successful and unsuccessful runs. The focus of *MUC - Utility Defining and Analysis* in A4 was on defining a utility function that scores potential actions within the simulation, aligning agent decisions with the system’s desired outcomes. Finally, the goal of the *comparative tests between MUC and MCTS* in E5 was to evaluate adaptability to diverse scenarios, providing an analytical comparison between MUC and the benchmarking MCTS method. A run was considered successful upon detecting all six transmitters.

## Experimental Setup

These experiments and analyses are all conducted in the Global Setup, with 20 agents and 6 transmitters. The velocity control algorithm was PSO Search, with specific coefficients (cognitive: 0.1, social: 0.05, inertia weight: 0.8, maximum velocity: 30  $m/s$ ) [Tam, 2021] and a clipped speed of 30  $m/s$ .

The **MCTS Parameter Analysis** in E2, included testing various parameters such as the number of steps per layer (ranging from 200,000 to 1,000,000) and the exploration constant  $C$  (evaluated at 0.5, 0.8, and 1.0). KPIs

like the Amount of Visits per Layer and Average Reward per Layer were analyzed to gauge the search strategy’s effectiveness under different parameter settings.

In the **Visual Analysis of Emergent Properties and Metric Quantification via MCTS** in A3, both successful and unsuccessful MCTS simulations were examined. This involved visually inspecting trajectory plots and decision-making patterns of MCTS to identify emergent behaviors. The aim of this experiment was to qualitatively assess these behaviors and to determine quantified metrics. These metrics are important in guiding the development of the utility function, providing insights into the characteristics of successful and unsuccessful runs.

In the **Visual Analysis of Emergent Properties and Metric Quantification via MCTS** conducted in A4, both successful and unsuccessful runs were examined. This involved visually inspecting trajectory plots and decision-making patterns from the simulation output of MCTS.

The last experiment related to the Multi Transmitter Search, consisted of **comparative tests between MUC and MCTS** in E5. The methods were deployed in altered Global Setups. Two variations were done, introducing: i) variation in agent initialization, and ii) variation in transmitter positioning. For each of the two variations a total of 100 runs were conducted for both MUC and MCTS. To implement these variations, different Sobol initialization seeds were used for the placement of the 20 agents and the 6 transmitters. The transmitters were located within a central 8x8 km area, following findings from exploratory experiments that transmitters positioned near borders were hardly ever found due to limitations in encapsulation capability. Important KPIs (detailed in Appendix 5) such as mean values for Total Time Steps Required, Value Score, and the amount of stationary and altering decision moments were used to measure efficiency, speed, and adaptability.

## Results

This section describes the results of each experiment related to the Multi Transmitter Search.

The **MCTS Parameter Analysis**, experiment E2, revealed significant insights. The study, as detailed in Table 4, involved varying the steps per layer and the exploration factor  $C$ . A notable finding was that increasing the steps per layer generally improved the decision path value, indicating enhanced solution discovery capabilities. However, this also led to a higher visit count at the initial node but a decrease in its average value, suggesting a complexity in balancing search depth and efficiency. The exploration factor  $C$  emerged as a critical element, with a setting of 0.5 and 1 million steps per layer yielding the most favorable outcomes in terms of the best path value. These results guided the optimal MCTS configuration for subsequent simulations, aiming to optimize the balance between thorough exploration and efficient decision-making.

Table 4: MCTS setting runs results

Run	Steps per Layer	C	# Layers	# Visits (1st layer)	Mean value (1st layer)	Decision path value score	Best path value score
1	200000	1	21	2	1.00	2.14	3.44
2	400000	1	20	6	0.3933	4.08	4.4
3	600000	1	21	8	0.17	3.9	4.36
4	800000	1	21	8	0.17	3.98	4.36
5	1000000	1	17	10	0.114	4.38	4.46
6	200000	0.8	21	2	1.00	2.32	3.44
7	400000	0.8	21	5	0.436	3.42	4.14
8	600000	0.8	18	8	0.065	4.3	4.32
9	800000	0.8	21	11	0.1	3.62	4.1
10	1000000	0.8	21	13	-0.2231	3.76	4.3
11	200000	0.5	21	2	1.00	0	3.44
12	400000	0.5	21	8	-0.17	3.78	4.22
13	600000	0.5	20	11	-0.4909	4.02	4.4
14	800000	0.5	21	15	-0.6267	3.86	4.24
<b>15</b>	<b>1000000</b>	<b>0.5</b>	16	16	-0.5175	<b>4.46</b>	<b>4.48</b>

In the **Visual Analysis of Emergent Properties and Metric Quantification via MCTS**, analysis A3 in Table 2, two distinct scenarios were evaluated. The first was a successful case where all six transmitters were located within 1800 timesteps. The second was an unsuccessful attempt, failing to locate any transmitters within 4000 timesteps. These outcomes are visualized in Figure 16. This analysis was not only focused on trajectory plots but also included visual examination of the dynamic simulation visualizations.

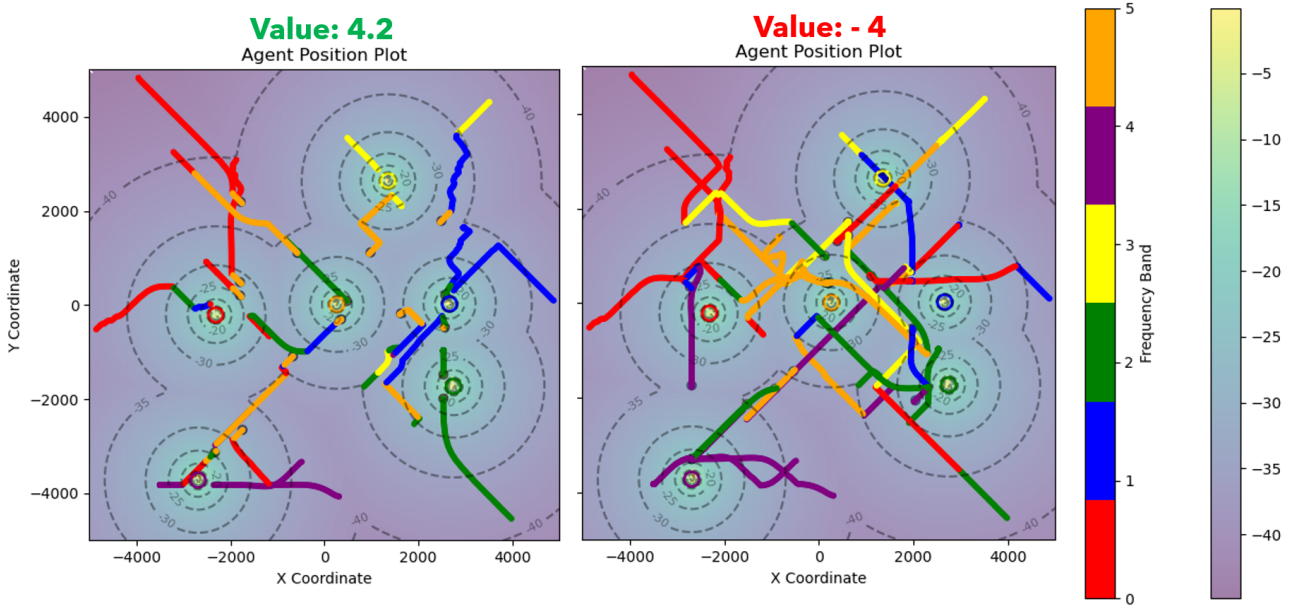


Figure 16: Trajectory plots of good (left) and poor (right) performing solutions. Different colors in the trajectories indicate the frequencies, ranging from 3.7 GHz to 4.3 GHz, under which an agent has traveled. The colored circles represent the respective RT of each transmitter, emitting unique frequencies. The warmth of the color gradient reflects the RSS levels in mdB, providing a visual representation of signal strength variations.

The discovered metrics from the analysis are the following:

*Maximum RSS:* This metric evaluates the highest received signal strength, in mdB, within each coalition. Successful scenarios featured agents recording high signal strengths, a direct result of their closeness to transmitter centers. In contrast, the unsuccessful attempt was marked by weaker signals, indicative of a less effective performance.

*Agent Similarity:* This is determined by counting the instances within a coalition where the distances between agents are within a set threshold of 300 meters. Successful runs showed lower similarity until just before detection, suggesting efficient area coverage through unique agent routes. However, the unsuccessful run revealed higher similarity, characterized by overlapping agent paths, pointing to less effective search patterns.

*Encapsulation:* This metric,  $\vec{v} \cdot \hat{d}$ , assesses how well an agent aligns its movement with the direction of the previously best-performing agent. It begins with the direction vector  $\vec{p}_{\text{prev\_best}} - \vec{p}_{\text{best}}$ , normalized to  $\hat{d}$ . The alignment's degree is quantified by the dot product of  $\hat{d}$  and the current best agent's normalized velocity vector  $\hat{v}$ , shown as  $\hat{v} \cdot \hat{d}$ . Successful scenarios exhibited efficient encapsulation, with agents clustering around transmitters, whereas unsuccessful ones showed poor encapsulation, with agents failing to effectively converge on target areas. This encapsulation pattern is depicted in Figure 17, highlighting the best agent (black) moving towards the previous best agent (red).

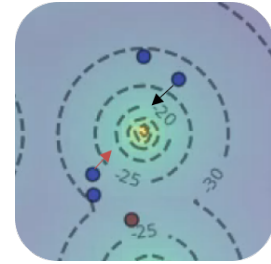


Figure 17: Visualization of the best agent (black) moving towards the previous best agent (red).

Further analysis of these metrics, as shown in Figure 18, provided insights into their expected behaviors in successful and unsuccessful MCTS executions. For Maximum RSS, successful runs were characterized by stable or increasing values, while unsuccessful runs displayed a decreasing and unstable trend. In the case of Agent Similarity, lower values leading to convergence were indicative of successful runs, whereas high values in unsuccessful runs pointed to ineffective search strategies. Lastly, strong encapsulation throughout was observed in successful runs, contributing significantly to locating targets, a contrast to the unsuccessful runs where encapsulation diminished, preventing effective coalition convergence.

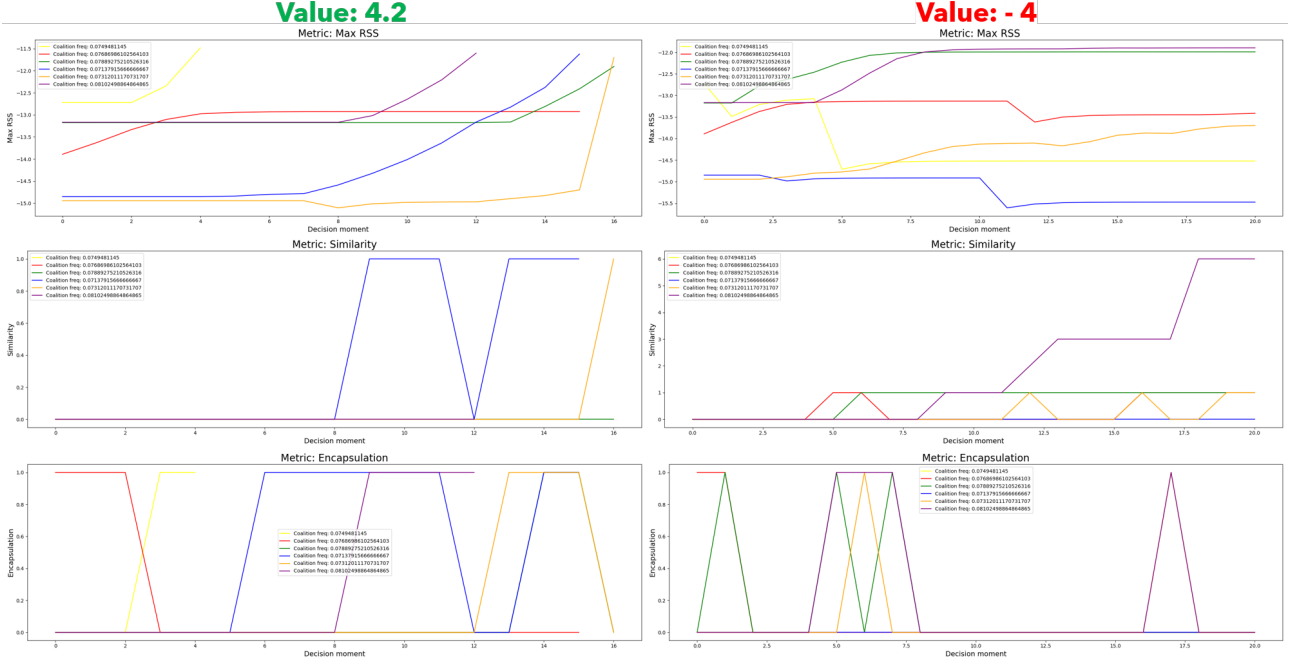


Figure 18: This figure displays the output metrics, including Maximum RSS, Agent Similarity, and Encapsulation. The left side shows a successful MCTS run, while the right side depicts an unsuccessful one. Each colored line in the diagram represents a coalition within the swarm, with each coalition assigned to a specific frequency.

For the **MUC - Utility Defining and Analysis**, in A4, a utility function was developed to quantify and guide agent behaviors based on specific metrics, aiding in decision-making aligned with desired outcomes. This utility function, as detailed in Equation 1, integrates the following metrics:

1. Maximum RSS: The RSS values are normalized for integration with other metrics. The normalization formula is:

$$\text{normalized\_RSS} = \frac{\text{RSS} - \text{min\_RSS}}{\text{max\_RSS} - \text{min\_RSS}}$$

Gaining a utility of 0.5 if the current RSS equals or surpasses the previous value.

2. Agent Similarity: This is quantified by the function  $S(c)$ , which counts agent pairs within a threshold distance. The function is defined as:

$$S(c) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N 1(|p_i - p_j| \leq 300)$$

The similarity penalty is computed as:

$$\text{similarity\_penalty} = \frac{\text{similarity\_count}}{\text{total\_pairs}}$$

3. Encapsulation: Measured using the dot product of the direction vector and the agent's velocity vector:

$$\text{dot\_result} = \vec{v} \cdot \vec{d}$$

The utility from encapsulation is calculated as the ratio of encapsulating agents to the total number of coalitions:

$$\text{encapsulation\_utility} = \frac{\text{encapsulation\_count}}{\text{total\_coalitions}}$$

These metrics are aggregated into a unified utility value through a weighted sum, ensuring each metric proportionally contributes to the utility, allowing for a balance between different behaviors and outcomes. For details on the utility function implementation of found metrics in multi-transmitter PSO subswarming, refer to Algorithm 12.

After implementing the utility function based on the found metrics, a solution is found and illustrated by the trajectory plot in Figure 19, which shows high stability in its ability to generate solutions. It experiences fewer fluctuations compared to the MCTS approach and successfully identifies five out of six transmitters. However, it appears to make a suboptimal decision early on by switching the agents assigned to the yellow frequency to blue and orange frequencies. This switch negatively impacts the potential to discover the yellow transmitter. A closer examination is required to identify the reason behind this choice and to develop a metric to refine the decision-making process encoded within the utility function.

According to the data in Figure 20, the utility function seems to effectively incorporate the selected metrics. The Max RSS values show a desired pattern of consistent and uninterrupted increases, which align with expectations of a successful search strategy. Low similarity between agents is noticed, with the exception when a coalition is nearing the discovery of a solution. Frequent occurrences of encapsulation suggest that this mechanism is a contributing factor in the successful detection of transmitters by the coalitions.

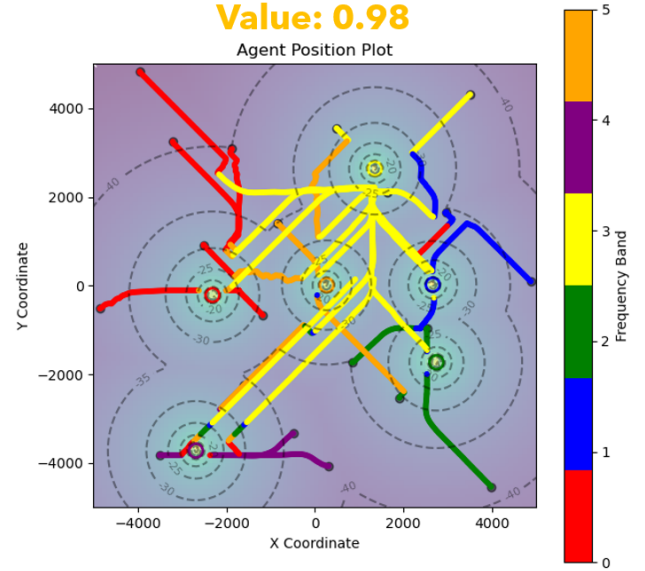


Figure 19: Trajectory map of the MUC solution.

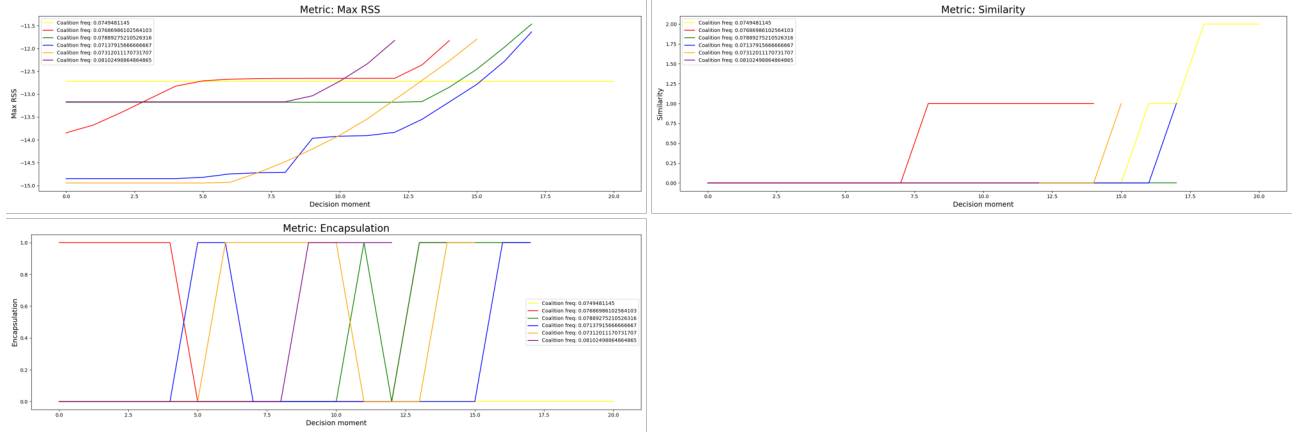


Figure 20: Metric outputs of the MUC solution, showcasing Maximum RSS, Agent Similarity, and Encapsulation metrics. Each colored line in the figure represents a distinct coalition within the swarm, with each coalition being assigned a unique frequency.

In the **comparative tests between MUC and MCTS (E5)**, the results show the outcome across varying agent initialization and transmitter positioning. Analyzing the data presented in Table 5, it is evident that there are notable differences in performance between the MUC and MCTS methods. In terms of locating transmitters, MUC found an average of 4.89 transmitters in agent initialization variations and 5.23 in transmitter positioning variations, whereas MCTS consistently located all six transmitters in both cases. This demonstrates MCTS's superior efficacy in discovery, as it not only consistently located more transmitters but also required significantly fewer steps to do so, with averages of 1820 and 1774 steps for agent and transmitter positioning, respectively, compared to MUC's 3749 and 3537 steps. From this it can also be noted that both methods had more difficulty with the agent initialization than with the transmitter positioning. This was probably especially the case when agents were initialized closely together, as spread is required for good PSO Search.

To accurately compare the speed of mission completion between the two methods, it is necessary to exclude the MUC runs where 4 to 5 transmitters were located. This is due to the 4000-time cap on unsuccessful runs, which could otherwise distort the analysis. After removing these runs, 44% of the runs with varying transmitter positions and 29% with varying agent initialization remain, as detailed in Appendix 7.2.3. This filtered data indicates that the MCTS method is about 45% faster than the MUC method for swarm initialization variation, and around 46% faster for transmitter position variation in successful runs.

Additionally, the table shows that MUC, even with more decision moments, underwent fewer coalition changes compared to MCTS. This is shown by the relatively large stationary/altering ratio for MUC, in contrast to the

smaller ratio observed for MCTS. This suggests that while MUC possesses leans towards stability, which was anticipated to benefit PSO Search, it did not translate into enhanced performance outcomes.

These insights highlight the necessity for a more in-depth analysis of MUC’s decision-making process, considering the variability in transmitter detection observed across different runs. Future analysis, as planned in Appendix 7.2.1, is focused on examining these elements more closely to refine and enhance the MUC method.

Table 5: Mean Performance Overview of MUC and MCTS for varying agent initialization and transmitter positioning

Experiment	Transmitters Found	Total Steps	Stationary	Altering	Value Score
Agent Initialization: MUC	4.89	3749	27.48	9.78	1.14
Agent Initialization: MCTS	6.00	1820	0.77	16.98	4.18*
Transmitter positioning: MUC	5.23	3537	27.61	7.49	1.69
Transmitter positioning: MCTS	6.00	1774	0.79	16.52	4.23*

\* denotes  $p < 0.05$ , indicating statistical significance.

### Statistical Analysis

The study used statistical methods to analyze the data. The Shapiro-Wilk test [SHAPIRO and WILK, 1965], was used to assess normality, along with QQ-plots. These indicated to not assume normality and to take a conservative approach. For each simulation, the coefficient of variation was calculated to establish whether the amount of runs were enough. Because the data was assumed to be not normally distributed, the Wilcoxon Signed-Rank Test [Rey and Neuhäuser, 2011] was chosen. This paired non-parametric test was used for identifying differences in value score, which is built up out of transmitters found and step time.

In the varying agent initialization scenario, the p-value of  $4.00 \times 10^{-18}$  indicates an almost uniform direction of effect across the dataset. The near-perfect effect size of 0.9998 corroborates the profound impact of swarm initiation variation on value scores. Similarly, for varying transmitter positioning, a p-value of  $4.25 \times 10^{-18}$  and an effect size of 0.9994 indicate a significant and substantial effect on value scores due to transmitter position variations. A complete overview of the statistical analysis can be found in Appendix 6.2.

### 6.3 Communication Line Mapping & Interference Point Finding (E6, E7)

The experiments related to the mission tasks of Communication Line Mapping and Interference Point Finding were designed to assess the capabilities of different algorithms. Within Communication Line Mapping, the focus was on evaluating the Boustrophedon, Spiral, and Tabular QL algorithms for their precision and coverage in mapping detected communication lines. This assessment aimed to identify the most effective approach for comprehensive and accurate mapping. Conversely, the experiments in Interference Point Finding concentrated on the efficiency and accuracy of various methodologies for locating interference points. The emphasis in these experiments was on measuring time effectiveness in determining the location and the precision of the localization, thereby evaluating the algorithms’ efficiency in operational contexts.

#### Experimental Setup

The experiments related to the mission tasks Communication Line Mapping and Inference Point Finding were conducted in the Base Setup, depicted in Figure 3. Each mapping algorithm utilized two agents with a maximum velocity of 30  $m/s$ . The experiment utilized Boustrophedon, Spiral, and Tabular QL algorithms, each starting from a uniform and randomly distributed initialization across the search area. The setup involved 8 agents, each moving at a velocity of 30  $m/s$ . Specific configurations for Boustrophedon and Spiral included a helper offset of 1000 m, with Boustrophedon also using a  $5^\circ$  turn angle, and Spiral employing a shrink factor of 0.97. The Tabular QL algorithm underwent pre-training of 200,000 episodes, followed by 30,000 training episodes in the NLR-EW environment, as shown in Figure 3. Alongside these mapping algorithms, the RL-Interference Point Finding algorithm was also evaluated. This algorithm differed from the others as it focused solely on locating the interference point directly, bypassing the preliminary mapping phase. The RL-Interference Point Finding algorithm was trained with a reward structure optimized to efficiently guide the agent to the interference point. The detailed setup and training parameters for the RL-Interference Point Finding algorithm are outlined in section Appendix 3.4.

The KPIs (detailed in Appendix 5) used to evaluate the algorithms included Mapping Step Time (Steps) for speed, Localization Error (Meters) for accuracy in interference point determination, Visited Cells Count (Number of Cells in Figure 21 for coverage, Multiple Visits to Cells (Number of Revisited Cells in Figure 21)



for redundancy, Swarm Energy Use ( $m/s^2$ ) for energy efficiency, and Compute Time (Seconds) for processing efficiency.

The procedure entailed conducting 100 runs for each algorithm, terminating when the mapped area ceased to expand or after a pre-set duration. For the RL-Interference Point Finding simulations, runs stopped when an interference point was determined. Runs that failed to fully map the area or resulted in crashes were excluded from the main analysis but were considered when evaluating algorithm robustness.

## Results

In the Communication Line Mapping experiment, E6, the performance of Boustrophedon, Spiral, and Tabular QL algorithms was compared using various performance metrics. The comparison, detailed in Table 6, highlights that Boustrophedon proved to be the fastest, completing the task in an average of 2246 steps (or 281 seconds), and had a high 'New Cells per Step' value of 0.73, suggesting efficient area coverage. However, there's potential for further speed enhancement through parameter optimization.

Spiral, while achieving the densest coverage with the highest 'Visited Cells Count' of 2185 cells, was the least energy-efficient, consuming 2778077 ( $m/s^2$ ). Adjustments in its parameters could improve energy efficiency without compromising its thorough mapping capability.

Tab-QL demonstrated high accuracy in localizing interference points with a Localization Error of only 14 meters but had limitations in extensive area coverage, as seen in its 'Visited Cells Count' of 1327 cells. Additionally, it performed well in avoiding redundant mapping, as shown by a 'Multiple Visits to Cells' value of 221 cells.

The mapping patterns, illustrated in Figure 22 and Figure 23, reveal further insights. While Spiral provided dense coverage, its mapping along the edges was somewhat imprecise. Tabular QL, though precise, sometimes became trapped in a loop due to its limited view, impacting its overall area coverage. Figure 21 shows the area mapping process, highlighting wide coverage with some overshoot and concentrated mapping activities in the cone's narrower section.

Overall, Boustrophedon offers efficiency in Communication Line Mapping, effectively covering new areas with each movement. The Spiral method, in contrast, provides thorough coverage but at a higher energy cost, as it requires more time and results in more repeated visits to the same cells. Meanwhile, Tab-QL stands out for its accurate localization of the interference point and its lower frequency of revisiting cells. It is not dependent on foreknowledge of the agent positions but requires adjustments to achieve broader area coverage, given its limited field of view.

Table 6: Mean Value KPIs for Comparison Between Boustrophedon, Spiral, and Tabular QL Mapping Algorithms and Interference Point Finding.

Algorithm	Mapping Step Time	Localization Error [m]	Visited Cells Count	Multiple Visits to Cells	Energy Use [ $(m/s)^2$ ]	New Cells per Step	Compute Time [s]
Boustrophedon	2246*	126	1638	359	2020255*	0.73	1.79*
Spiral	3118	110	2185*	435	2778077	0.70	2.35
Tab-QL	2346	14*	1327	221*	1951424*	0.57	41
RL-Interference Point Finding	184	174	-	-	58026	-	0.11

\* denotes  $p < 0.05$ , indicating statistical significance.

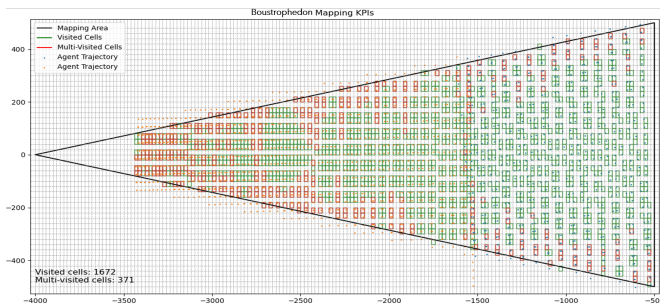


Figure 21: 20x20 grid cells (Boustrophedon)

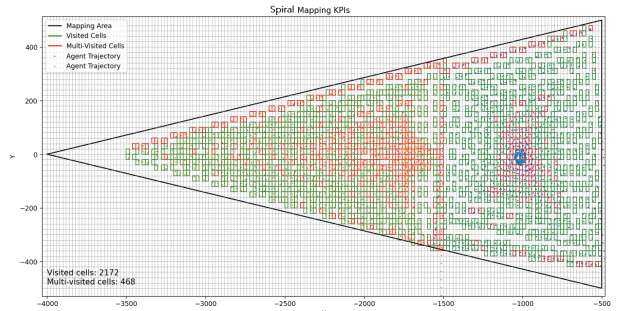


Figure 22: 20x20 grid cells (Spiral)

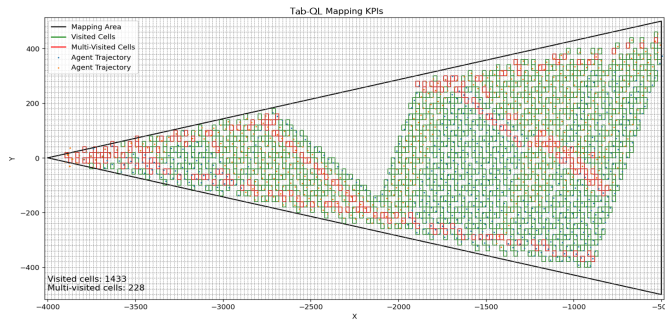


Figure 23: 20x20 grid cells (Tabular QL)

In the Interference Point Finding experiment, E7, the performance of various algorithms was compared with a focus on locating interference points efficiently. As indicated in Table 6, the RL-Interference Point Finding algorithm stood out for its quickness and low energy consumption, a result of its direct approach to finding the interference point. However, this method compromised on accuracy, as RL-Interference Point Finding had the highest error rate in localization. In contrast, mapping algorithms like Tab-QL demonstrated higher precision in locating interference points, albeit at a slower pace and with greater resource usage. This contrast highlights the trade-offs between speed, resource efficiency, and accuracy in Interference Point Finding strategies.

This analysis shows that in Interference Point Finding, the RL-Interference Point Finding method is much faster, needing 91.8% fewer steps than Boustrophedon. However, it is less accurate, having a localization error of 184 m. On the other hand, for tasks where precise localization is key, mapping methods like Tab-QL are a better choice. They provide more accurate results with a 14 m error but require more resources.

### Statistical Analysis

The study used statistical methods to analyze the data. The Shapiro-Wilk test [SHAPIRO and WILK, 1965], was used to assess normality, along with QQ-plots. These indicated to not assume normality and to take a conservative approach. For each simulation, the coefficient of variation was calculated to establish whether the amount of runs were enough. Because the data was assumed to be not normally distributed, the Wilcoxon Signed-Rank Test [Rey and Neuhäuser, 2011] was chosen as the data was paired.

In the statistical analysis conducted using the Wilcoxon Signed-Rank Test at an alpha level of 0.05, several KPIs of mapping algorithms were compared, yielding insightful results. The comparison between Boustrophedon and tab-QL mapping algorithms revealed no significant difference in mapping time, as indicated by a p-value of 0.6344. However, when examining localisation error between Spiral and tab-QL, a significant difference was noted, with a p-value of  $2.39 \times 10^{-11}$ . In the aspect of visited cells count, Boustrophedon and Spiral showed a notable difference (p-value:  $1.78 \times 10^{-17}$ ), highlighting variations in their mapping efficiency. The analysis also revealed a significant discrepancy in the number of multiple visits between Boustrophedon and tab-QL, as evidenced by a p-value of  $2.38 \times 10^{-11}$ . Additionally, while the energy usage between Boustrophedon and tab-QL did not show a significant statistical difference (p-value: 0.0968), both performed significantly better than Spiral. Lastly, the computational time between Boustrophedon and Spiral was found to be significantly different, with a p-value of  $2.44 \times 10^{-17}$ , underscoring distinct operational efficiencies. These findings, focusing on specific KPIs, effectively highlight the performance contrasts between the algorithms. For a complete overview, further details can be found in Appendix 6.3.

## 7 Supplementary Experiments & Analyses

This section presents an overview of the supplementary experiments and analyses. To better understand the methodologies and their effectiveness, various analyses were performed, which are detailed in Table 7. This table provides details of each experiment, categorizing them by ID, the type of experiment conducted, the specific property being examined, the method used, and the experimental setup. The experiments included examining the sensitivity of algorithm parameters, a deeper dive into the utility function in for coalition formation in Multi Transmitter Search. For a full account of all the experiments and analyses, refer to Appendix 7.



Table 7: Overview of Supplementary Experiments and Analyses

ID	Experiment Type	Property	Method	Setup
Se1	Sensitivity Analysis	Search Area $20km^2$	PSO Search, Pair-gradient Search	Modified Base Setup
Se2	Sensitivity Analysis	Sobol Initialization	PSO Search, Pair-gradient Search	Base Setup
Se3	Sensitivity Analysis	Agent Count Variations	PSO Search, Pair-gradient Search	Modified Base Setup
Se4	Sensitivity Analysis	Coefficients Adjustment	PSO Search	Base Setup
Se5	Sensitivity Analysis	Follower Offset and Velocity	Pair-gradient Search	Base Setup
Se6	Sensitivity Analysis	Helper Agent Offset	Boustrophedon	Base Setup
Se7	Sensitivity Analysis	Turn Angle Variation	Boustrophedon	Base Setup
Se8	Sensitivity Analysis	Agent Velocity	Boustrophedon, Spiral	Base Setup
Se9	Sensitivity Analysis	Shrink Factor	Spiral	Base Setup
Se10	Sensitivity Analysis	Border Angle Variation	Spiral	Base Setup
Se11	Sensitivity Analysis	Pre-train	Tab-QL	Simple Triangle Grid
Se12	Sensitivity Analysis	EW Train	Tab-QL	Base Setup
Ma1	Methodological Analysis	Utility Function Refinement	MUC	Global Setup
Ma2	Methodological Analysis	Metric Exploration for PSO subswarming	MUC	Global Setup

Experiments with notable outcomes like Se2, Se4, Se11, and the methodological analyses Ma1, Ma2 will be briefly discussed. To enhance agent initialization for the Single Transmitter Search methods, the **Sobol Initialization (Se2)** was implemented. This was due to the PSO Search and Pair-gradient Search methods aiming at improving the initial positioning of agents to address premature convergence. Notably, this modification in PSO Search led to a significant improvement in its performance, with the implementation of Sobol initialization and parameter fine-tuning increasing the success rate to 100%. This indicates that a refined initialization strategy can effectively reduce non-converging solutions. For the Pair-gradient Search, while there was a slight increase in energy usage, these outcomes underscored the importance of initial agent distribution on the efficiency of these algorithms.

The second notable analysis was an improvement on the PSO Search **Coefficients Adjustment (Se4)**. The objective of this analysis was to adjust the algorithm’s parameters to better balance convergence and exploration. This tuning of parameters effectively optimized the step time to reach the RT and eliminated unsuccessful searches, demonstrating the significance of parameter adjustments in optimizing algorithm performance.

Another contributing analyses was focused on **Pre-train (Se11) for Tabular Q-Learning**. This involved pre-training the algorithm in a simpler environment to improve its performance in more complex settings. This approach significantly improved the mapping thoroughness, illustrating the value of learning fundamental actions before deploying in a more complex and scaled environment.

Also the **Utility Function Refinement in Multi Transmitter Search (Ma1)** led to a notable outcome. The analysis focused on improving decision-making in frequency switching. The utility function was revised to include a penalty for premature switching by outer agents in a cluster, represented by the formula:

$$\Delta\text{Utility} = - (\text{Penalty}_{\text{switch}} \times N_{\text{outer agents in cluster switching}})$$

This adjustment led to more efficient searches, evidenced by the successful detection of all targets with reduced steps and improved energy use.

Another contributing analyses was the **Metric Exploration for Subswarming in PSO Search (Ma2)**. This analysis involved identifying new potential metrics to enhance subswarming coalition formation. This qualitative study produced coalition-based and cluster-based metrics, laying a foundation for more reliable outcomes in Multi Transmitter Searches.

## 8 Discussion

The methods explored have shown sufficient performance in various domains (described in section 2). The results of this research (shown in section 6) have demonstrated effectiveness of the methods in CEMA environments. This section will discuss the limitations of these methods in this specific context and the broader implications of the research. First the Single Transmitter Search methods will be discussed, followed by the Multi Transmitter Search and by the Communication Line Mapping & Interference Point Finding. Finally, the implications of the assumptions in this research are discussed.

In the first place, **Single Transmitter Searches** methods were evaluated, specifically PSO Search and Pair-gradient Search. The PSO Search proved to be fast in detecting the RT, but its occasional failure to converge suggests the need for improvements. The sensitivity analysis revealed the potential of Sobol initialization to improve PSO Search’s success and efficiency. On the other hand, the Pair-gradient Search, although slower due to its pairing process, performed effectively with fewer agents. The importance of triangulation and gradient calculation accuracy was emphasized in its sensitivity analysis. While incorporating the inverse square law into gradient calculations could aid Pair-gradient Search, the algorithm may still be impacted by local signal

disruptions. Moreover, the Pair-gradient Search appeared to be more energy-efficient. The energy efficiency of Pair-gradient Search is partially due to the leader agent’s ability to hover and navigate directly towards the transmitter, theoretically requiring no energy at zero velocity. However, it is important to acknowledge that hovering in real-world scenarios does consume energy. A hybrid approach, combining the strengths of the PSO Search and Pair-gradient Search, could be promising. This approach could benefit from the gradient search’s ability to prevent premature convergence while relying less on local gradient determination. Also, supplementary experiments could investigate fall-out resilience, leveraging the strengths of swarm intelligence.

Secondly, in the context of the **Multi Transmitter Search**, the PSO Search was selected as the velocity controller. The sensitivity analysis for the PSO coefficients (Se4) demonstrated that agents could avoid premature convergence due to momentum. However, this property is not considered useful for Multi Transmitter Search, as it relies on local gradient; thus, the initial PSO parameters were retained. Regarding the MCTS, which guides coalition formation, it was observed that the best node selection does not always result in the best outcome due to its stochastic nature. Reducing exploration in higher nodes and implementing a decreasing cap on steps per layer could improve the MCTS. Additionally, deeper decision tree rollouts often led to successful outcomes, suggesting potential underlying trends. The HURMS framework showed that the distinction between good and bad simulations per metric can be ambiguous, indicating that metrics are not always definitive. Interactions between metrics sometimes contaminate outcomes. For example, maximizing RSS Max for each coalition occasionally led to suboptimal results due to premature convergence. Furthermore, metrics can influence each other, potentially contaminating outcomes. Initiating the HURMS with fewer overlapping metrics might reduce complexity and clarify effects. Despite these challenges, the current metrics appear to positively impact performance, showing promise for the method in this PSO subswarming coalition formation use case. Extending this approach to other search applications, including UAV-based multi-objective search and rescue operations or wildlife monitoring, could be advantageous. Moreover, applying these techniques to domains involving sequential decision-making, such as energy grid management and traffic systems, may also be promising.

In the third place, this research evaluated the **Communication Line Mapping** methods, which include the Boustrophedon, Spiral, and Tabular QL mapping methods. The Boustrophedon method’s simplicity ensures coverage, but it requires customization and pre-knowledge about starting points, posing limitations in multi-agent coordination. The Spiral mapping method, while adaptable and precise in known environments, faces scalability challenges due to its need for extensive coordination and customization, especially as the number of agents increases. The Tabular QL, anticipated to be effective for adaptive multi-agent mapping, confirmed its adaptability and scalability in the research by [Nakanishi et al., 2021]. It stands out for not requiring foreknowledge about agents’ relative position to the communication line. Additionally, the importance of curriculum learning in preparing agents for complex mapping tasks emerged. To improve effectiveness, future experiments could focus on addressing its limitations in the field of view, which currently leave some areas uncovered. It is essential to recognize that the Tabular QL utilizes grid-based actions and observations. The cell size matches the maximum velocity and oriented approximately perpendicular to the communication line. In assessing the performance of all mapping algorithms, a separate consideration is the selection of grid cell size, shape, and orientation for evaluating KPIs. Here, a cell size of 20  $m$  was adopted, smaller than the maximum velocity of 30  $m/s$ , to minimize the risk of skipping cells. While these grid parameters were chosen for a reason, they have not been extensively tested. The grid choice could influence the effectiveness of the mapping algorithms and present areas for potential future exploration. The study also explored the RL-Interference Point Finding for the Interference Point Finding. The results showed that while it is faster, it is less precise than mapping-based methods. However, the algorithm could be refined to improve its accuracy and fully capitalize on its speed. The RL-Interference Point Finding method has a higher error rate and is more susceptible to noise and signal disruptions in real-world scenarios. Conversely, the mapping methods might offer better robustness, as they provide comprehensive data collection before localization.

Lastly, it is crucial to consider the **real-world applicability of these simulation-based findings**, especially as we step into a relatively undiscovered research domain. While the study aimed to align the simulated environment with real-world scenarios, necessary assumptions were made, as detailed in Appendix 1.3, to manage the experimental workload and excel in algorithm development. These assumptions are related to real-world environment, operations, and simulation. The environmental model using the NLR-EW Toolbox, did not include critical factors like refraction, reflection, environmental interference, and atmospheric conditions, which significantly affect signal propagation and detection accuracy. The simulation model in this research assumes instant UAV communication and employs a sequential time-step approach, which may not accurately reflect real-world complexities like communication delays and asynchronous decision-making. It also simplifies transmitter communication, not accounting for the discontinuity in real-world signal interactions. The UAV operations approach in this research shows a disparity between simulated and actual conditions, particularly in energy use, communication, and signal processing. Assumptions in UAV dynamics and sensing, while efficient for simulation,

may not fully represent real-world complexities, including energy limitations and environmental factors. The need for UAVs to adapt to disruptive environments emphasizes the importance of self-awareness, as discussed in Appendix 4.3.4, and the development of confidence metrics. Extended research is crucial for understanding UAV adaptability in challenging conditions, especially in maintaining positioning and RSS accuracy. These reflections highlight the need for ongoing development and refinement of simulation models to more accurately mirror real-world conditions and challenges. A more detailed discussion on the implications of these assumptions is presented in Appendix 4.3.

## 9 Conclusion and Future Work

This research developed an agent-based model to research the defined CEMA mission objective of deploying a UAV swarm operating to selectively interfere with communication lines in a CEMA environment. The mission tasks supporting this objective are: i) Transmitter Search, ii) Communication Line Search, iii) Communication Line Mapping & Interference Point Finding. Furthermore, this research contributes in assessing the most effective methodologies for each of these mission tasks. The four most important conclusions are the following.

The first conclusion regarding the Single Transmitter Search, is that the PSO Search is an effective method in fast transmitter detection. The PSO Search was approximately 38% faster than the Pair-gradient Search in locating a single transmitter, but its reliability in convergence is less certain. The Pair-gradient Search demonstrated more energy efficient operations and initially had a higher success rate. Further analysis found two adjustments to improve the success rate of PSO Search to 100%, which are the Sobol initialization and the improved parameter selection. The PSO Search’s independence from local gradients suggests higher robustness against real-world environmental factors, like signal noise or other disruptions. All this led to finding the PSO Search as the best fit for Single Transmitter Search. Due to the distributed character of the PSO Search, it serves as a velocity driver for the Multi Transmitter Search.

Secondly, for the Multi Transmitter Search task, this research showed that using the HURMS framework in PSO subswarming works for getting successful solutions with the MUC method. This framework is an insightful real-time alternative compared to the RL-MCTS method. The MUC method, utilizing this framework, was able to be successful and find all transmitters, although sometimes it was less effective due to early changes in frequency by agents. Important metrics in the utility function, like maximum RSS, agent similarity, and encapsulation, guided the MUC method in choosing strategic coalition formations. In the comparative analysis between MUC and MCTS, MCTS exhibited a 56-71% higher success rate in locating all transmitters. In the successful runs it also completed the task 45-46% faster. This shows that while MUC is capable of finding successful solutions, however, it falls short in time efficiency and success rate compared to MCTS’s consistent performance. Therefore, the results indicate that the MUC algorithm needs more work to reach or surpass the performance levels of the benchmarking MCTS in managing the swarm. Further utility refinement can be inspired by the qualitative analysis of potential valuable metrics for subswarming with the PSO Search in Appendix 7.2.2.

Thirdly, for Communication Line Mapping, this research found the Boustrophedon method to be the most time-efficient compared to the Spiral and Tab-QL method. This method stands out due to its ability to cover new areas more effectively with each movement, leading to faster task completion and efficient area coverage. In contrast, the Spiral approach provided the most comprehensive coverage, visiting 33% more cells than Boustrophedon. However, this broader coverage demanded more time and led to 21% more repeated visits to the same cells, particularly in central areas of the communication lines. On the other hand, Tabular Q-Learning was noteworthy for its accurate localization of the interference point and a lower frequency of revisiting cells, resulting in precise mapping. Nonetheless, its limited field of view (FoV) resulted in some uncovered areas. Additionally, Tabular Q-Learning’s adaptability in multi-agent contexts is a significant advantage, as it does not require prior knowledge of agent positions. In the fourth place, for the Interference Point Finding, RL-Interference Point Finding proved to be the quickest method, needing 91.8% fewer steps than Boustrophedon, which was the fastest among mapping methods. However, its localization accuracy was higher, with a 184 *m* error compared to the 14 *m* error of Tabular Q-Learning, which was the most precise mapping method for the Interference Point Finding.

Synthesizing this, the most effective methods for developing a swarm intelligence model for each mission task are the PSO Search for Single Transmitter Search, the HURMS framework combined with the MUC and the MCTS for the Multi Transmitter Search, the Boustrophedon method for Communication Line Mapping, and the Tabular Q-Learning for precise localization in the Interference Point Finding. These methods successfully deploy a UAV swarm operating to selectively interfere with communication lines in a CEMA environment.

Future research holds the potential for advancements in mission task methodologies and in enhancing real-world environmental robustness. Introducing self-awareness amongst agents, refer to Appendix 4.3.4, could significantly improve system performance in uncertain conditions. For coalition formation in PSO subswarming,

integrating the RL-MCTS AlphaZero methodology could enable real-time behavioral optimization of agents through learning and adaptation. Additionally, utilizing these neural networks trained on swarm simulation data may enhance decision-making and refine subswarming strategies. However, it's important to note that the insights gained from combining the MCTS and the MUC within the HURMS framework are not inherently present and require the application of explainable AI approaches to be realized. Furthermore, future research should consider tackling complexities in 3D mapping. Enhancing 3D mapping resolution, possibly through techniques like the COMA algorithm, could address issues related to mapping quality and effective credit assignment among agents.

## References

- [sta, ] Stable baselines 3. <https://stable-baselines3.readthedocs.io/en/master/>. Accessed: 2023-04-18.
- [Abd, 2021] (2021). Aerial swarms: Recent applications and challenges. *Current Robotics Reports*, 2:309–320.
- [Den, 2023] (2023). A distributed collaborative allocation method of reconnaissance and strike tasks for heterogeneous uavs. *Drones*, 7.
- [Arnold et al., 2020] Arnold, R., Jablonski, J., Abruzzo, B., and Mezzacappa, E. (2020). Heterogeneous uav multi-role swarming behaviors for search and rescue; heterogeneous uav multi-role swarming behaviors for search and rescue.
- [Arnold et al., 2018] Arnold, R., Yamaguchi, H., and Tanaka, T. (2018). Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. *International Journal of Humanitarian Action*, 3:18.
- [Baldazo et al., 2019] Baldazo, D., Parras, J., and Zazo, S. (2019). Decentralized multi-agent deep reinforcement learning in swarms of drones for flood monitoring. *27th European Signal Processing Conference (EUSIPCO)978-9-0827-9703-9/19/©2019 IEEE*.
- [Changder et al., 2019] Changder, N., Aknine, S., and Dutta, A. (2019). An effective dynamic programming algorithm for optimal coalition structure generation. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 721–727.
- [Cho et al., 2022] Cho, S., Park, J., Park, H., and Kim, S. (2022). Multi-uav coverage path planning based on hexagonal grid decomposition in maritime search and rescue. volume 10. MDPI.
- [Derr and Manic, 2009] Derr, K. and Manic, M. (2009). Multi-robot, multi-target particle swarm optimization search in noisy wireless environments. Idaho Falls, ID, USA. Idaho National Laboratory Department of Computer Science, University of Idaho at Idaho Falls.
- [Di Caro, 2014] Di Caro, G. (2014). An introduction to swarm intelligence issues. [http://staff.washington.edu/paymana/swarm/dicaro\\_lecture1.pdf](http://staff.washington.edu/paymana/swarm/dicaro_lecture1.pdf).
- [Everything RF, 2021] Everything RF (2021). Zigbee frequency bands. Available at: <https://www.everythingrf.com/community/zigbee-frequency-bands>.
- [Gym, 2023] Gym (2023). Gym: A Toolkit for Developing and Comparing Reinforcement Learning Algorithms. <https://www.gymnasium.dev>. Online; accessed April 18, 2023.
- [Ha, 2018] Ha, I. K. (2018). A probabilistic target search algorithm based on hierarchical collaboration for improving rapidity of drones. *Sensors (Switzerland)*, 18.
- [Hollinger and Sukhatme, 2014] Hollinger, G. A. and Sukhatme, G. S. (2014). Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287.
- [Iba, 2013] Iba, H. (2013). *Agent-Based Modeling and Simulation with Swarm*. Chapman & Hall.
- [Ioannidis et al., 2011] Ioannidis, K., Sirakoulis, G., and Andreadis, I. (2011). A path planning method based on cellular automata for cooperative robots. *Applied Artificial Intelligence*, 25:721–745.
- [June, 2021] June, L. H. (2021). Efficient swarm robotic persistent surveillance by use of stigmergy.
- [Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.
- [Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE.

- [King and Peterson, 2019] King, D. and Peterson, G. (2019). The emergence of division of labor in multi-agent systems. volume 2019-June, pages 107–116. IEEE Computer Society.
- [Larson and Sandholm, 1999] Larson, K. S. and Sandholm, T. W. (1999). Anytime coalition structure generation: an average case study. *Journal of Experimental Theoretical Artificial Intelligence*, 12(1):23–42.
- [Lu et al., 2020] Lu, L., Redondo, C., and Campoy, P. (2020). Optimal frontier-based autonomous exploration in unconstructed environment using rgb-d sensor. *Sensors*, 20(22):6507.
- [Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- [Mousavi et al., 2019] Mousavi, S., Afghah, F., Ashdown, J. D., and Turck, K. (2019). Use of a quantum genetic algorithm for coalition formation in large-scale uav networks. *Ad Hoc Networks*, 87:26–36.
- [Mugan, 2022] Mugan, J. (2022). Rllib for deep hierarchical multi-agent reinforcement learning. <https://deumbra.com/2022/08/rllib-for-deep-hierarchical-multiagent-reinforcement-learning/>.
- [Nakanishi et al., 2021] Nakanishi, D., Singh, G., and Chandna, K. (2021). Developing autonomous drone swarms with multi-agent reinforcement learning for scalable post-disaster damage assessment. CUSP-GX-5006: Urban Science Intensive II (Fall 2021).
- [Ouden et al., 2023] Ouden, O. d., Smit, B., Gerth, J., and Wezel, J. (2023). Cema delivery systems for detecting, listening, and interfering with data communication lines. Technical Report NLR-CR-2023-066, Royal NLR - Netherlands Aerospace Centre. CUSTOMER: Ministry of Defence.
- [P and Ghosh, 2022] P, M. G. S. and Ghosh, S. (2022). Gradient direction turn switching strategy for source localization. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3754–3759.
- [Remote Flyer, 2021] Remote Flyer (2021). How fast can a drone fly? with examples. Available at: <https://www.remoteflyer.com/how-fast-can-a-drone-fly-with-examples/>.
- [Rey and Neuhäuser, 2011] Rey, D. and Neuhäuser, M. (2011). Wilcoxon-signed-rank test. In Lovric, M., editor, *International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg.
- [Saculinggan and Balase, 2013] Saculinggan, M. and Balase, E. A. (2013). Empirical power comparison of goodness of fit tests for normality in the presence of outliers. *Journal of Physics: Conference Series*, 435(1).
- [Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- [SHAPIRO and WILK, 1965] SHAPIRO, S. S. and WILK, M. B. (1965). 52(3-4):591–611.
- [Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., and Guez, A. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.
- [Soemers et al., 2018] Soemers, D. J. N. J., Sironi, C. F., Schuster, T., and Winands, M. H. M. (2018). Enhancements for real-time monte-carlo tree search in general video game playing. *Department of Data Science and Knowledge Engineering, Maastricht University*.
- [Stef Janssen, 2019] Stef Janssen, Alexei Sharpanskykh, R. C. K. L. (2019). Using causal discovery to analyze emergence in agent-based models. *Simulation Modelling Practice and Theory*, 96:101940.
- [Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition.
- [Tam, 2021] Tam, A. (2021). A gentle introduction to particle swarm optimization. <https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/>. Accessed: [Insert the date you accessed the website].
- [Utilities One, 2023] Utilities One (2023). Mitigating network congestion strategies for telecommunications traffic management. <https://utilitiesone.com/mitigating-network-congestion-strategies-for-telecommunications-traffic-management>. Accessed: 2023-12-12.
- [Vlassis, 2003] Vlassis, N. (2003). A concise introduction to multiagent systems and distributed ai. Technical report, University of Amsterdam, The Netherlands.
- [Weiss, 2013] Weiss, G. (2013). *Multiagent Systems*. Intelligent robotics and autonomous agents. The MIT Press, Cambridge, Massachusetts ; London, England, 2 edition.

- [Westheider et al., 2023] Westheider, J., Rückin, J., and Popović, M. (2023). Multi-uav adaptive path planning using deep reinforcement learning. *arXiv preprint arXiv:2303.01150*, 1(cs.RO). Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy - EXC 2070 – 390732324. Authors are with the Cluster of Excellence PhenoRob, Institute of Geodesy and Geoinformation, University of Bonn. Corresponding: jwesthei@uni-bonn.de.
- [Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.
- [Wu and Ramchurn, 2020] Wu, F. and Ramchurn, S. D. (2020). Monte-carlo tree search for scalable coalition formation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, School of Computer Science and Technology, University of Science and Technology of China. sdr1@soton.ac.uk.
- [Zhou et al., 2021] Zhou, Y., Chen, A., He, X., and Bian, X. (2021). Multi-target coordinated search algorithm for swarm robotics considering practical constraints. *Frontiers in Neurorobotics*, 15.

# II

Literature Study  
previously graded under AE4020





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Problem Definition</b>	<b>3</b>
2.1	Problem Statement . . . . .	3
2.2	Base Scenario . . . . .	4
2.3	Enhancements for a More Realistic and Complex Model . . . . .	6
<b>3</b>	<b>Signal Theory in Mission Context</b>	<b>7</b>
3.1	Signal Detection and Interference . . . . .	7
3.1.1	Signal Propagation . . . . .	7
3.1.2	Modulation Techniques for Communication . . . . .	10
3.1.3	Noise and Interference . . . . .	11
3.1.4	Adaptive Strategies: Dynamic Frequency Selection and Adaptation . . . . .	12
3.2	Locating Emitters and Receivers . . . . .	13
3.2.1	Challenges in Classifying Emitters and Receivers . . . . .	13
3.2.2	Signal Source Localization Techniques . . . . .	14
3.3	Wireless Communication Lines . . . . .	15
3.3.1	Basic Principles of wireless Communication . . . . .	15
3.3.2	Wireless Communication Interference and Countermeasures . . . . .	16
3.4	Signal Theory in Relevance for the Mission . . . . .	16
<b>4</b>	<b>UAV Swarm Intelligence</b>	<b>17</b>
4.1	UAVs as a Solution to the Problem . . . . .	17
4.1.1	Role of UAVs in Signal Interference . . . . .	17
4.1.2	Advantages of UAV Swarms for Signal Interference . . . . .	17
4.2	Swarm Intelligence and UAV Applications . . . . .	18
4.2.1	Swarm Intelligence Overview . . . . .	18
4.2.2	Biological Inspiration and UAV Swarms . . . . .	18
<b>5</b>	<b>Mission Planning &amp; Task Discussion</b>	<b>21</b>
5.1	Mission Planning . . . . .	21
5.1.1	Requirements for Each Mission Phase . . . . .	23
5.2	Task Allocation . . . . .	24
5.2.1	Cooperative Task Allocation for UAV Swarms . . . . .	25
5.2.2	Centralized, Decentralized, and Distributed Task Allocation Approaches . . . . .	27
5.3	Searching . . . . .	29
5.3.1	UAV Swarm-based Search Strategies . . . . .	29
5.3.2	Adaptive and Dynamic Searching . . . . .	30
5.4	Mapping . . . . .	31
5.4.1	UAV Swarm-based Mapping Techniques . . . . .	31
5.4.2	Real-time and Collaborative Mapping . . . . .	32
5.5	Monitoring and Interfering . . . . .	33
5.5.1	UAV Swarm-based Monitoring and Interference Strategies . . . . .	33
5.5.2	Adaptive and Collaborative Digital Communication Interference . . . . .	33
<b>6</b>	<b>Task Specifications &amp; Algorithms</b>	<b>34</b>
6.1	Task Architecture . . . . .	34
6.2	Searching Phase . . . . .	34
6.2.1	Tasks . . . . .	34
6.2.2	Algorithms . . . . .	35
6.3	Mapping Phase . . . . .	36

6.3.1	Tasks	36
6.3.2	Algorithms	36
6.3.3	Frontier-based Exploration for Multi-UAV Mapping	37
6.4	Monitoring and Interfering Phases	38
6.4.1	Tasks	38
6.4.2	Algorithms	38
<b>7</b>	<b>Reinforcement Learning to Enhance Mission Efficiency &amp; Adaptability</b>	<b>39</b>
7.1	Reinforcement Learning	39
7.2	Reinforcement Learning (RL) for Interference Point Finding (critical)	40
7.2.1	State Representation	40
7.2.2	Action Space	41
7.2.3	Reward Function	41
7.3	RL for Multi-agent Mapping of Communication Line (desirable)	41
7.3.1	State Representation	41
7.3.2	Action Space	42
7.3.3	Reward Function	42
<b>8</b>	<b>Research Proposal</b>	<b>43</b>
8.1	Research Questions	44
8.2	Methodology	44
8.2.1	Strategy	44
8.2.2	Algorithmic Approaches	45
8.2.3	Model Environment	45
8.3	Model Enhancements	46
8.4	Evaluation Metrics	47
8.5	Experiments	47
8.5.1	Baseline Performance	47
8.5.2	Environmental, UAV Performance, and Strategic Algorithm Enhancements	47
8.5.3	Comprehensive Scenario	48
<b>A</b>	<b>Project Planning</b>	<b>49</b>
	<b>References</b>	<b>51</b>

# List of Figures

2.1	Base Scenario . . . . .	4
3.1	Propagation Paths [10] . . . . .	8
3.2	Omni-directional and Directional Signal Propagation [58] . . . . .	8
3.3	Comparison of a horn antenna (top) and an omni-directional dipole (down) receiver antenna. [1] . . . . .	8
3.4	The inverse square law for electromagnetic radiation in free space, showing how the intensity decreases with the square of the distance from the source as the power spreads over a larger area. [70] . . . . .	9
3.5	Amplitude -, Frequency -, Phase Modulation visualised . . . . .	10
3.6	Signal decomposed in frequency domain [33] . . . . .	11
3.7	Signal superimposed on signal plus noise [23] . . . . .	11
3.8	Example of signal interference caused by two stations emitting a radio wave, resulting in time-varying interference patterns due to the superposition of the signals. [41] . . . . .	11
3.9	Theoretical overview of single antenna detection with varying noise conditions. From top to bottom, the same signal impinges on the antenna as noise increases. The 'noise' detection threshold is at 1.75 (red dashed line), minimum pulse duration is 40 samples (green dashed line), and hysteresis minimum threshold is set at 1 (yellow dashed line). [53]	12
3.10	Schematic overview of direction finding over an antenna array in theta/phi and azimuth/elevation coordinates. [62] . . . . .	14
3.11	Schematic overview of direction finding over an antenna array in theta/phi and azimuth/elevation coordinates. [53] . . . . .	15
4.1	Examples of natural swarm systems include bird flocks, bee colonies, fish schools, and ant swarms. [66] . . . . .	19
5.1	State-Machine for agent . . . . .	21
5.2	Visualisation of Mission Flow for UAV swarm . . . . .	22
5.3	Communication Line Mapping & Interference Point . . . . .	23
5.4	Communication Line Mapping & Interference Point . . . . .	28
6.1	Particle Swarm Optimization in action: UAVs dynamically update their positions and velocities to converge towards the optimal solution(s) based on their personal best and global best positions. . . . .	36
6.2	Circling RSSI Threshold Line (RT) until Mapping Point is found . . . . .	36
6.3	Visualisation of Frontier-based Exploration for Multi-UAV Mapping . . . . .	37
7.1	Generic flow RL agent . . . . .	40
7.2	RL Environment for Interference Point Search . . . . .	40
7.3	Discretized mission environment into a grid consisting of $m \times m$ square cells. [48] . . . . .	41
8.1	Visualisation of Mission Flow for UAV swarm . . . . .	44
A.1	Gantt chart of the project planning . . . . .	50

# List of Tables

5.1	Qualitative Trade-off for Approaches in Task Allocation . . . . .	27
5.2	Qualitative Trade-off for Centralized, Decentralized, and Distributed Approaches . . . .	28
5.3	Qualitative Trade-off for Search Algorithms . . . . .	30
5.4	Qualitative Trade-off for Mapping Techniques . . . . .	32

# 1

## Introduction

Activities in cyberspace and the electromagnetic spectrum, collectively known as Cyber Electromagnetic Activities (CEMA), have become increasingly important in society. These activities address Cyber-attack/defense focusing on Radio Frequency (RF) signals and cyber dealing with the flow of information, 3G/4G/5G networks, and applications within individual networks. UAVs or drones, equipped with specialized payloads, can be utilized to effectively position themselves and interfere with wireless data communication between cyber systems. [53]

In the context of modern society, Swarm Intelligence (SI) has emerged as a promising solution to the challenges posed by complex and dynamic environments, such as those can be found in wireless communication interference, or within CEMA. Some of these challenges include the need for rapid response, coordination among multiple agents, and adaptability in the face of changing conditions. Inspired by the collective behavior of biological systems such as ants, bees, and birds, SI algorithms can be applied to control a group of Unmanned Aerial Vehicles (UAVs) or drones, coordinating their actions to achieve a common goal. These UAV swarms offer numerous advantages, such as adaptability, robustness, scalability, and enhanced intelligence. The adaptability of SI enables the swarm to efficiently react to changes in the environment, such as frequency changes or moving transmitters, allowing them to maintain their effectiveness in the face of evolving challenges. The robustness of SI ensures that the swarm can continue functioning despite difficulties like frequency noise patterns or UAV fall-outs, providing a reliable solution in complex situations. The scalability of SI allows the swarm to easily accommodate multiple transmitters and communication lines or adapt to a larger search area, making them particularly suited for tackling the challenges of CEMA. The enhanced intelligence of UAV swarms comes into play by leveraging information from different positions at the same timestamp, enabling the development of smarter concepts such as signal direction of arrival determination or geo-location of a signal. This collective intelligence empowers the swarm to make more informed decisions and execute more effective strategies, further strengthening their capabilities in complex environments.

Existing applications of UAV SI in related fields, such as search and rescue or environmental monitoring [7, 11], have demonstrated the potential effectiveness of this approach in addressing complex, real-world problems. This highlights the potential for SI to make a significant impact on communication line interception and other challenges in the CEMA domain.

The approach for addressing the problem of wireless communication interference involves the development of a high-level strategy that leverages SI techniques and proposes innovative algorithms to enhance mission efficiency and adaptability. The UAV swarm will be tasked with searching, mapping, monitoring, and interfering with the communication lines of enemy systems while maximizing the efficiency and adaptability of the swarm.

To effectively achieve these tasks, we employ a combination of algorithms, including Particle Swarm Optimization (PSO) [32] for the searching phase, Grid Search [47], and Reinforcement Learning (RL) for the mapping phase [48], as well as RL for the monitoring and interfering phases [47]. By leveraging RL, the UAVs can improve their ability to locate the optimal interference point and map communication

lines more effectively, resulting in more efficient and adaptable mission performance. This approach becomes particularly valuable in complex environments where it is challenging to develop an adequate formula for the agent to work with effectively. The use of RL allows the swarm to learn from their experiences, adapt to dynamic situations, and make more informed decisions, ensuring efficient coordination of the UAVs, enabling them to systematically explore and evaluate the environment, while maintaining adaptability and robustness throughout the mission.

This report is organized in a structured manner that connects each chapter, providing a comprehensive understanding of the research. The introduction of the problem definition in chapter 2 sets the foundation for understanding the challenges in the mission context. Building on this, chapter 3 provides essential background on Signal Theory, discussing topics like signal detection, interference, emitter and receiver localization, and wireless communication lines, highlighting the opportunities for addressing these challenges. In chapter 4, the potential of UAV Swarm Intelligence is explored, showcasing its benefits for signal interference. The mission planning, task requirements, and strategies for searching, mapping, and monitoring are laid out in chapter 5, which connects the various components of the approach. Subsequently, chapter 6 demonstrates how different algorithms contribute to each mission phase, while chapter 7 emphasizes the role of Reinforcement Learning in enhancing efficiency and adaptability in the mapping/monitoring and interference phase. Finally, chapter 8 synthesizes the insights gathered throughout the report and proposes a roadmap for future work in this domain, addressing research questions, evaluation metrics, model enhancements, and experiments. In Appendix A, a Gantt chart of the project planning can be found.

# 2

## Problem Definition

This chapter provides an overview of the problem by presenting the problem statement in section 2.1, base scenario in section 2.2, and enhancements for a more realistic and complex model in section 2.3. This sets the stage for developing and evaluating a novel SI state-machine algorithm for communication line interception using a swarm of UAVs.

### 2.1. Problem Statement

In the context of CEMA, complex environments pose significant challenges for detection, localization, mapping, and interference with communication lines. The initial approach involves using a single UAV for these tasks. [53] The flexibility of a UAV in positioning itself within the CEMA scene allows it to adapt to various environments and operational requirements. This mobility and adaptability make UAVs valuable assets in signal interference tasks, as they can quickly change their location to optimize signal reception or interference performance.

However, this method has proven to be ineffective, particularly in complex environments, due to several reasons, including limited coverage, lack of redundancy, reduced adaptability to dynamic changes, and inefficient resource allocation. These limitations underscore the need for a more robust and adaptable solution. SI can effectively address the challenges associated with communication line interception in complex CEMA environments by offering numerous advantages, such as adaptability, robustness, scalability, and enhanced intelligence.

Although SI has been successfully applied in various fields, its implementation in communication line interception remains limited. A comprehensive approach to communication line interception using SI should involve a workflow consisting of searching for signals, mapping the signal sources, monitoring the communication lines, and interfering with them as necessary. The inherent adaptability, robustness, and scalability of SI enable UAV swarms to react efficiently to environmental changes, maintain functionality in the face of difficulties, and accommodate multiple transmitters and communication lines or adapt to larger search areas. Additionally, the enhanced intelligence of the swarm allows for leveraging information from different positions simultaneously, leading to smarter strategies such as signal direction of arrival determination or geo-location of a signal.

Despite the potential benefits of SI for communication line interception, the existing literature falls short in addressing this comprehensive approach. Therefore, there is a need for further research and development to explore the full potential of SI in the context of CEMA and communication line interception.

Although the paper by Jang et al. [17] touches upon several relevant aspects, it primarily focuses on the control part of radar jamming and lacks a comprehensive approach to signal environments, including the aforementioned workflow. This suggests a research gap in the strategic application of agent-based SI for communication line interception.

The primary objective of this thesis is to address these challenges by developing and evaluating a novel, bio-inspired SI state-machine algorithm for a swarm of UAVs in the CEMA domain. A state-machine

mechanism is utilized because it ensures convergence to the solution for the swarm, rather than relying on a random search for the communication lines. It also allows agents to act in multiple phases of the mission. The swarm-based approach aims to effectively detect, locate, map, and interfere with communication lines in complex RF environments, leveraging the collective intelligence and distributed capabilities of multiple UAVs. This research will contribute to filling the gap in the strategic application of agent-based SI for communication line interception and provide a more robust and adaptable solution for the CEMA domain.

## 2.2. Base Scenario

This section introduces a base scenario to illustrate the CEMA play field. Within the area of interest the major goals are to detect and interfere with the existing communication lines, which is attempt to achieve with a UAV swarm. The remainder of this section elaborates on the goals, environment specifications, and states of an agent.

Figure 2.1 depicts the base scenario featuring an omni-directional transmitter/receiver ( $TR_x$ ) and a directional transmitter/receiver ( $TR_x$ ). Both are included as the omni-directional signal is often easier to find, but the directional signal is easier to map and found more often within tactical communication. A swarm of 10 UAVs is deployed with their starting positions uniformly distributed across a 2D search area. The orange circle represents an Received Signal Strength Indication (RSSI) Threshold (RT), indicating a theoretical boundary beyond which an agent should not pass to remain visually unseen, or stealthy.

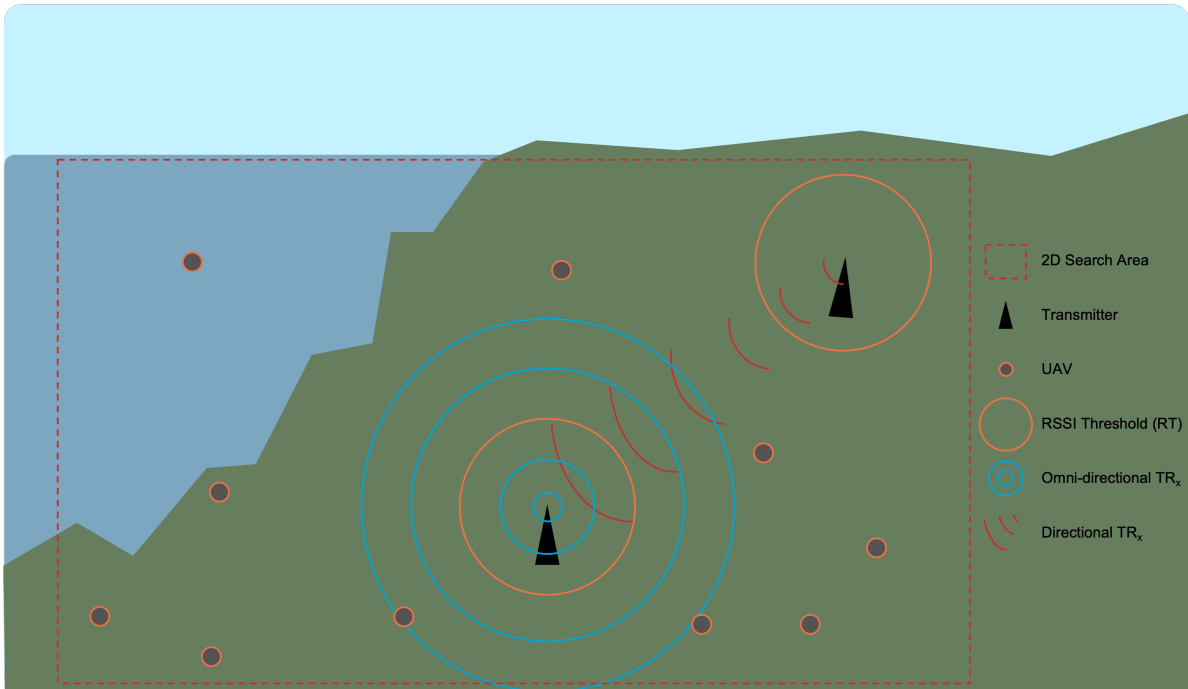


Figure 2.1: Base Scenario

The main goal of the swarm is to effectively detect and interfere with communication lines. However, to achieve this goal, it can be broken down into several key tasks that enable the swarm to accomplish its objective. These tasks include:

1. Signal detection, where the swarm identifies the presence of a transmitter in the search area.
2. Finding the communication line, which involves locating the path of the transmission from the transmitter to its intended receiver.
3. Mapping the communication line, where the swarm creates a signal map of the communication line for monitoring and interference purposes.



4. Monitoring the communication line, which involves continuously monitoring the transmission to detect any changes or interference.
5. Finding the interference point, where the swarm locates the source of interference and deploys an interference agent to disrupt the transmission.

The environment specifications for the proposed scenario are drafted. The environment is divided into three main components: the field, transmitters, and UAVs. The specifications for each component are described below:

**Environment:**

**Field:**

- The scenario will initially be in 2D.
- Known search area of x by y km.
- Simple noise model.

**Transmitters:**

- $TRO_x$ :  $N_{TRO}$  omni-directional transmitters/receivers, where  $N_{TRO} = 1$ . Position:  $(x_{TRO}, y_{TRO})$ . Static.
- $TRD_x$ :  $N_{TRD}$  directional transmitters/receivers, where  $N_{TRD} = 1$ . Position:  $(x_{TRD}, y_{TRD})$ . Static.
- $F_{TR_x}$ : Transmitter frequency of 2.4 GHz and 2.42 GHz. (Zigbee communication [26])
- All transmitters act in the same bandwidth.
- $\theta_{TRD_x}$ : Range of direction of transmission:  $205^\circ - 250^\circ$ .
- One signal at a time: signals are not simultaneous to distinguish signals.
- Pulse frequency and time of transmitters: continuous data stream, but enough pause to distinguish signals.

**UAVs:**

- A swarm of  $N_{UAV}$  UAVs or agents, with  $N_{UAV} = 10$ .
- Bandwidth and RTs are known by UAVs.
- The locations for releasing UAVs are uniformly distributed over the search area.
- Maximum total velocity of 20 m/s for UAVs. [56]
- Maximum flying time for UAVs: unrestricted.

In a coordinated system of UAVs, effective communication and collaboration play vital roles in achieving the desired objectives.

**Communication & Collaboration:**

- UAVs can communicate with each other, sharing information such as received signals, position, and current task.
- With environment inputs, and data sharing, agents can switch tasks or state, and can collaborate where needed.

In the context of UAVs operating in a coordinated manner, states, inputs, and actions can be defined for each agent to ensure efficient performance of tasks. Here is a brief introduction to the states, inputs, and actions for an agent:

- **States:** The UAVs can have different states, such as searching, mapping, monitoring & interfering. Each state corresponds to specific tasks that the UAV is performing.
- **Inputs:**
  - Position and velocity of the UAVs.
  - RSSI and frequency.
  - Signal-to-Noise Ratio (SNR) thresholds for frequency detection (10-20 dB). [44]

- RT of transmitters.
- phase dependent environment information from other agents.
- **Actions:**
  - Adjust velocity and direction to navigate the search area.
  - Share information with other UAVs about the received signal strength, position, current task, and other relevant information.
  - Collaborate with other UAVs to map the communication line and locate the signal source.
    - \* Collaborate for Signal source search.
    - \* Collaborate for mapping. (desirable)
    - \* Collaborate for interference point search.
  - Interfere with the signal by disrupting the communication line.

## 2.3. Enhancements for a More Realistic and Complex Model

Various enhancements can be introduced to further challenge the performance of the system and assess its adaptability to more complex scenarios. These scenarios will be divided into the environment, UAV performance, and strategic algorithm enhancements. These enhancements will increase the difficulty and realism of the problem, allowing for a more comprehensive evaluation of the algorithm's capabilities. A distinction has been made between critical and desirable enhancements of the model.

### Environment Enhancements:

- Increased number of transmitters. (critical)
- Dynamic environment with moving transmitters. (critical)
- Vary number of UAVs. (critical)
- Complex noise patterns and interference from various sources. (desirable)
- Frequency-hopping transmitters to simulate agile communication systems. (desirable)

### UAV Performance Enhancements:

- Single or multiple points of release for the UAVs, impacting their initial positioning and coordination. (critical)
- Constraints on maximum flying time for the UAVs, requiring efficient task execution and energy management. (desirable)
- Incorporating UAV fall-out in the model and assess task reallocation. (desirable)
- Actively defining and expanding the search area during the mission, adapting to the evolving RF landscape. (desirable)
- Unknown bandwidth of interest, requiring the UAVs to adapt their monitoring capabilities. The UAVs' bandwidth is set at 10 MHz, with adaptability to iterate over 40 bands (covering a total bandwidth of 400 MHz). In a spectrum of interest ranging from 2 to 4 GHz, UAVs listen to channels 1, 2, 3, 4, and 5. (desirable)

### Strategic Algorithm Enhancements:

- Collaborative techniques like beamforming and phase difference can enhance Direction of Arrival (DoA) estimation. (desirable)
- Triangulation can be used to pinpoint the position of a signal source. (desirable)
- Free Space Path Loss (FSPL) formula can improve distance estimation with the collaboration of at least two UAVs. (desirable)

# 3

## Signal Theory in Mission Context

In this chapter, various aspects of signal theory in the context of the mission are discussed. This chapter is divided into three main sections. In section 3.1, signal detection and interference are discussed, including topics such as omni-directional and directional signal propagation, modulation techniques for communication, noise and interference. In addition adaptive strategies like dynamic frequency selection and adaptation will be discussed. Following this, section 3.2 addresses the challenges in locating emitters and receivers, as well as the techniques used for signal source localization. Then, in section 3.3, wireless communication lines are explored, discussing the basic principles of wireless communication and the various methods used for interference and countermeasures. Finally, section 3.4 concludes the chapter reflecting on how the theory discussed comes in play for the development of the model.

### 3.1. Signal Detection and Interference

Within modern wireless communication systems, signal detection and active interference management play an essential role in ensuring reliable and secure information exchange, particularly in scenarios where one needs to disrupt unauthorized or unintended transmissions. This section explores the fundamental concepts and techniques related to detecting and identifying signals, as well as the challenges and methods associated with actively causing signal interference. We will discuss different signal propagation characteristics, modulation techniques, sources of noise and interference, and the importance of dynamic frequency selection and adaptation in maintaining the performance of communication systems.

#### 3.1.1. Signal Propagation

The fundamental characteristics of electromagnetic waves emitted by a transmitter can be described as follows:

1. The emitted waves are transverse electromagnetic waves, with electric and magnetic fields oscillating perpendicular to each other and to the direction of propagation.
2. The emitted waves require no supporting medium.
3. The emitted waves can undergo reflection, refraction, and diffraction, and are subject to interference and Doppler effects.
  - I. Reflection occurs when the wavefront is diverted back from the interface of two media. The reflection may be specular (that is, direct) or diffuse, according to the nature of the contact surface.
  - II. Refraction occurs when a radio wave moves from a medium of one density to a medium of a different density, causing the velocity of the wave to change. If the wave moves obliquely from one medium to another, the part that enters first will travel either faster or slower than the part that enters last, resulting in the wavefront being bent or refracted.

- III. Diffraction occurs when waves bend around an object, causing a diversion of part of the energy. This can be received at some distance below the top of an object, such as a mountain.
- 4. Radio waves can pass through opaque objects such as walls, buildings, and trees, and are attenuated to varying degrees. Attenuation is greater at higher frequencies, and rain does not affect radio waves unlike light waves. [10]

Figure 3.1 depicts the three primary paths that radio waves can travel between a transmitter and receiver:

1. The surface wave, which follows the earth’s contour.
2. The sky wave, which returns after reflection from the ionosphere.
3. The space wave. A combination of space and surface waves is termed the ground wave.

The space wave assumption is applicable when the distance between the transmitter and receiver is sufficiently small, allowing the radar horizon to be neglected. In the context of this study, communication signals are treated as space waves because the transmitter and receiver are relatively close to each other.

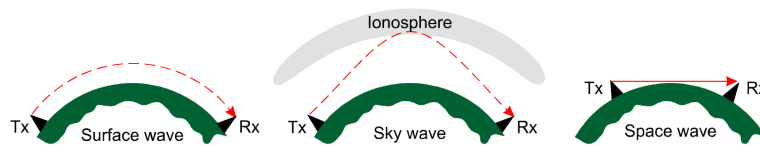


Figure 3.1: Propagation Paths [10]

Signal propagation affects the use and effectiveness of a wireless communication line. Two major wave propagations are assumed. Either the wave propagation is omni-directional or directional.

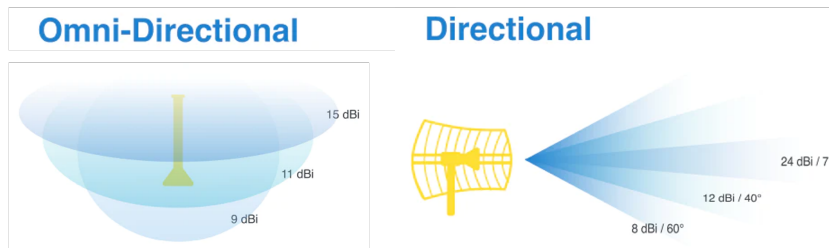


Figure 3.2: Omni-directional and Directional Signal Propagation [58]

Omni-directional signal propagation ensures that radio waves are transmitted uniformly in all directions, covering a wide area and providing broad coverage. Omni-directional antennas are well-suited for general-purpose communication applications where the location of receiving devices may be unknown or the devices may be moving relative to the transmitter.

Directional signal propagation involves transmitting radio waves in a focused beam, targeting a specific direction and covering a smaller area. Directional signals offer several advantages, including higher signal strength in the targeted direction, reduced interference with other signals, and lower susceptibility to eavesdropping or interfering.

In Figure 3.3, two distinct types of antennas are displayed: a directional horn antenna and an omni-directional dipole receiver antenna. The horn antenna features a narrower beamwidth but possesses a longer range, allowing it to detect signals at greater distances. Conversely, the omni-directional dipole receiver antenna can receive signals from all directions

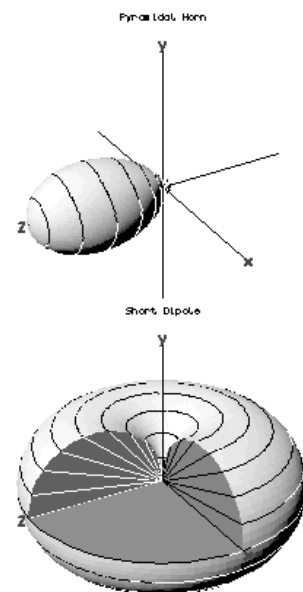
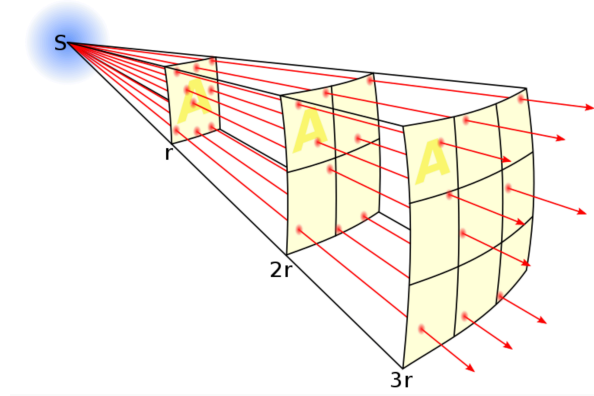


Figure 3.3: Comparison of a horn antenna (top) and an omni-directional dipole (down) receiver antenna. [1]

but has a shorter range compared to the horn antenna. This trade-off between the range and directionality makes each antenna suitable for different applications, depending on the specific requirements of the task at hand. For instance, a horn antenna might be more suitable for long-range directional communication, while an omni-directional dipole receiver antenna could be ideal for monitoring or broadcasting signals in environments where the direction of the signal source is unknown or constantly changing.

Figure 3.4 illustrates the inverse square law for electromagnetic radiation in free space, demonstrating how the intensity of the radiation decreases with distance. According to the inverse square law, the power of the electromagnetic radiation spreads over an area proportional to the square of the distance from the source. As a result, the intensity decreases at a rate inversely proportional to the square of the distance. This principle is important in understanding signal propagation and the diminishing signal strength as the distance between the transmitter and receiver increases.



**Figure 3.4:** The inverse square law for electromagnetic radiation in free space, showing how the intensity decreases with the square of the distance from the source as the power spreads over a larger area. [70]

For both omni-directional and directional radio signal propagation the frequency remains unchanged, although the signal strength diminishes with increasing distance between the transmitter and receiver. The FSPL formula can be used to estimate the signal loss due to distance in an ideal, unobstructed environment. The RSSI from a certain radio signal source depends on the transmission loss (TL) that a signal, which is originally at a given source level (SL), experiences propagating from source to receiver. In the context of free space and ideal, unobstructed conditions, the transmission loss (TL) is equal to the FSPL. This can be summarized by the passive radio signal equation in Equation 3.1.

$$RSSI = SL - TL = SL - FSPL \quad (3.1)$$

The FSPL can be calculated using the formula in Equation 3.2 [71]:

$$FSPL(dB) = 20 \times \log_{10}(d) + 20 \times \log_{10}(f) + 20 \times \log_{10}\left(\frac{4\pi}{c}\right) \quad (3.2)$$

Where:

- FSPL (dB) is the free space path loss in decibels
- $d$  is the distance between the transmitter and receiver in meters
- $f$  is the frequency of the radio signal in Hz
- $c$  is the speed of light in meters per second (approximately  $3 \times 10^8 m/s$ )

As the signal propagates through free space, its amplitude also decreases with increasing distance between the transmitter and receiver. The change in amplitude can be related to the FSPL by considering that the power intensity is proportional to the square of the amplitude. In real-world scenarios, the total

transmission loss may be higher than the FSPL due to additional factors like reflections, diffractions, or absorptions by objects in the environment.

By understanding the characteristics of omni-directional and directional signal propagation, as well as the FSPL formula for estimating signal loss, appropriate detection and interference strategies can be developed to address the challenges associated with each type of signal propagation.

### 3.1.2. Modulation Techniques for Communication

Modulation techniques are essential for encoding information onto a carrier signal by varying its characteristics, such as amplitude, frequency, or phase. Different modulation schemes are used based on the requirements of the communication system. Some common modulation techniques include:

**Amplitude Modulation (AM):** In AM, the amplitude of the carrier signal is varied in proportion to the message signal. The frequency and phase of the carrier signal remain constant. A simple formula for an AM signal is:

$$x(t) = (A + m(t)) \cos(2\pi f_c t) \quad (3.3)$$

where  $x(t)$  is the modulated signal,  $A$  is the amplitude of the carrier signal,  $m(t)$  is the message signal, and  $f_c$  is the carrier frequency.

**Frequency Modulation (FM):** In FM, the frequency of the carrier signal is varied according to the message signal, while the amplitude and phase remain constant. The formula for an FM signal is:

$$x(t) = A \cos \left( 2\pi f_c t + k_f \int_0^t m(\tau) d\tau \right) \quad (3.4)$$

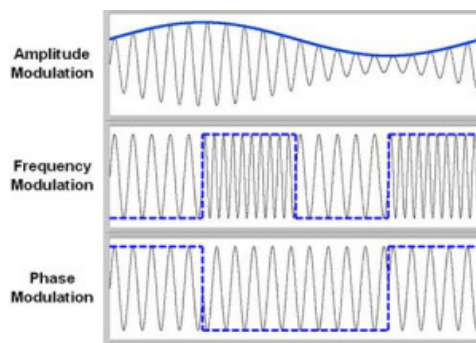
where  $k_f$  is the frequency sensitivity and  $\tau$  is a dummy variable for integration.

**Phase Modulation (PM):** In PM, the phase of the carrier signal is varied according to the message signal, while the amplitude and frequency remain constant. The formula for a PM signal is:

$$x(t) = A \cos (2\pi f_c t + k_p m(t)) \quad (3.5)$$

where  $k_p$  is the phase sensitivity. [37]

The modulation techniques mentioned above are visualized in Figure 3.5.



**Figure 3.5:** Amplitude -, Frequency -, Phase Modulation visualised

Modulation techniques, such as Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), are also widely used in wireless communication systems. These techniques involve modulating the carrier signal based on discrete values representing information [37].

When multiple signals are transmitted simultaneously in the same bandwidth, they can be distinguished and isolated using a signal processor in the frequency domain. A signal processor can analyze the spectral components of the received signal, which represent the frequency content of the signal. By using techniques such as Fourier analysis or signal processing, the processor can separate the different frequency components by decomposing the summed signal. [52] The process is visualised in Figure 3.6. This allows the UAV swarm to identify and locate multiple transmitters operating in the same frequency band, even if they are transmitting simultaneously. The ability to distinguish and isolate multiple signals is essential for effective communication monitoring and interference operations.

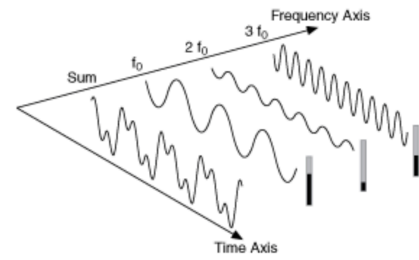


Figure 3.6: Signal decomposed in frequency domain [33]

### 3.1.3. Noise and Interference

Ideally only signals of interest are resolved within the processing chain. However, within the CEMA environment often noise or other interfering sources are present. Noise and interference are usually unwanted signals that can degrade the performance of communication systems. There are internal and external sources. Internal noise is generated from the hardware of a communication system, while external noise sources occur due to the environment.

Noise can be classified into various types, including thermal noise, shot noise, and quantization noise. Thermal noise, also known as Johnson-Nyquist noise, arises from the random motion of electrons in conductors. Shot noise is generated from the discrete nature of current flow in a conductor, while quantization noise arises from the limited resolution of analog-to-digital converters in digital communication systems. [54]

External noise sources can cause interference with the desired signal, which can be classified into co-channel interference and adjacent channel interference. Co-channel interference occurs when two or more signals share the same frequency band, while adjacent channel interference occurs when a signal spills into an adjacent frequency band. Interference can also be generated intentionally to disrupt or jam a signal. To visualize the effects of superimposing signals, consider two simple signals that interfere with each other in Figure 3.8. Depending on their phase and amplitude, the overlaid signals can either cancel out or alter the original signal. A signal superimposed on the signal plus noise is shown in Figure 3.7.

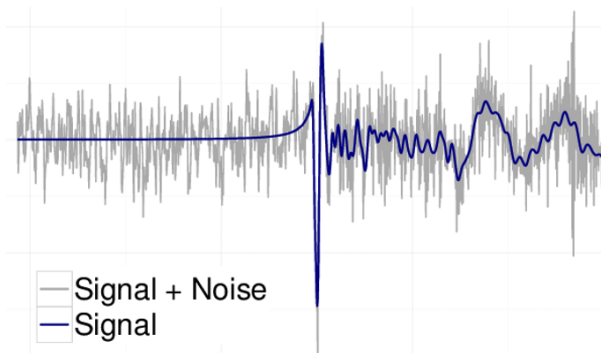


Figure 3.7: Signal superimposed on signal plus noise [23]

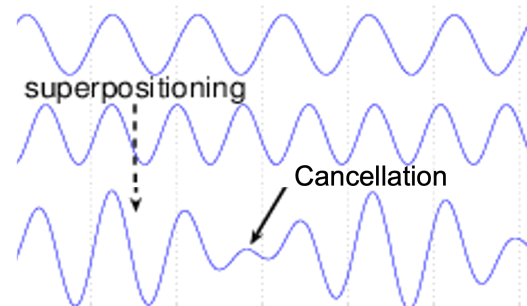
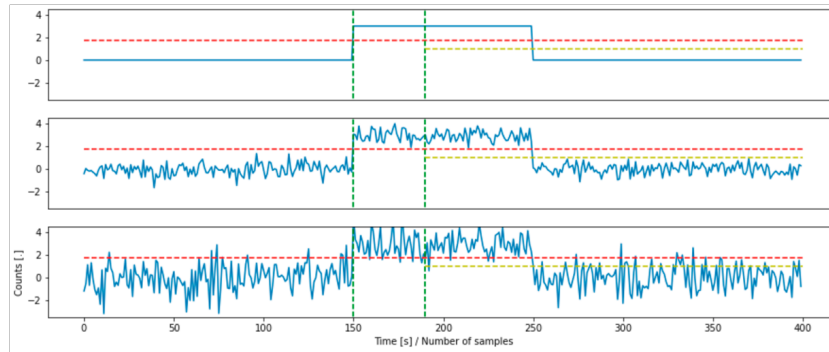


Figure 3.8: Example of signal interference caused by two stations emitting a radio wave, resulting in time-varying interference patterns due to the superposition of the signals. [41]

For a receiver antenna, having a broader bandwidth increases the likelihood of detecting a signal, as it can capture a wider range of frequencies. However, this also increases the noise level, as a broader bandwidth may capture more unwanted signals and interference. Figure 3.9 illustrates a theoretical overview of single antenna detection under varying noise conditions. [53] From top to bottom, the same signal impinges on the antenna, with noise conditions increasing in severity. The 'noise' detection threshold is positioned at the red dashed line, the minimum pulse duration is determined at a predefined



number of samples (green line). The hysteresis minimum is a threshold set at the yellow dashed line. This example highlights the trade-off between increasing the chance of signal detection and managing the noise level in the system.



**Figure 3.9:** Theoretical overview of single antenna detection with varying noise conditions. From top to bottom, the same signal impinges on the antenna as noise increases. The 'noise' detection threshold is at 1.75 (red dashed line), minimum pulse duration is 40 samples (green dashed line), and hysteresis minimum threshold is set at 1 (yellow dashed line). [53]

Understanding the sources and types of noise and interference in a communication system's environment is essential for developing effective strategies to address their impact on signal detection and interference. To measure the quality of a communication system, the SNR is used, which is the ratio of the power of the desired signal to the power of the background noise.

$$SNR = \frac{P_s}{P_n} \quad (3.6)$$

where  $P_s$  is the power of the desired signal, and  $P_n$  is the power of the background noise. A higher SNR indicates better signal quality and improved system performance.

In summary, understanding noise and interference and their impact on communication systems is essential for developing effective signal detection and interference strategies. Additionally, being aware that interference can be intentionally generated highlights the importance of implementing security measures to protect against malicious attacks.

### 3.1.4. Adaptive Strategies: Dynamic Frequency Selection and Adaptation

In an increasingly connected world, the number of devices utilizing radio frequency (RF) communication has grown exponentially. This surge in usage has led to a congested radio spectrum, making it essential for devices to adapt their communication strategies to avoid interference and maximize efficiency. This subsection discusses the role of receiver antennas in detecting signals, along with adaptive strategies and bandwidth selection methods for detecting wireless communication lines.

A receiver antenna is an essential component in RF communication systems, designed to intercept and convert electromagnetic waves into electrical signals. It captures the incoming radio waves, which contain the transmitted information, and passes the received signal to the receiver for further processing. The receiver antenna's ability to detect signals depends on its design, polarization, and radiation pattern, which can be adapted to improve signal reception in different environments.

To ensure efficient communication in a crowded radio spectrum, adaptive strategies such as Dynamic Frequency Selection (DFS) and adaptation can be employed. DFS is a technique that enables devices to monitor the RF environment and switch to less congested frequency bands to ensure optimal communication. [50] By continuously assessing the available frequency bands, devices can identify the ones with the lowest levels of interference and adjust their communication accordingly. This adaptability is particularly beneficial in environments with high mobility and varying RF conditions.

The proper selection of bandwidth is essential for detecting wireless communication lines, as it can significantly impact the system's performance. Several factors must be considered when selecting the



appropriate bandwidth, such as the SNR, the expected data rate, and the available frequency bands. Some strategies for bandwidth selection include:

**Fixed Bandwidth:** This approach involves selecting a fixed bandwidth based on the anticipated data rate and channel conditions. While simple to implement, it may not be the most efficient method in dynamic RF environments.

**Adaptive Bandwidth:** Adaptive bandwidth selection involves adjusting the bandwidth based on the current channel conditions. By monitoring the RF environment, devices can increase or decrease their bandwidth as needed to optimize communication performance. Let's denote the frequency range as  $F_R$ , the bandwidth as  $B$ , the processing time for reliable signal intersection as  $t_p$ , and the speed of the UAV as  $v$ .

We can calculate the number of non-overlapping bands,  $N_b$ , as:

$$N_b = \frac{F_R}{B} \quad (3.7)$$

Next, we can determine the total time,  $t_t$ , needed to scan the entire spectrum:

$$t_t = N_b \times t_p \quad (3.8)$$

Finally, we can calculate the distance,  $d$ , covered by the UAV during this time:

$$d = t_t \times v \quad (3.9)$$

Using these equations, we can analyze the efficiency of the bandwidth allocation based on the UAV's speed and the processing time required for reliable signal intersection. Using illustrative numbers, the following values can be obtained:

$$F_R = 2 \times 10^9 \text{ Hz} \quad B = 10 \times 10^6 \text{ Hz} \quad t_p = 0.1 \text{ s} \quad v = 30 \text{ m/s}$$

**Cognitive Bandwidth Selection:** This strategy employs machine learning algorithms and artificial intelligence to analyze the RF environment and determine the optimal bandwidth for communication. Cognitive bandwidth selection enables devices to adapt to varying channel conditions, enhancing the overall efficiency of the radio spectrum. [68]

Incorporating adaptive strategies such as DFS and intelligent bandwidth selection, devices can effectively detect wireless communication lines and maintain reliable connections in congested RF environments. These techniques help devices optimize their communication performance, reduce interference, and adapt to changing conditions in the radio spectrum.

## 3.2. Locating Emitters and Receivers

This section discusses the challenges associated with classifying emitters and receivers in communication systems and introduces signal source localization techniques.

### 3.2.1. Challenges in Classifying Emitters and Receivers

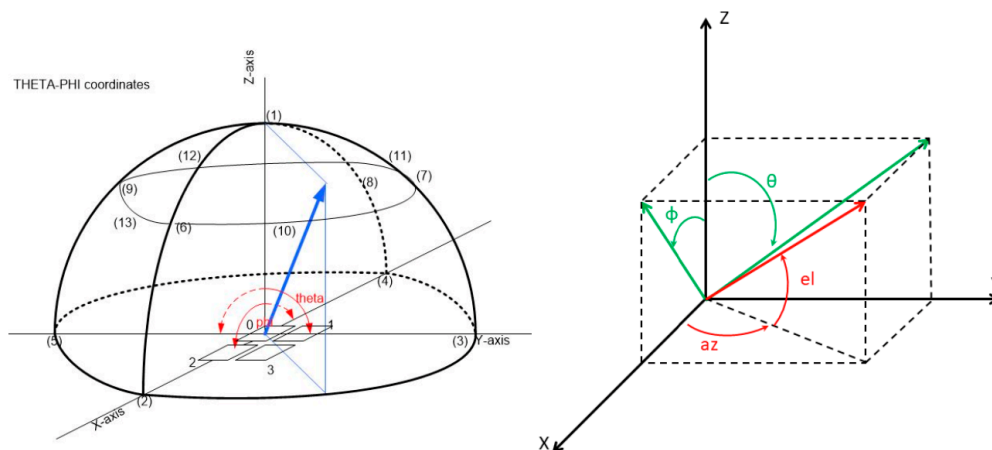
Several challenges arise when attempting to classify emitters and receivers in communication systems:

- **Encryption:** The use of encryption techniques helps protect the content of communications from eavesdropping and unauthorized access, making it challenging to identify the emitters and receivers.
- **Frequency hopping:** To avoid interfering attempts, transmitters may continuously change their transmission frequency, complicating the task of locating them. [31]

- **Moving Sources:** Utilizing the Doppler effect to identify and track moving transmitters or receivers can be challenging, especially in dynamic environments.
- **Unseen monitoring:** It is essential to maintain a safe distance from communication sources to avoid detection while monitoring a communication line, which can be difficult.
- **Positioning:** Choosing the optimal location for transmitters and receivers, either omni-directional or directional, to minimize interference is another challenge faced in this process.

### 3.2.2. Signal Source Localization Techniques

Antenna arrays play an essential role in enhancing direction finding for data links. They improve signal detection in noisy environments by boosting the SNR through the combination of signals from individual array elements. Moreover, these arrays help estimate the direction of incoming signals, acting as spatial filters by focusing on the area of interest. An aschematic overview of direction finding over an antenna array is given in Figure 3.10.



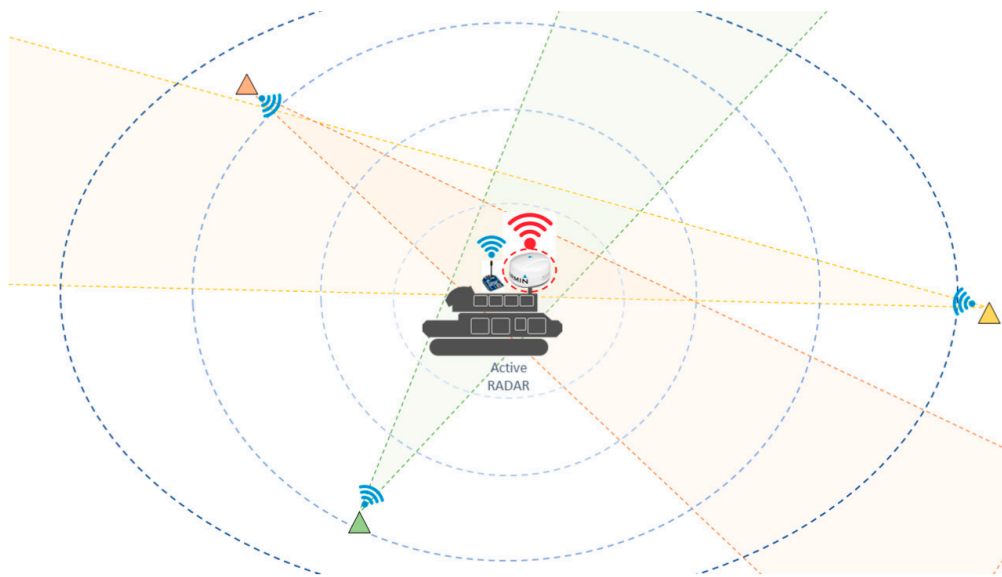
**Figure 3.10:** Schematic overview of direction finding over an antenna array in theta/phi and azimuth/elevation coordinates. [62]

Various techniques exist for determining the direction of arrival, which can be based on amplitude differences across the array or phase differences between recordings of array elements. Amplitude-based methods are often used in near-real-time applications, while phase-based methods require more computation but provide higher resolution, such as increasing from 10 degrees to 2 degrees RMS.

Phase-based direction finding techniques can be divided into two main categories: interferometry, which relies on two or more antenna elements, and beamforming. Both methods enable the detection of signals from spatially distributed sources. The ability to detect and classify sources is influenced by the resolution, which depends on the technique used and the array layout. A low-resolution system could result in a dominant source obscuring a large spatial area. [53]

In the context of exact geolocation, triangulation can be employed as a method to pinpoint the position of a signal source. Triangulation leverages multiple array measurements to determine the source location. While theoretically, two independent measurements suffice for localization, incorporating additional measurements enhances accuracy by mitigating the impact of noise and measurement inter-positioning. Moreover, when two agents know their respective locations and differences in RSSI, an assessment of the distance to the source can be theoretically made using the FSPL formula given in Equation 3.2.

With three antenna arrays illustrated in Figure 3.11, each array employs a DoA algorithm to detect the incoming signal. By analyzing the three individual solutions, a 'cross-bearing location' can be determined. In an ideal situation, DoA algorithms would offer perfect resolution, resulting in minimal uncertainty when determining the location. However, in real-world scenarios, the cross-bearing location is often identified as a region rather than a specific point due to resolution imperfections.



**Figure 3.11:** Schematic overview of direction finding over an antenna array in theta/phi and azimuth/elevation coordinates. [53]

It is essential to acknowledge that the cross-bearing resolution diminishes when antenna arrays are situated close together or aligned. This highlights the importance of optimizing array positioning to achieve the most accurate geolocation results. [53]

### 3.3. Wireless Communication Lines

This section explores wireless communication lines, elaborating on their basic principles, and the frequency bands of interest for wireless communication systems.

#### 3.3.1. Basic Principles of wireless Communication

Wireless communication lines refer to communication systems that transmit and receive wireless signals, allowing the exchange of information between devices or systems. These signals, comprised of discrete values, can represent text, images, audio, and video data. Back and forth communication between devices, such as mobile phones, satellites, and computers, indicates the presence of a communication line.

Various frequency bands are used for different communication applications, some of which are mentioned below [25]:

- L-band (1-2 GHz): This band is commonly utilized for long-range communication applications, such as satellite communication systems. The bandwidth can range from a few megahertz to several tens of megahertz.
- S-band (2-4 GHz): Typically employed for short-range and tactical communication applications, the S-band's bandwidth varies from a few megahertz to several tens of megahertz.
- C-band (4-8 GHz): The C-band is often used for high-frequency and high-bandwidth communication applications like radar systems. Its bandwidth ranges from several tens of megahertz to several hundred megahertz.
- X-band (8-12 GHz): This band is frequently used for high-frequency and high-resolution imaging applications, such as radar systems. The X-band's bandwidth can range from several hundred megahertz to several gigahertz.

Wireless communication lines play an essential role in contemporary wireless communication standards, such as 3G, 4G, and 5G networks. These networks utilize specific frequency bands to transmit data over long distances, ensuring smooth and uninterrupted communication between devices and systems globally.

For instance, the frequency bands for each network are as follows [30]:

- 3G : Operates in the range of 1.9 - 2.2 GHz.
- 4G : Employs various frequency bands ranging from 600 MHz to 3.8 GHz.
- 5G : Uses a broader range of frequency bands, spanning from sub-6 GHz to millimeter-wave bands, which can reach up to 52.6 GHz.

Additionally, tactical communication lines employed in military and emergency services operations also rely on various networks like TETRA, SINCGARS, HF radios, and SATCOM. [59] These networks offer secure, reliable, and resilient communication in challenging environments, enhancing the capabilities of wireless communication systems.

These specific frequency bands enable modern wireless communication standards to accommodate the growing demand for high-speed data transmission and support the increasing number of connected devices worldwide.

### 3.3.2. Wireless Communication Interference and Countermeasures

Interference strategies in wireless communication involve various methods to disrupt, intercept, or manipulate transmitted information. These strategies can significantly impact the performance and reliability of communication systems. Some common interference strategies include:

- Eavesdropping: Monitoring a communication line to intercept and gather sensitive information without the knowledge or consent of the communicating parties.
- Noise generation: Introducing noise into the communication channel, which can degrade the signal quality and impede effective communication.
- Signal cancellation: Employing techniques to nullify or counteract the original signal, rendering it unintelligible or ineffective.
- Influence: Manipulating the communication process by injecting false or misleading information, potentially causing confusion or the dissemination of incorrect data.

To counteract these interference strategies, various countermeasures can be employed to maintain the integrity, confidentiality, and availability of wireless communication systems.

## 3.4. Signal Theory in Relevance for the Mission

This conclusion highlights the significance of each section in this chapter for the mission and the setup of the RF environment. The signal propagation section (subsection 3.1.1) is vital for modeling the RF environment, which helps simulate the mission scenario's complexities. The Modulation Techniques for Communication section (subsection 3.1.2) aims to comprehend incoming signals and decompose them into multiple signals with varying frequencies and signal strengths, offering valuable insights for the swarm. The Noise and Interference section (subsection 3.1.3) is dedicated to understanding how noise and interference impact emitted signals, crucial for receivers to perform accurate signal processing. The Adaptive Strategies: Dynamic Frequency Selection and Adaptation section (subsection 3.1.4) develops techniques for interpreting signals and tuning in to appropriate frequency bands for agents, ensuring efficient and effective swarm communication. The Locating Emitters and Receivers section (section 3.2) emphasizes the challenges in classifying emitters and highlights the enhanced intelligence of UAV swarms, which can leverage simultaneous information from different positions. This approach enables smarter strategies development, such as signal direction of arrival determination or geo-location of a signal. Lastly, the Wireless Communication Lines section (section 3.3) defines the RF spectrum as the playfield for setting up the environment and elaborates on strategic wireless communication interference and countermeasures, ensuring a comprehensive understanding of the mission context.

# 4

## UAV Swarm Intelligence

This chapter focuses on UAV swarm intelligence and its application to the problem of communication line interception. The chapter is organized into two main sections. section 4.1 discusses the role of UAVs in signal interference and the advantages of employing UAV swarms for this purpose. Subsequently, section 4.2 provides an overview of swarm intelligence, emphasizing its biological inspiration, and explores its application to UAV swarms in the context of the mission.

### 4.1. UAVs as a Solution to the Problem

UAVs have emerged as a promising solution to address various challenges in wireless communication, particularly in signal interference and monitoring tasks. UAVs can be employed individually or as swarms, significantly enhancing their capabilities in detecting, localizing, and interfering with communication signals.

#### 4.1.1. Role of UAVs in Signal Interference

UAVs can be equipped with specialized payloads and sensors, such as CEMA equipment, which enable them to detect, analyze, and manipulate communication signals effectively. By leveraging these capabilities, UAVs can be deployed to monitor and interfere with communication lines, providing a tactical advantage in various scenarios, including military and security operations.

The flexibility of UAVs in positioning themselves within the CEMA scene allows them to adapt to different environments and operational requirements. [53] This mobility and adaptability make UAVs a valuable asset in signal interference tasks, as they can rapidly change their location to optimize signal reception or interference performance.

In addition to their agility, UAVs can operate at various altitudes, enabling them to access different propagation environments and overcome obstacles such as terrain or urban structures. This flexibility in altitude offers an added advantage in monitoring and interfering with communication signals, as it allows UAVs to adapt their position based on the specific signal characteristics and the surrounding environment.

Overall, the versatile capabilities and maneuverability of UAVs make them a powerful tool in signal interference tasks, enhancing their effectiveness in a wide range of military and security operations.

#### 4.1.2. Advantages of UAV Swarms for Signal Interference

Utilizing UAV swarms offers several benefits in signal interference tasks, which include:

- Signal detection advantage:
  - Multi-bandwidth coverage: UAV swarms can cover a broader range of frequency bands, as band allocation could be done. This enhances their ability to detect various communication signals.

- Downlink Beamforming: By coordinating their actions, UAV swarms can focus their signal reception in specific directions, improving the signal-to-noise ratio and detection capabilities.
- Signal direction/localization:
  - Based on amplitude difference: UAV swarms can estimate the DoA of a signal based on the amplitude differences between the received signals at different UAV locations.
  - Based on phase difference: Swarms can also estimate the DoA based on the phase differences between the signals received at different UAV positions, offering higher resolution in some cases.
- Mapping coverage advantage: UAV swarms can cover larger areas more efficiently than individual UAVs, providing more comprehensive mapping and surveillance capabilities.
- Monitoring advantage due to coverage: The extended coverage offered by UAV swarms improves the ability to monitor communication lines, enhancing situational awareness and operational effectiveness.
- Interference:
  - Uplink Beamforming: UAV swarms can coordinate their actions to focus interference signals in specific directions, increasing the effectiveness of interference techniques and minimizing unintended impacts on other communication systems.

UAV swarms represent a versatile and effective solution for signal interference tasks, offering numerous advantages in detection, localization, monitoring, and active interference. One key attribute of UAV swarms is their robustness, which enables them to adapt and respond to unexpected changes in their environment or mission requirements. This robustness is achieved through decentralized control, redundancy, and self-organization. Additionally, the adaptability of UAV swarms ensures their effectiveness in the face of evolving challenges, while their scalability allows them to accommodate multiple transmitters and communication lines or adapt to larger search areas. The enhanced intelligence of UAV swarms leverages information from different positions at the same timestamp, empowering the swarm to make more informed decisions and execute more effective strategies. As technology continues to advance, UAV swarms will likely become an increasingly important tool in addressing the challenges of wireless communication interference, providing a resilient and adaptable solution for complex scenarios.

## 4.2. Swarm Intelligence and UAV Applications

### 4.2.1. Swarm Intelligence Overview

SI is a field of artificial intelligence that focuses on the collective behavior of decentralized, self-organized systems, typically inspired by natural phenomena such as the behavior of social insects, bird flocking, and fish schooling. [22] The basic idea behind swarm intelligence is that the collective intelligence of a group can surpass the intelligence of its individual members, allowing for more effective problem-solving and decision-making.

SI systems are characterized by their decentralized control, local interactions among agents, and a lack of centralized authority. This leads to emergent behavior, which arises as a result of simple local interactions between agents without any explicit control from a central authority. The emergent behavior often results in global patterns or structures that enable the swarm to effectively achieve its goals.

Swarm intelligence has been successfully applied to a variety of fields, including optimization, robotics, and distributed computing. In the context of (UAVs, swarm intelligence provides a natural framework for designing algorithms and control strategies that leverage the distributed nature and inherent redundancy of UAV swarms. By mimicking the cooperative behavior found in nature, UAV swarms can be programmed to solve complex tasks that would be difficult or impossible for a single UAV to accomplish alone. [5]

### 4.2.2. Biological Inspiration and UAV Swarms

SI is deeply rooted in the observation and understanding of natural systems, particularly those involving the collective behavior of social insects, bird flocking, and fish schooling, see Figure 4.1. These bio-



logical systems exhibit remarkable problem-solving abilities and robustness, which researchers have sought to emulate in artificial systems such as UAV swarms [12, 14].

In the case of social insects, such as ants, bees, and termites, their colonies can perform complex tasks such as foraging, nest building, and path optimization without any central control. Instead, their success is attributed to simple local interactions among individual insects and their environment, mediated by pheromones or other indirect communication methods, a process known as stigmergy [12, 22].

Birds, especially those that form large flocks, exhibit remarkable coordination and synchronization while flying. This phenomenon, known as flocking, is governed by a few simple rules, such as alignment, cohesion, and separation, that dictate the behavior of individual birds. These rules help the flock maintain its structure and avoid predators while minimizing energy expenditure [14].

Similarly, fish schools demonstrate a high degree of coordination and organization, allowing them to navigate their environment efficiently and avoid predators. Research has shown that the schooling behavior of fish is driven by a small set of local interaction rules, which lead to the emergence of complex global patterns [14].



**Figure 4.1:** Examples of natural swarm systems include bird flocks, bee colonies, fish schools, and ant swarms. [66]

Drawing inspiration from these biological systems, UAV swarms can be designed to exhibit similar coordination, robustness, and adaptability. By employing decentralized control, local interactions, and simple rules, UAV swarms can achieve complex tasks such as search and rescue, environmental monitoring, and signal detection and interference [7, 11]. The inherent redundancy and distributed nature of UAV swarms ensure their resilience in the face of challenges, making them well-suited for various applications in robotics and beyond.

Reinforcement learning (RL) is another bio-inspired approach that has been applied to UAV swarms, offering a way for individual UAVs to learn and adapt their behavior based on their interactions with the environment and each other. RL, which is inspired by the learning mechanisms observed in animals, can be used to develop control policies for UAV swarms that can adapt to changing conditions and optimize their performance in complex tasks [36].

By combining swarm intelligence and reinforcement learning techniques, UAV swarms can not only exhibit the coordinated and robust behavior found in nature, but also learn and adapt to dynamic environments and changing mission requirements. This adaptability is crucial for UAV swarms operating in uncertain or adversarial conditions, such as those encountered in signal detection and interference tasks.

The use of bio-inspired approaches like swarm intelligence and reinforcement learning allows UAV swarms to harness the collective intelligence of the group, enabling them to solve complex problems and operate more efficiently. As research in this area continues to progress, it is expected that UAV swarms will become increasingly capable and versatile, finding applications in a wide range of domains

---

and providing solutions to some of the most pressing challenges in robotics, communication, and beyond. In the following chapter, we will dive into the mission planning and task requirements for a UAV swarm in an RF environment, exploring high-level strategies and reviewing various techniques and algorithms that allow the swarm to effectively execute the mission.



# 5

## Mission Planning & Task Discussion

This chapter delves into the intricacies of mission planning and task requirements for UAV swarms in complex environments. section 5.1 discusses mission planning, elaborating on the requirements for each mission phase. In section 5.2, we explore the process of task allocation, focusing on cooperative task allocation for UAV swarms and comparing centralized, decentralized, and distributed task allocation approaches. section 5.3 examines searching strategies, including UAV swarm-based search techniques and adaptive and dynamic searching methods.

Mapping techniques are the subject of section 5.4, which covers UAV swarm-based mapping methods and real-time, collaborative mapping strategies. Lastly, section 5.5 addresses monitoring and interfering, discussing UAV swarm-based monitoring and interference strategies, as well as adaptive and collaborative digital communication interference.

### 5.1. Mission Planning

In this section, the mission planning strategy for a swarm of UAVs acting in the base scenario described in section 2.2 is introduced. A state-machine mechanism is utilized because it ensures convergence to the solution for the swarm, rather than relying on a random search for the communication lines. It also allows agents to act in multiple phases of the mission. Figure 5.1 presents the high level state-machine for an individual agent, where:

1. Arrived at RT Entry Point
2. Mapping Point is Found
3. Mapping is completed
4. Closest agent to interference point moves there to interfere
5. Interference is done

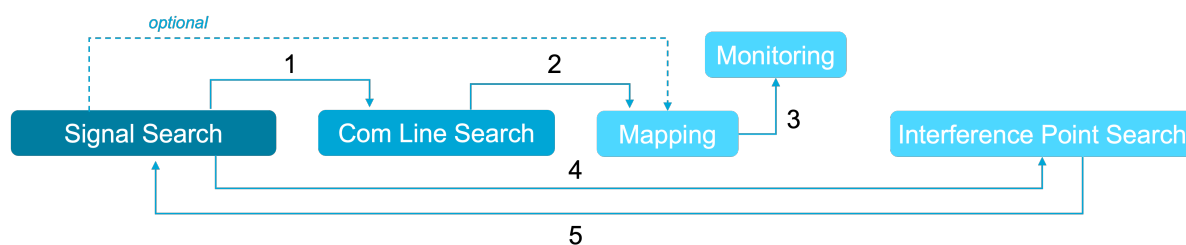
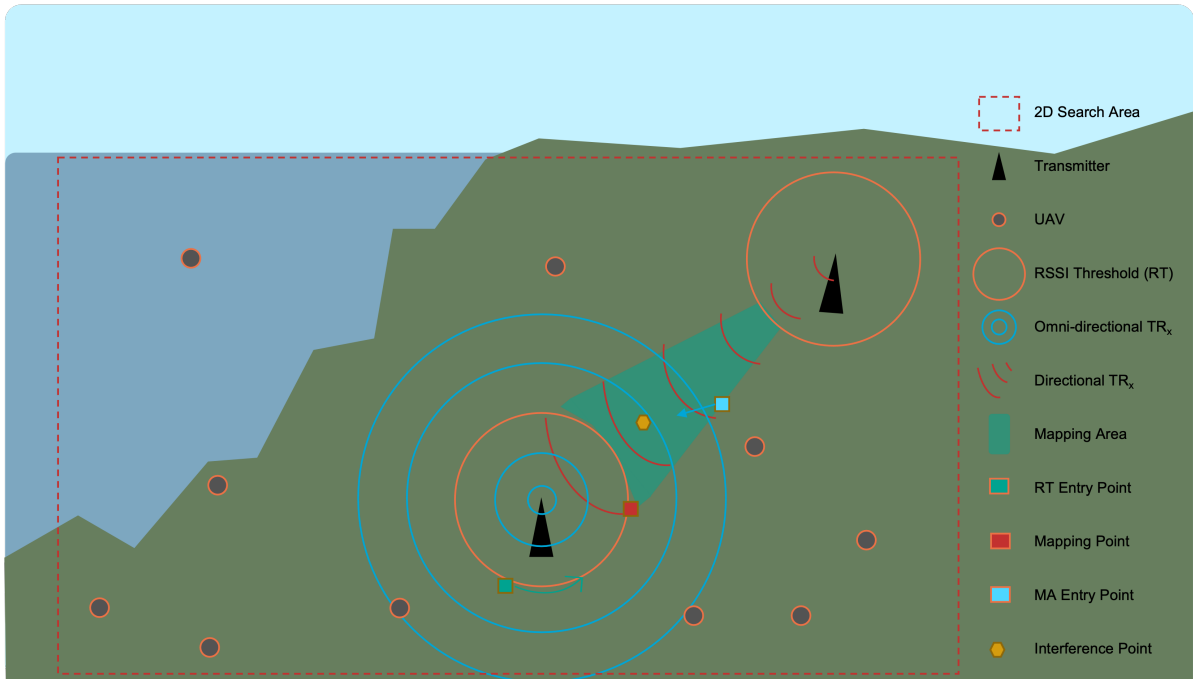


Figure 5.1: State-Machine for agent

The proposed strategy comprises the following steps, as visualized in Figure 5.2:

- Collaboratively find the RT, represented by the orange circle.
- Agent exploits RT to find the mapping point, denoted by the red square.
- Agent initiates the mapping of the communication line, depicted by the green area.
- Upon completion of mapping, the agent continues monitoring the communication line.
- Concurrently, a new agent crosses the MA Entry Point (blue square) and advances towards the interference point (yellow hexagon).



**Figure 5.2:** Visualisation of Mission Flow for UAV swarm

Mapping is a crucial component for the success of the mission in this strategy. It plays a vital role for two primary reasons: firstly, mapping provides essential intelligence for the overall mission, ensuring that the swarm has a comprehensive understanding of the communication lines and their locations. Secondly, mapping could serve as feedback for the interference agent, allowing it to make informed decisions and execute effective interference actions to achieve the desired mission objectives. Figure 5.3 shows the communication line mapping and interference point.

Figure 5.3 displays the communication line mapping and interference point.

To fully implement this strategy, several definitions must be established:

#### Definitions

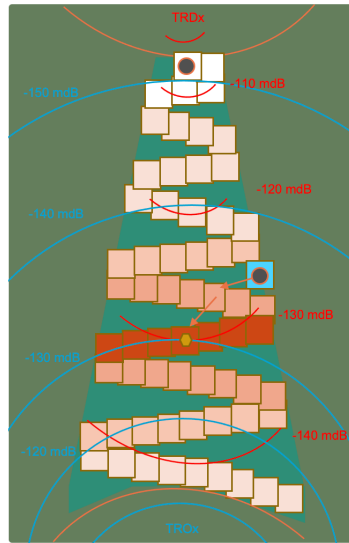
- SNR thresholds for frequency detection (10-20 dB). [44]
- RT for maintaining stealth are predefined with 10-20 mdB. [26]
- A TRO/D is identified when the RT is reached by an agent.
- A communication link is established when bidirectional signals are exchanged between transmitters. An agent can pinpoint the location of the intersection between the received signal and its conjugate signal to determine the mapping point.
- A signal is considered mapped when the mapping area is covered for a TBD percentage (60-80%). The mapping area is defined by the two RTs and two cumulative Cumulative RSSI Line

(CRL)s, ensuring the agent captures sufficient data from both signals.

$$CRL : RSSI_1 + RSSI_2 \geq \text{TBD mdB} \quad (5.1)$$

- An area is deemed monitored if there is no change in the signal strength for a specified duration (2-5 seconds).
- A communication line is interfered with when an agent maintains its position on the interference point for a predefined duration (TBD seconds). The tactical interference point is located where the sum of the two RSSI levels is maximized, ensuring the signal interference is as effective as possible.

$$\text{Interference Point} = \min(RSSI_1 - RSSI_2) \quad (5.2)$$



**Figure 5.3:** Communication Line Mapping & Interference Point

In summary, the mission planning strategy presented here outlines a state-machine mechanism for efficiently managing a swarm of UAVs. The strategy involves collaboratively finding the RT, exploiting the RT to find mapping points, mapping and monitoring communication lines, and interfering with communication lines at the appropriate points. This approach ensures that the swarm converges to a solution, providing valuable intelligence and feedback for mission success.

### 5.1.1. Requirements for Each Mission Phase

This section outlines the general requirements for each phase of the mission planning strategy, ensuring that all essential tasks are covered and efficiently executed. The requirements are organized according to the mission phases: Search, Mapping, Monitoring, Interference, and requirements for Simultaneous Task Execution.

#### 1 Search

- 1.1 Optimize search area coverage and energy consumption balance by collaboratively finding the RT while minimizing energy usage.
- 1.2 Perform comprehensive frequency band scanning by sharing frequency band information and detected signals among agents for efficient collaboration. (desirable)
- 1.3 Balance exploration and exploitation by adapting agent movements based on signal detection and swarm coordination.
- 1.4 Maintain robust swarm communication for seamless collaborative sensing and search optimization.

#### 2 Mapping

- 2.1 Minimize mapping duration by exploiting the RT to find mapping points and streamline agent movements.
- 2.2 Achieve high precision in identifying the signal source and its characteristics by accurately determining the intersection between received signals and their conjugate signals.
- 2.3 Preserve stealth during the mapping process by staying below predefined RTs.
- 2.4 Enhance agent collaboration by sharing mapping progress and signal information among agents to adapt to changing signal conditions.

### 3 Monitoring

- 3.1 Sustain continuous surveillance of the signal source and its vicinity by iterating over mapping area.
- 3.2 Promptly detect alterations in signal characteristics, emitter movement, or the emergence of new emitters by continuously scanning the environment.
- 3.3 Effectively monitor the interference drone's performance by sharing interference results among agents. (desirable)
- 3.4 Adapt to dynamic changes in the environment, emitter behavior, and swarm conditions by adjusting agent positions and communication strategies accordingly. (desirable)

### 4 Interference

- 4.1 Maximize interference effectiveness by positioning the agent at the tactical interference point, where both RSSI levels are the closest to equal.
- 4.2 Swiftly adapt to changes in signal characteristics, emitter location, or the detection of new emitters by adjusting the interference agent's position. (desirable)
- 4.3 Continuously evaluate interference effectiveness by monitoring the signal strength and making strategy adjustments as needed. (desirable)

### 5 Simultaneous Task Execution

- 5.1 During Search: Agents can simultaneously explore the search area, maintain stealth, scan frequency bands, measure RSSI levels, collaborate on signal detection, exploit the RTs to find mapping points.
- 5.2 During Mapping: Agents can concurrently define mapping strategies, maintain stealth, share mapping progress and signal information, and collaborate with other agents.
- 5.3 During Monitoring: Agents can simultaneously perform surveillance over mapping area, maintain stealth, detect changes in signal characteristics or emitter movements, monitor interference UAV performance, and maintain communication with fellow agents.
- 5.4 During Interference: Agents can concurrently determine the optimal interference point, maintain stealth, execute interference actions, assess the effectiveness of interference (desirable), and coordinate with monitoring agents.

## 5.2. Task Allocation

Task allocation is a critical aspect of SI for UAV swarms to achieve efficient and coordinated mission success. [14] The objective of task allocation is to distribute tasks among swarm members in a way that optimizes overall performance while minimizing energy consumption and maximizing mission success. In the context of strategic digital communication interference, UAV swarms must perform various tasks within four distinct phases: Searching, Mapping, Monitoring, and Interfering.

Within each phase, different tasks can be allocated to agents based on their capabilities and the current mission requirements. For instance, during the Searching phase, agents may be assigned to perform tasks such as area exploration, exploitation manoeuvres, frequency scanning, collaborative sensing, and energy management. Similarly, tasks can be allocated within the Mapping, Monitoring, and Interfering phases.

By allocating these tasks within each phase and effectively managing transitions between phases, the overall performance of the swarm can be optimized, leading to more efficient and effective mission outcomes. Proper task allocation and coordination are key to achieving mission success.

### 5.2.1. Cooperative Task Allocation for UAV Swarms

Cooperative task allocation in UAV swarms involves multiple UAVs working together to perform assigned tasks, such as searching for a target, mapping an area, or conducting surveillance. Cooperative task allocation has several advantages, including increased efficiency, reduced mission time, and improved robustness to individual UAV failures. Literature has identified several specific tasks that UAVs may need to perform in order to achieve this goal.

1. Area partitioning: Divide the search or mapping area into subregions to be explored by individual UAVs. This allows for efficient coverage and prevents duplication of efforts. [47]
2. Path planning: Calculate optimal paths for each UAV to follow, ensuring that they cover their designated regions while minimizing energy consumption and avoiding obstacles. [27]
3. Communication and data sharing: Establish communication links between UAVs to share information about their findings, such as the location of targets, obstacles, or points of interest. This enables the swarm to update its knowledge and adapt its behavior accordingly. Even division of labor can emerge in homogeneous populations from communication links. [40]
4. Target tracking: In surveillance missions, some UAVs may be assigned to track specific targets, such as vehicles or individuals. These UAVs will need to maintain a safe distance while keeping the target in sight, adjusting their positions as the target moves. [72]
5. Dynamic task reassignment: As the mission progresses, the circumstances or priorities may change, requiring UAVs to adapt by reallocating tasks among themselves. For instance, if a UAV fails or depletes its battery, another UAV may need to take over its task. [69]
6. Formation control: In certain scenarios, it may be beneficial for the UAV swarm to maintain a specific formation, such as a grid or a line. This requires the UAVs to coordinate their positions and velocities to maintain the desired formation while avoiding collisions. [63]
7. Environmental monitoring: Some missions may require UAVs to gather environmental data, such as temperature, humidity, or air quality. These UAVs will need to perform sensor fusion, combining data from multiple sensors to generate an accurate representation of the environment. [65]
8. Obstacle avoidance: As UAVs navigate through their assigned areas, they must detect and avoid obstacles, such as buildings, trees, or other UAVs. This requires real-time processing of sensor data and coordination with other UAVs to ensure safe operation. [2]

In recent years, various task allocation strategies have been proposed to optimize multi-agent systems. Jang et al. (2017) [17] introduced the GRAPE framework, a decentralized game-theoretic approach for multi-robot task allocation based on anonymous hedonic games. The framework utilizes a merge-and-split algorithm (Apt and Witzel, 2009) for agents to converge to a Nash stable partition in a decentralized manner. A "round" represents an iterative step where an arbitrary agent evaluates all available task-coalition pairs given the current partition and determines which coalition to join. Each agent is assumed to be able to communicate its status to all other agents in every round. The results showed that the framework is scalable, requiring an average of 510 and at most 564 rounds for agents to converge to an agreed solution. Additionally, the solutions provided by the framework guarantee 75% of the global optimality. The study also included an example of the visualized task allocation results to provide a better understanding of the partitioning of agents based on their position with respect to the tasks.

Arnold et al. (2020) [7] presented a framework for task allocation in search and rescue scenarios using heterogeneous drones with different personality types. The personality types include Relay, Social Searcher, and Anti-Social Searcher drones, with each type focusing on a specific aspect of the search and rescue process. The results suggest that the rescue time is strongly associated with the proportion of Relay drones and the size of the swarm. Additionally, the addition of Relay drones to an existing swarm degrades the overall performance of the swarm except when the swarm was entirely composed of Relay drones. The study concludes that the Relay drones, with the behavioral profile implemented

in the DroneLab simulator, antagonistically interact with the Social Searcher and Anti-Social Searcher drones, and suggests that more work is needed to verify the results.

Garcia et al. (2016) [19] proposed a task allocation principle based on a cost matrix that evaluates the expected cost of each vehicle performing each task. The algorithm uses bid vectors and communication to resolve conflicts and produce conflict-free assignments. The algorithm can produce conflict-free assignments under some assumptions and for sufficiently large cost matrix thresholds. Additionally, the paper proposes an adaptive re-planning method to deal with disruptions caused by wind disturbances. Overall, the method reduces inter-agent communication and achieves a trade-off between communication reduction and conflict-free assignment plans.

Papaioannou et al. (2023) [18] proposed a cooperative simultaneous tracking and jamming (CSTJ) system for autonomous UAV agents. Each agent has two specific tasks: a tracking control module responsible for maintaining tracking of the rogue drone and a jamming control module responsible for maximizing the received jamming power at the rogue drone while maintaining interference constraints.

In the context of UAV swarms for signal detection, mapping, monitoring, and interference, task allocation can be informed by various approaches derived from assignment and scheduling, multi-agent decision-making. Online scheduling approaches are useful for allocating tasks among swarm members in real-time, as tasks are often observed and need to be addressed promptly [6]. Hard real-time tasks with time window constraints for completion can be relevant in a dynamic environment with moving signal emitters [21]. Scheduling under uncertainty can be applied to handle the unpredictable nature of signal characteristics, emitter behavior, and environmental factors [15].

The Markov Decision Process (MDP) model can be extended to multi-agent MDP (MMDP) for UAV swarms to coordinate their activities for signal detection, mapping, monitoring, and interference [13]. Reinforcement learning techniques can be employed to learn the values of different states and actions in this multi-agent setting [13].

The literature on task allocation for multi-agent systems presents various approaches, including assignment and scheduling, multi-agent decision-making. For UAV swarms engaged in signal detection, mapping, monitoring, and interference, it is crucial to address real-time responsiveness and handling uncertainties associated with signal characteristics, emitter behavior, and environmental factors.

Jang et al. (2017) [17] explored a game-theoretic approach, which may not be directly applicable to the mission's task allocation due to challenges in designing suitable utility functions and assessing Nash equilibrium quality. Arnold et al. (2020) [7] focused on search and rescue scenarios, which may not be directly transferable to other contexts. Garcia et al. (2016) [19] proposed a bidding algorithm suitable for various UAV swarm tasks, offering real-time responsiveness and adaptability. Finally, Papaioannou et al. (2023) [18] proposed a system relevant to UAV tasks but focused on specific tracking and jamming tasks instead of general task allocation.

Online scheduling approaches are useful for real-time task allocation among swarm members in dynamic environments [6]. However, the immediate outcome of the allocation decision may not be directly interpretable as good or bad, which makes it difficult to assess the quality of an allocation decision at the moment it is made [15]. This lack of immediate feedback makes it challenging for online scheduling algorithms to optimize their decisions effectively in such complex scenarios.

The MDP model can be extended to multi-agent MDP (MMDP) for UAV swarms to coordinate their activities [13]. Reinforcement learning techniques can be employed to learn the values of different states and actions in this multi-agent setting, providing adaptability and real-time responsiveness [13].

Rule-based and bidding algorithms for task allocation have proven effective in multi-agent systems, offering real-time responsiveness, reduced communication overhead, and adaptability. Garcia et al. (2016) [19] proposed a bidding algorithm for task allocation that offers real-time responsiveness and adaptability, making it a suitable choice for a variety of UAV swarm tasks.

In conclusion, Table 5.1 shows the discussed approaches for UAV swarm task allocation, where each approach has its advantages and disadvantages. A green color in the initial cell indicates that the approach is favorable for the use case presented in this report. Yellow indicates that it is less desirable, while red indicates that it is not advisable to continue with the respective approach within the scope of

this study. Based on the trade-off, the rule-based approach is recommended to adopt for both single-agent sequential and multi-agent parallel tasks within the proposed mission planning strategy. This approach ensures convergence to a solution and proper task allocation. By employing rule-based task allocation principles, which are defined by human logic and emphasize optimality and practicality, we address the challenges of coordination, real-time responsiveness, and uncertainty in signal detection, mapping, monitoring, and interference tasks. While other approaches, such as online scheduling, may offer real-time responsiveness, they might struggle to optimize decisions effectively due to the lack of immediate feedback on the allocation decision quality.

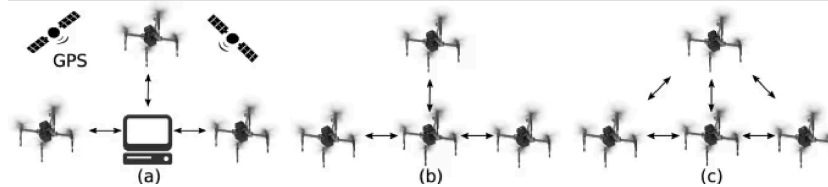
**Table 5.1:** Qualitative Trade-off for Approaches in Task Allocation

<b>Approach</b>	<b>Advantages</b>	<b>Disadvantages</b>
Rule-Based Algorithms	<ul style="list-style-type: none"> <li>• Easy to implement for the proposed mission</li> <li>• Ensures convergence</li> <li>• Focuses on optimality and practicality</li> <li>• Addresses coordination and real-time responsiveness</li> </ul>	<ul style="list-style-type: none"> <li>• Less adaptive than reinforcement learning</li> <li>• Rules need careful definition</li> </ul>
Bidding Algorithms	<ul style="list-style-type: none"> <li>• Robust and efficient solutions</li> <li>• Promotes competition among agents</li> <li>• Might be better in overall performance than rule-based</li> </ul>	<ul style="list-style-type: none"> <li>• Difficulties in optimizing decisions</li> <li>• Unforeseen unwanted task allocation might appear</li> <li>• Needs parameter fine-tuning</li> </ul>
Online Scheduling	<ul style="list-style-type: none"> <li>• Real-time task allocation</li> <li>• Applicable in dynamic environments</li> </ul>	<ul style="list-style-type: none"> <li>• Immediate outcomes not directly interpretable</li> <li>• Difficult to assess allocation quality</li> <li>• Optimization challenging without feedback</li> </ul>
MMDP RL	<ul style="list-style-type: none"> <li>• Robust and efficient solutions</li> <li>• Adaptive learning for agents</li> </ul>	<ul style="list-style-type: none"> <li>• Convergence may take time</li> <li>• Higher computational resources</li> <li>• Difficult implementation</li> </ul>
GRAPE	<ul style="list-style-type: none"> <li>• Decentralized approach</li> <li>• Scalable convergence</li> <li>• 75% of global optimality [17]</li> </ul>	<ul style="list-style-type: none"> <li>• May not be directly applicable to the mission</li> <li>• Challenges in utility function design</li> <li>• Difficulties in assessing Nash equilibrium</li> </ul>

### 5.2.2. Centralized, Decentralized, and Distributed Task Allocation Approaches

Efficient task allocation in UAV swarms is vital for optimal operations. There are three primary approaches for task allocation: centralized, decentralized, and distributed methods, as illustrated in Figure 5.4.





**Figure 5.4:** Communication Line Mapping & Interference Point

Centralized methods rely on a single central control unit or ground station to coordinate swarm actions, providing better global optimization and planning but with increased complexity as the system scales. Decentralized approaches emphasize local communication and decision-making among multiple individual UAVs, striking a balance between centralized and distributed methods while scaling more easily with an increasing number of agents or variables [34]. In contrast, distributed planning involves all UAVs in the swarm working together and sharing information to make collective decisions. This approach allows for better coordination and information sharing compared to decentralized methods, while avoiding the single point of failure and scalability issues associated with centralized methods.

Decentralized strategies, such as self-organization and stigmergy, have proven effective in search, mapping, and surveillance missions. They enable UAV swarms to adapt to changing environments and maintain performance despite the loss of individual drones [35, 40].

Centralized approaches assign tasks to individual UAVs based on their capabilities and positions, enabling better global optimization and planning. However, these approaches may be more vulnerable to the loss of the central control unit, which could significantly impact swarm performance [3].

Table 5.2 gives a qualitative trade-off on the three approaches. A hybrid approach, combining the benefits of centralized, and decentralized methods, may offer the most effective solution for task allocation in UAV swarms [16]. In scenarios such as strategic digital communication interference, centralized communication between the agents is crucial for optimizing overall performance, minimizing energy consumption, and maximizing mission success. Implementing a decentralized approach for task allocation allows every agent the potential to be assigned a specific task based on rules. Once an agent is in a certain phase, it can delegate tasks to other agents as needed. Balancing the strengths of approaches enables UAV swarms to achieve efficient and coordinated operations while maintaining resilience against potential challenges.

**Table 5.2:** Qualitative Trade-off for Centralized, Decentralized, and Distributed Approaches

Approach	Advantages	Disadvantages
Centralized	<ul style="list-style-type: none"> <li>Better global optimization and planning</li> <li>Task assignment based on capabilities and positions</li> </ul>	<ul style="list-style-type: none"> <li>Single complex machine</li> <li>Vulnerable to loss of this central control unit</li> <li>Limited scalability with increasing agents</li> </ul>
Decentralized	<ul style="list-style-type: none"> <li>Scales more easily with increasing number of agents</li> <li>Adaptive to changing environments</li> <li>Maintains performance despite the loss of individual drones</li> </ul>	<ul style="list-style-type: none"> <li>Less global optimization compared to centralized methods</li> <li>Local communication and decision-making</li> </ul>
Distributed	<ul style="list-style-type: none"> <li>Needs less infrastructure</li> <li>Avoids single point of failure</li> </ul>	<ul style="list-style-type: none"> <li>Potentially higher communication overhead</li> <li>May require more complex algorithms for coordination</li> <li>May not converge and be optimal</li> </ul>



## 5.3. Searching

UAV swarm searching is a process where multiple UAVs work together to locate specific targets or areas of interest in a given environment. In this section, we will discuss different UAV swarm-based search strategies and explore adaptive and dynamic searching algorithms.

### 5.3.1. UAV Swarm-based Search Strategies

There are several search strategies that can be employed by UAV swarms to improve the efficiency and effectiveness of a search operation. Some notable examples are:

#### Exhaustive Grid Search

In this strategy, the search area is divided into a grid, and each UAV is assigned a specific region to cover. The UAVs traverse their respective regions in a predetermined pattern (e.g., row-by-row or zigzag) until the entire area is covered. While this method ensures comprehensive coverage, it may not be the most efficient approach, especially for large or complex environments. [47]

#### Probabilistic Search

Probabilistic search algorithms use statistical information about the target's likely location to guide the search process. For example, the UAVs could be directed to prioritize areas with a higher probability of containing the target based on historical data or environmental features. This can significantly reduce the time required to locate the target compared to an exhaustive search. [29]

#### Bio-inspired Search

This strategy involves modeling the search behavior of UAV swarms after biological systems, such as the foraging behavior of ants, the flocking patterns of birds, or the honeybee waggle dance. By mimicking these natural processes, the swarm can achieve a more efficient and coordinated search. Some bio-inspired search algorithms include Ant Colony Optimization (ACO), PSO, and Bee Colony Optimization (BCO). [32]

**ACO:** ACO is inspired by the foraging behavior of ants, where they find the shortest path to food sources by depositing pheromones on their trails. In the context of UAV swarm searching, each UAV represents an ant, and the search area is divided into discrete nodes. As the UAVs traverse the area, they deposit virtual pheromones on their paths, with higher concentrations indicating more promising search routes. Over time, the swarm converges on the optimal search path based on the collective pheromone information. [32]

**PSO:** PSO is based on the social behavior of bird flocks or fish schools, where individuals move and adapt their positions based on their own experiences and those of their neighbors. In a UAV swarm, each UAV represents a particle with a position and velocity in the search space. The UAVs adjust their positions iteratively based on their own best-known position and the swarm's global best-known position. [32] PSO can be enhanced with Reinforcement Learning (RL) techniques to better adapt to dynamic environments and improve search efficiency. By incorporating RL, the UAVs can learn from their experiences and the experiences of the swarm, updating their search strategies accordingly. [20]

**BCO:** BCO or Artificial Bee Colony (ABC) algorithm is inspired by the foraging behavior of honeybees, which communicate the location of food sources through the waggle dance. In the context of UAV swarm searching, the swarm is divided into scout bees and worker bees. The scout bees explore the search area for potential targets, while the worker bees exploit the information gathered by the scouts to narrow down the search area. By using this division of labor and communication mechanism, the UAV swarm can effectively balance exploration and exploitation, leading to an efficient search process. [32]

A qualitative trade-off is given in Table 5.3. Bio-inspired search algorithms offer unique and effective ways to model UAV swarm searching behaviors based on natural systems. These algorithms, such as ACO, PSO, and BCO, can lead to more efficient and coordinated search operations. PSO stands out as an effective and versatile method for UAV swarm searching, enabling efficient and coordinated operations in various environments and scenarios. Its adaptability for new scenarios, good convergence properties, and scalability make PSO particularly well-suited for dynamic search environments. For more challenging environments the PSO could make use of some worker division properties of

the BCO. By employing these bio-inspired algorithms, UAV swarms can achieve great adaptability and effectiveness in their search operations.

**Table 5.3:** Qualitative Trade-off for Search Algorithms

Approach	Advantages	Disadvantages
Exhaustive Grid Search	<ul style="list-style-type: none"> <li>• Comprehensive coverage</li> <li>• Simple to implement</li> </ul>	<ul style="list-style-type: none"> <li>• Inefficient for large or complex environments</li> <li>• Does not make use of multi-agent resourcing</li> </ul>
Probabilistic Search	<ul style="list-style-type: none"> <li>• Uses statistical information to prioritize areas</li> <li>• Reduces search time compared to exhaustive search</li> </ul>	<ul style="list-style-type: none"> <li>• Requires reliable historical data or environmental features</li> <li>• Performance depends on accuracy of probability estimates</li> </ul>
ACO	<ul style="list-style-type: none"> <li>• Converges on optimal search path</li> <li>• Collective pheromone information</li> </ul>	<ul style="list-style-type: none"> <li>• Requires many iterations</li> <li>• Sensitive to pheromone parameters</li> </ul>
PSO	<ul style="list-style-type: none"> <li>• Utilises swarm advantage</li> <li>• Good convergence and scalability</li> <li>• Relatively easy to implement</li> </ul>	<ul style="list-style-type: none"> <li>• Sensitive to parameter selection</li> <li>• Requires fine-tuning for different environments</li> </ul>
BCO	<ul style="list-style-type: none"> <li>• Division of labor between scout and worker bees</li> <li>• Balances exploration and exploitation</li> </ul>	<ul style="list-style-type: none"> <li>• Sensitive to parameter selection</li> <li>• Requires fine-tuning for different environments</li> </ul>

### 5.3.2. Adaptive and Dynamic Searching

Adaptive and dynamic searching algorithms enable UAV swarms to adjust their search strategies in response to real-time information and changing conditions. This can lead to significant improvements in search efficiency and effectiveness. Some notable examples of adaptive and dynamic searching algorithms are:

**Dynamic Task Allocation:** This algorithm allows UAVs to dynamically allocate search tasks based on their individual capabilities and the requirements of the search mission. The swarm's overall performance is optimized by assigning the most suitable UAVs to each task, considering factors such as flight endurance, sensor capabilities, and local environmental conditions. [7]

**Multi-objective Optimization:** In some search scenarios, UAV swarms may need to consider multiple objectives, such as minimizing search time, energy consumption, and risk of detection. Multi-objective optimization algorithms allow UAV swarms to balance these competing objectives while searching for the target. One example of a multi-objective optimization algorithm is the Non-dominated Sorting Particle Swarm Optimizer (NSPSO). [4]

**Reactive Search:** Reactive search algorithms enable UAV swarms to respond to real-time changes in the environment, such as the sudden appearance of obstacles or the discovery of new target-related information. In these cases, the swarm can adapt its search strategy by adjusting the search area, changing the search formation, or prioritizing areas of interest. [8]

**Collaborative Learning:** In this approach, UAVs in a swarm share information and learn from each other during the search process. By exchanging data about their individual experiences and observations, the swarm can collectively build a more accurate and up-to-date understanding of the search

environment. This shared knowledge can then be used to guide the swarm's search strategy, leading to a more efficient and effective search operation.[9]

In summary, UAV swarm searching leverages the power of multiple UAVs working together to locate targets or areas of interest in a given environment. By employing various search strategies and adaptive, dynamic algorithms, swarms can significantly improve the efficiency and effectiveness of search operations.

## 5.4. Mapping

Mapping an environment is a critical task for UAVs, as it provides them with a thorough understanding of their surroundings, which is necessary for a variety of applications. The ability to effectively map an environment not only allows UAVs to optimize their performance, enhance their operational safety, and achieve greater autonomy but also enables mission operators to better grasp the situation and make informed decisions. Swarm intelligence algorithms play a vital role in improving the performance of UAVs in mapping tasks by enabling coordination and collaboration among the UAVs. In this section, we discuss the key techniques involved in UAV swarm-based mapping and how real-time and collaborative mapping can be realized using swarm intelligence algorithms.

### 5.4.1. UAV Swarm-based Mapping Techniques

Swarm intelligence algorithms for UAV mapping primarily focus on path planning, coordination, and collaboration among the UAVs. The general goal is to minimize the time required for mapping the environment while ensuring complete coverage and avoiding collisions, but other constraints could hold. Here are some of the key techniques used in UAV swarm-based mapping:

**Cellular Automata-based Methods:** Cellular automata are discrete models consisting of a grid of cells that update their states based on the states of their neighbors. UAVs can be modeled as cells in a cellular automaton, where their movement is determined by the states of the neighboring UAVs. This approach allows for efficient local communication and decentralized decision-making, enabling the swarm to adapt to dynamic environments and navigate complex terrains. [49]

**Exhaustive Grid Search:** In scenarios where the optimal interference point needs to be located, an agent employs an exhaustive grid search method. This approach is suitable for simple environments, particularly when the agent starts from a corner point. The agent zigzags through the communication line area, systematically exploring the search space and evaluating the objective function at each point to find the best point of interference. Although computationally less efficient, it guarantees finding the optimal point within the search space. [47]

**Graph-based Methods:** The environment can be modeled as a graph, where nodes represent discrete locations, and edges represent the connections between these locations. In graph-based methods, UAVs use algorithms like Dijkstra's or A\* to find the shortest path for efficient mapping. By combining these methods with swarm intelligence, UAVs can collaboratively explore and cover the environment, resulting in faster and more accurate mapping. [42]

**Consensus-based Methods:** Consensus-based methods involve the UAVs reaching a common agreement on their tasks and coordination. The swarm uses techniques such as leader-follower, auction-based algorithms, or consensus algorithms to allocate tasks and plan their paths. By achieving consensus, the UAVs can optimize their routes and avoid redundant coverage. [24]

**Frontier-based Exploration:** Frontier-based exploration is a suitable approach for multiple UAVs in scenarios where the total area is unknown. By focusing on the frontiers, the boundaries between known and unknown areas, UAVs can efficiently explore the environment. This algorithm is particularly useful for multi-UAV mapping, as it enables UAVs to distribute the exploration task among themselves, reducing redundant efforts and increasing efficiency. [43]

**Reinforcement Learning for Area Mapping:** Multiple UAVs can employ RL techniques to efficiently learn and adapt their mapping strategies in various scenarios, including disaster area mapping as detailed in the reference [48]. This approach is particularly suitable for dynamic and uncertain environments, where the agents collaboratively explore and map the area. Through continuous interactions with the environment, the UAVs update their knowledge and improve their performance, resulting in

more effective multi-agent mapping. Although computationally more complex, RL-based approaches offer the potential for superior adaptability and coordination among the agents, leading to better overall mapping outcomes. For a more detailed discussion of this approach, refer to section 7.3.

In conclusion, a qualitative trade-off on the various approaches for UAV swarm-based mapping is given in Table 5.4. Each technique is suitable for different scenarios and offers specific advantages. Exhaustive Grid Search mapping is ideal for single-agent scenarios in simple environments where precision is crucial and the starting point is known, as it guarantees finding the optimal point within the search space. On the other hand, Frontier-based Exploration works well for multi-agent scenarios with unknown total areas, as it focuses on efficient exploration, distribution, and coverage by multiple UAVs. Reinforcement Learning for Area Mapping will also be investigated for its adaptability and potential for superior coordination among the agents, which could lead to better overall mapping outcomes in dynamic and uncertain environments.

**Table 5.4:** Qualitative Trade-off for Mapping Techniques

Approach	Advantages	Disadvantages
Cellular Automata-based	<ul style="list-style-type: none"> <li>• Efficient local communication</li> <li>• Decentralized decision-making</li> <li>• Adapts to dynamic environments</li> </ul>	<ul style="list-style-type: none"> <li>• Requires fine-tuning of cell states</li> <li>• Performance depends on initial conditions</li> </ul>
Exhaustive Grid Search	<ul style="list-style-type: none"> <li>• Comprehensive coverage</li> <li>• Simple to implement</li> </ul>	<ul style="list-style-type: none"> <li>• Inefficient for large or complex environments</li> <li>• Limited adaptability to dynamic changes</li> </ul>
Graph-based Methods	<ul style="list-style-type: none"> <li>• Efficient mapping with shortest path algorithms</li> <li>• Collaborative exploration</li> </ul>	<ul style="list-style-type: none"> <li>• Requires graph modeling of the environment</li> <li>• Computationally intensive for large graphs</li> <li>• Does not provide full coverage</li> </ul>
Consensus-based Methods	<ul style="list-style-type: none"> <li>• Optimal route planning</li> <li>• Avoids redundant coverage</li> </ul>	<ul style="list-style-type: none"> <li>• Requires complex communication mechanisms</li> <li>• Can be slow to reach consensus</li> </ul>
Frontier-based Exploration	<ul style="list-style-type: none"> <li>• Efficient exploration of unknown areas</li> <li>• Distributes tasks among multiple UAVs</li> </ul>	<ul style="list-style-type: none"> <li>• Not optimal but converges</li> <li>• Requires effective frontier detection</li> </ul>
RL for Area Mapping	<ul style="list-style-type: none"> <li>• Adapts to dynamic and uncertain environments</li> <li>• Enables collaborative exploration and mapping</li> <li>• Continuous learning and performance improvement</li> </ul>	<ul style="list-style-type: none"> <li>• Computationally complex</li> <li>• Requires tuning of learning parameters</li> </ul>

#### 5.4.2. Real-time and Collaborative Mapping

Real-time and collaborative mapping with UAV swarms require an integrated approach that combines different aspects of swarm intelligence algorithms. This integration allows for efficient communication, data sharing, and coordination among the UAVs, leading to faster and more accurate mapping.

**Dynamic Path Planning** is a crucial aspect of this integration, enabling UAVs to adapt to changes in the environment and adjust their paths accordingly. Techniques such as Exhaustive Grid Search, Frontier-based Exploration, and graph-based methods can be tailored to ensure efficient coverage in dynamic environments.

**Decentralized Decision-making** empowers UAVs to make local decisions based on their observations and information received from neighboring UAVs. This approach preserves the swarm's flexibility and resilience, as it avoids a single point of failure.

**Communication and Data Sharing** are vital for real-time collaboration among the UAVs. By exchanging information about their positions, sensor data, and local maps through various communication protocols, the UAVs can create a comprehensive global map of the environment.

Lastly, **Task Allocation and Coordination** play a critical role in swarm performance. UAVs must dynamically allocate tasks based on their capabilities, proximity, and available resources. Consensus-based methods or auction-based algorithms can be employed to effectively allocate tasks and coordinate the UAVs' actions.

In conclusion, by leveraging the advantages of Exhaustive Grid Search for single-agent scenarios with known starting points and Frontier-based Exploration for multi-agent scenarios in unknown environments, real-time and collaborative mapping can be achieved more effectively.

## 5.5. Monitoring and Interfering

### 5.5.1. UAV Swarm-based Monitoring and Interference Strategies

For UAV swarm-based monitoring strategies, the swarm can continue utilizing the mapping algorithms discussed in section 5.4 to maintain continuous surveillance of the entire communication line. By persistently mapping the environment, agents can monitor changes in signal characteristics, track emitter movement, and detect the emergence of new emitters. In addition, agents can effectively monitor the interference drone's performance by sharing interference results among themselves.

Decentralized cooperative aerial surveillance enables UAV swarms to adapt to dynamic changes in the environment, emitter behavior, and swarm conditions by adjusting agent positions and communication strategies accordingly. This adaptive approach facilitates efficient coordination between monitoring and interference agents, as they can share information about the interference point and any changes in signal characteristics.

### 5.5.2. Adaptive and Collaborative Digital Communication Interference

UAV swarm-based interference techniques focus on maximizing interference effectiveness by positioning the agent at the tactical interference point, where the sum of the two RSSI levels is maximized. To achieve this, the agent can adjust its velocity to move towards the best-mapped point based on the monitoring data collected by the swarm.

Efficient coordination between monitoring and interference agents is facilitated by sharing information about the interference point and any changes in signal characteristics. This enables the interference agent to swiftly adapt to changes in signal characteristics, emitter location, or the detection of new emitters by adjusting its position accordingly.

Continuously evaluating interference effectiveness is crucial for the success of the mission. This can be achieved by monitoring the signal strength and making strategy adjustments as needed based on the feedback from the monitoring agents.

In summary, UAV swarm-based monitoring and interference strategies can leverage the mapping algorithms and techniques discussed earlier to achieve effective surveillance and interference in dynamic environments. By maintaining continuous communication and coordination among agents, the swarm can promptly adapt to changes in the environment and emitter behavior, ensuring the success of the mission.

# 6

## Task Specifications & Algorithms

In this chapter, we describe the different tasks and corresponding algorithms for each state of the UAV swarm: searching, mapping, and monitoring & interfering. We also discuss the hybrid approach of centralized communication and decentralized decision-making, which combines the advantages of both centralized and decentralized systems. This approach enables better coordination and improved situational awareness while maintaining resilience against individual UAV failures. For a detailed discussion on the choice of this hybrid approach, please refer to subsection 5.2.2.

The subsequent sections outline the tasks and algorithms associated with each state, providing insights into how the UAV swarm operates and collaborates to achieve mission success.

### 6.1. Task Architecture

Task allocation in a UAV swarm is crucial for overall performance and efficiency. Efficient task allocation relies on decentralized decision-making among all agents while maintaining centralized communication, ensuring improved coordination and situational awareness while preserving resilience against individual UAV failures.

During the search state, base agents allocate tasks to the swarm through decentralized decision-making. Each UAV follows a set of rules, making local decisions based on their observations and information from neighboring UAVs. Centralized communication enables all agents to exchange information with one another, ensuring effective cooperation.

The choice between sequential and parallel task allocation depends on the nature of the tasks and UAV capabilities. Sequential allocation assigns tasks one at a time, suitable for interdependent tasks or strict prioritization. In contrast, parallel allocation assigns tasks to multiple UAVs simultaneously, allowing concurrent work on independent tasks, maximizing swarm efficiency.

In conclusion, effective task allocation in a UAV swarm relies on decentralized decision-making with centralized communication. This approach ensures efficient and coordinated operations while maintaining resilience against potential challenges. Task allocation can be performed sequentially or in parallel, depending on the nature of the tasks and the capabilities of the UAVs.

### 6.2. Searching Phase

#### 6.2.1. Tasks

The primary objective of the searching phase is to detect the presence of signals within the search area. UAVs navigate the search area to collect signal strength measurements and identify potential communication lines. Each UAV is assigned an initial position and velocity, with the goal of covering the search area as effectively as possible.

### 6.2.2. Algorithms

The PSO algorithm is employed for the searching phase due to its effectiveness and versatility in UAV swarm searching. Each UAV represents a particle in the swarm, updating its position and velocity iteratively based on its own best position and the swarm's global best. PSO's adaptability, good convergence properties, and scalability make it particularly well-suited for dynamic search environments and various scenarios, as discussed in section 5.3.

The objective function for the PSO algorithm is based on the received signal strength  $S_{rx}$ . By optimizing the swarm's positions concerning the received signal strength, the UAVs are guided towards the locations with stronger signals, potentially identifying the communication lines more efficiently. To achieve this, each UAV measures the received signal strength at its current position  $x_i(t)$  and compares it to its previous best position  $p_i(t)$ . If the current position yields a higher signal strength, the UAV updates its personal best position. [32]

Similarly, the global best position  $g(t)$  is updated based on the best signal strength found by any UAV in the swarm. The UAVs then update their velocities  $v_i(t)$  and positions  $x_i(t)$  based on their personal best and the global best positions using the following equations:

$$v_i^{(t+1)} = wv_i^{(t)} + c_1r_1(p_i^{(t)} - x_i^{(t)}) + c_2r_2(g^{(t)} - x_i^{(t)}) \quad (6.1)$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (6.2)$$

where  $v_i(t)$  and  $x_i(t)$  are the velocity and position of UAV  $i$  at iteration  $t$ ,  $w$  is the inertia weight,  $c_1$  and  $c_2$  are acceleration coefficients,  $r_1$  and  $r_2$  are random factors,  $p_i(t)$  is the personal best position of UAV  $i$ , and  $g(t)$  is the global best position of the swarm. [32] The PSO algorithm can be adapted to handle multiple transmission sources, moving transmitters, and other enhancements discussed in Section 2.3.

The pseudo code for the PSO algorithm is given in algorithm 1.

---

#### Algorithm 1 PSO Algorithm [38]

---

```

1: Initialize swarm with random positions and velocities
2: Evaluate fitness of each particle (UAV)
3: Set personal best position  $p_i$  of each particle to its initial position
4: Set global best position  $g$  to the best initial personal best position
5: while termination criteria not met do
6:   for each particle  $i$  do
7:     Update particle velocity:  $v_i^{(t+1)} = wv_i^{(t)} + c_1r_1(p_i^{(t)} - x_i^{(t)}) + c_2r_2(g^{(t)} - x_i^{(t)})$ 
8:     Update particle position:  $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$ 
9:     Evaluate fitness of particle at new position
10:    if new fitness better than  $p_i$  then
11:      Update personal best position  $p_i$  of particle
12:    end if
13:  end for
14:  Update global best position  $g$  to the best personal best position
15: end while

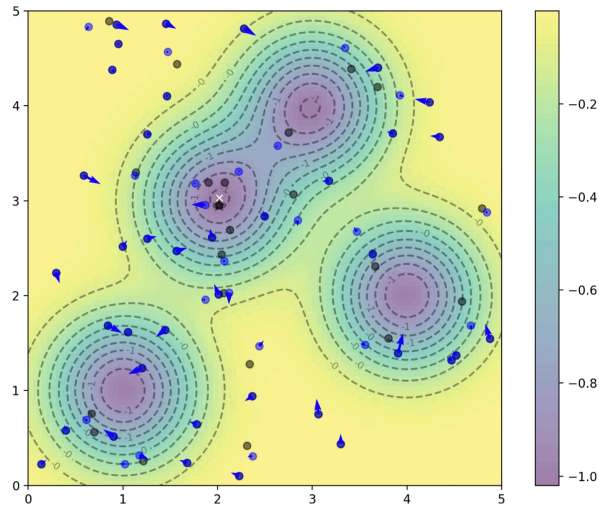
```

---

The algorithm can be adapted to incorporate an exploration factor for individual agents by adjusting their velocity update equation or introducing randomness. This enhancement encourages the agents to explore different regions of the search space, reducing the likelihood of getting trapped in local minima and promoting the discovery of global or multiple optima for the optimization problem. Additionally, similar modifications can be applied to handle moving targets, enabling agents to dynamically adjust their search strategies and effectively track targets as they change position over time.

A visual representation of the Particle Swarm Optimization algorithm in action can be seen in Figure 6.1, where the UAVs dynamically update their positions and velocities to converge towards the optimal solution(s).

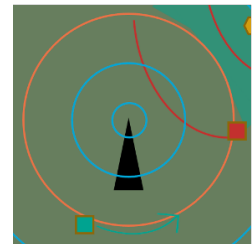




**Figure 6.1:** Particle Swarm Optimization in action: UAVs dynamically update their positions and velocities to converge towards the optimal solution(s) based on their personal best and global best positions.

The PSO algorithm has the potential to be adapted for more complex scenarios, as described in section 2.3. This includes handling multiple transmission sources, moving transmitters, frequency hopping, and other challenging environmental factors. By adjusting the objective function and incorporating additional constraints, the algorithm can be tailored to effectively address these challenges, ensuring efficient and accurate communication line identification in complex RF environments.

Once a UAV arrives at the RT of a transmitter, it can begin searching for both incoming and outgoing transmissions by circling around the threshold line, like shown in Figure 6.2. This allows the UAV to detect and confirm the presence of a communication line more effectively. This method ensures more reliable detection of communication lines compared to a random search, which may not always converge.



**Figure 6.2:** Circling RT until Mapping Point is found

## 6.3. Mapping Phase

### 6.3.1. Tasks

Once a communication line is detected, the UAVs switch to the mapping phase. In this phase, UAVs collaborate to find the interference point. A signal is considered mapped when an interference point is found. The tactical interference point is where the sum of the two SNR levels is maximal, ensuring that the signal interference is as effective as possible.

### 6.3.2. Algorithms

To locate the optimal interference point, an agent employs an exhaustive grid search method. This approach is suitable for simple environments, particularly when the agent starts from a corner point. As discussed in section 5.4, Exhaustive Grid Search mapping is ideal for single-agent scenarios in simple environments where precision is crucial and the starting point is known, as it guarantees finding the optimal point within the search space. The agent zigzags through the communication line area, systematically exploring the search space and evaluating the objective function at each point to find the best point of interference [47]. The objective function in this case is the minimisation of the difference in RSSI levels of the two communication signals.

Although the grid search algorithm is computationally less efficient, it guarantees finding the optimal interference point within the search space. For the mapping phase, it is a viable method as it ensures the highest possible effectiveness in signal interference.

Here is a pseudo code representation of the grid search algorithm applied to the interference point search:



**Algorithm 2** Grid Search for Optimal Interference PointBoundaries of the search grid:  $x_{min}, x_{max}, y_{min}, y_{max}$ Optimal interference point:  $(x', y')$   $max_{objective}$ **for**  $x \leftarrow x_{min} x_{max}$  **do****for**  $y \leftarrow y_{min} y_{max}$  **do**  $objective \leftarrow$  Evaluate objective function at  $(x, y)$ **if**  $objective > max_{objective}$  **then**  $max_{objective} \leftarrow objective$   $(x', y') \leftarrow (x, y)$ 

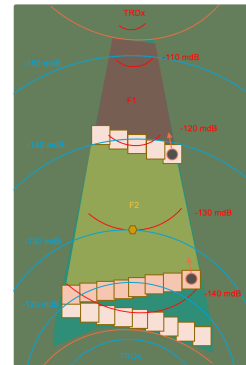
In algorithm 2, the boundaries of the grid are determined by the RT and the side bounds of the directional signal. The UAV explores the grid systematically, calculating the objective function for each point and updating the optimal interference point when a better candidate is found.

**6.3.3. Frontier-based Exploration for Multi-UAV Mapping**

In scenarios where the total area is unknown, and the agent only knows when it hits a border, Frontier-based Exploration can be a suitable approach for multiple UAVs. This approach allows the UAVs to efficiently explore the environment by focusing on the frontiers, which are the boundaries between known and unknown areas, as visualized in Figure 6.3.

Frontier-based Exploration defines frontiers by discretising the environment into a grid of cells, marking cells as explored or unexplored based on the UAVs' observations. Frontiers are the border cells that have at least one unexplored neighbor, guiding the UAVs to prioritize their exploration efforts.

The Frontier-based Exploration algorithm is particularly useful for multi-UAV mapping, as it enables UAVs to distribute the exploration task among themselves, reducing redundant efforts and increasing efficiency. By identifying frontiers and sharing this information among the UAVs, they can coordinate their exploration, avoid revisiting the same areas, and achieve better coverage of the environment. [43]



**Figure 6.3:** Visualisation of Frontier-based Exploration for Multi-UAV Mapping

**Algorithm 3** Multi-UAV Frontier-based Exploration (desirable)UAVs' initial positions:  $UAV\_1\_start, UAV\_2\_start$ 

Global map of the environment Initialize UAVs with their respective starting positions

**while** unexplored frontiers exist **do****for** each UAV **do**

- Explore the environment using sensors and update the local map
- Identify frontiers: boundaries between explored and unexplored areas
- Select the nearest frontier
- Navigate to and explore the selected frontier
- Update the local map with new information from the exploration

Periodically communicate local maps and frontiers among UAVs

Update the global map with the combined information

In algorithm 3, each UAV starts from its initial position and explores the environment independently. As the UAVs encounter borders and identify frontiers, they select the nearest frontier and explore the unexplored space adjacent to it. Periodic communication among UAVs allows them to share information about their local maps and frontiers, helping them coordinate their exploration and avoid redundant efforts.

The main advantage of Frontier-based Exploration for multi-UAV mapping in situations with unknown area boundaries is that it focuses on exploring the boundaries between known and unknown areas. This can lead to more efficient coverage of the environment and quicker identification of the optimal interference point.

By employing Frontier-based Exploration for multi-UAV mapping, the UAVs can explore the environment more effectively and find the optimal interference point faster than with single-UAV methods.

section 7.3 comes with theory on how to implement learning for efficient multi-agent area mapping.

## 6.4. Monitoring and Interfering Phases

### 6.4.1. Tasks

After the communication line is mapped, the closest UAV moves to the interference point, which is the position where an agent experiences the best objective in Equation 5.2. The mapping agent continues the exhaustive grid search to monitor the mapping area and report changes in the environment.

During the monitoring phase, UAVs hover around the communication line, staying out of sight while keeping track of any changes in the signal strength, frequency, or other relevant parameters.

When required to actively disrupt the detected communication line, the UAVs enter the interfering phase. In this phase, the UAV at the interference point attempts to disrupt the communication while staying out of sight.

### 6.4.2. Algorithms

During the interfering phase, the UAV at the interference point alters its speed vector to approach and maintain its position at the interference point, using the previously obtained information about the communication line's location. This allows the UAV to employ signal disruption techniques effectively while remaining undetected. section 7.2 comes with theory on how to implement learning for efficient interference point finding

In the monitoring phase, the mapping agent continues performing the exhaustive grid search algorithm to track any changes in the environment and compare with the previously mapped data.

# 7

## Reinforcement Learning to Enhance Mission Efficiency & Adaptability

This chapter introduces RL techniques to enhance the efficiency and adaptability of the mission, specifically in the tasks of interference point finding and multi-agent mapping of communication lines. Reinforcement learning, a branch of machine learning, enables agents to learn from their interactions with the environment and make decisions based on these experiences. By leveraging RL, the UAVs can improve their ability to locate the optimal interference point and map communication lines more effectively, resulting in more efficient and adaptable mission performance. This is especially desirable where environments are getting to complex to put it in an adequate formula for the agent to work with. This approach becomes particularly valuable in complex environments where it is challenging to develop an adequate formula for the agent to work with effectively.

The chapter serves as a demonstrator for future enhancements, suggesting that incorporating learning techniques into the model might provide benefits over traditional algorithmic approaches. The discussion begins with a brief overview of reinforcement learning, followed by the application of RL algorithms to interference point finding and multi-agent mapping of communication lines. In each case, the problem formulation, the proposed RL algorithm, and the results are presented, demonstrating the potential of reinforcement learning to optimize the mission's efficiency and adaptability.

### 7.1. Reinforcement Learning

RL is a machine learning technique that focuses on training agents to make decisions based on their interactions with an environment. A small diagram of a generic RL flow with State (S), Action (A), Reward (R), is given in Figure 7.1. In RL, an agent learns to perform actions that maximize cumulative rewards over time by constantly exploring and exploiting its knowledge of the environment.

An RL problem is typically characterized by the following components:

- **State (S):** A representation of the agent's current situation in the environment.
- **Action (A):** A set of possible moves or decisions the agent can take in a given state.
- **Reward (R):** A scalar feedback signal that indicates how well the agent is performing its task. The agent's goal is to maximize the cumulative reward over time.
- **Policy ( $\pi$ ):** A mapping from states to actions, which defines the agent's behavior. The policy is typically learned through interactions with the environment.
- **Value function (V):** A function that estimates the expected cumulative reward for each state, given a specific policy.

The learning process in RL involves updating the agent's policy to maximize the expected cumulative reward. There are two main approaches for learning the optimal policy: value-based methods and policy-gradient methods. [61]

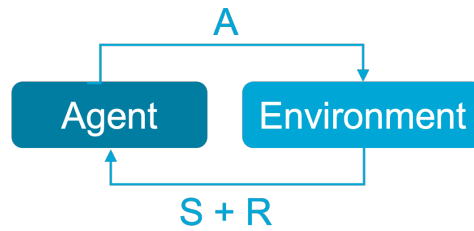


Figure 7.1: Generic flow RL agent

**Value-based methods** focus on learning an optimal value function, which is then used to derive the optimal policy. The agent iteratively updates the value function using techniques such as Q-learning or Deep Q-Networks (DQN) until it converges to the optimal value function. [45]

**Policy-gradient methods** directly optimize the policy by adjusting its parameters to maximize the expected cumulative reward. These methods compute the gradient of the expected reward with respect to the policy parameters and apply gradient ascent to update the policy. Examples of policy-gradient methods include the REINFORCE algorithm. [67]

**Actor-critic methods**, such as Proximal Policy Optimization (PPO) [57], combine the strengths of both value-based and policy-gradient approaches. They employ separate networks for the policy (actor) and value function (critic). The actor is updated using policy gradients, while the critic is updated using a value-based approach. PPO introduces a surrogate objective function with a trust region constraint to stabilize the learning process, making it more sample-efficient and robust compared to other policy-gradient methods.

In the context of multi-agent systems, there are several approaches to organizing and governing the interactions between agents, mainly centralized multi-agent systems and distributed systems. Centralized systems have a single authority controlling multiple agents or a single agent, making decisions on their behalf and coordinating their behavior. This central system can potentially lead to better performance but may face issues related to single points of failure or communication bottlenecks. Distributed systems involve decentralized interactions between agents, where each agent makes decisions based on its knowledge and information from others. Agents can either learn from each other or not, depending on the implementation. This approach promotes adaptability and resilience but may face challenges in achieving global coordination. [46]

In the following sections, RL will be applied to improve the efficiency and adaptability of the UAVs in the tasks of interference point finding and multi-agent mapping of communication lines. The exact RL algorithms are still to be determined as an optimal one will be chosen in the development phase.

## 7.2. RL for Interference Point Finding (critical)

In this section, we will discuss how Reinforcement Learning (RL) can be applied to improve the efficiency and adaptability of finding the interference point. We will define the state, action, and reward components of the RL problem to optimize the search process for the interference point. The environment is depicted in Figure 7.2.

### 7.2.1. State Representation

The state representation includes the following information:

- Relative position of the UAV with respect to the mapping point (corner point of the communication line area):  $(\Delta x, \Delta y)$
- RSSI of both communication lines:  $(RSSI_1, RSSI_2)$
- Mapping area boundaries defined by RT and CRL.

The state can be represented as  $s = (\Delta x, \Delta y, RSSI_1, RSSI_2)$ .

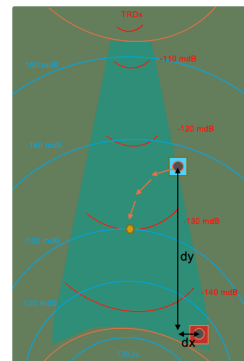


Figure 7.2: RL Environment for Interference Point Search

### 7.2.2. Action Space

The action space consists of adjusting the UAVs' velocity in the x and y directions and choosing the interference point. Actions can be represented as:

- Adjust velocity in the x direction:  $a_x$
- Adjust velocity in the y direction:  $a_y$
- Pick interference point:  $a_p$

An action can be represented as  $a = (a_x, a_y, a_p)$ .

### 7.2.3. Reward Function

The reward function aims to guide the agent towards finding the interference point as quickly and accurately as possible. The reward function can be designed as follows:

- Positive reward for decreasing the difference in RSSI levels:  $r_{RSSI} = k_1(RSSI_1 - RSSI_2)$ , where  $k_1 > 0$  is a scaling factor.
- Negative reward for time spent searching:  $r_{time} = -k_2\Delta t$ , where  $k_2 > 0$  is a scaling factor and  $\Delta t$  is the time step.
- Large positive reward for successfully identifying the interference point within the mapping area constraints.
- Negative reward for moving outside the mapping area boundaries.

The total reward for a given time step can be calculated as  $R = r_{RSSI} + r_{time}$ .

## 7.3. RL for Multi-agent Mapping of Communication Line (desirable)

In this section, we will discuss how Reinforcement Learning (RL) can be applied to improve the efficiency and adaptability of multi-agent mapping of the communication line. We will define the state, action, and reward components of the RL problem to optimize the collaborative mapping process, drawing inspiration from the paper on using RL for disaster area mapping with drones. [48]



	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0
2	0	1	1	1	1	1	0	0	0	0
3	0	0	0	0		1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0		0	0
7	0	0	0	0	0	0	1	1	0	0
8	0	0	0	0	1	1	1	0	0	0
9	0	0	0	0	1	1	1	1	1	1

Figure 7.3: Discretized mission environment into a grid consisting of  $m \times m$  square cells. [48]

### 7.3.1. State Representation

The state representation for each UAV includes the following information:

- Location of the UAV with respect to the corner point of the communication line area:  $h_i(t) = [h_i^x(t), h_i^y(t)]$ , where  $h_i^x(t)$  and  $h_i^y(t)$  are the corresponding x and y coordinates.

- Local mapping status of the four neighboring cells:  $s_i(t) \in S^4$ , referring to the mapping status matrix  $M$ .

The state for UAV  $i$  can be represented as  $s_i = (h_i(t), s_i(t))$ .

### 7.3.2. Action Space

The action space for each UAV consists of four possible directions:

- Move up:  $a_{up}$
- Move down:  $a_{down}$
- Move left:  $a_{left}$
- Move right:  $a_{right}$

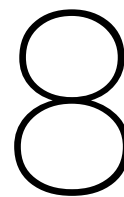
An action for UAV  $i$  can be represented as  $a_i \in a_{up}, a_{down}, a_{left}, a_{right}$ .

### 7.3.3. Reward Function

The reward function aims to guide the agents towards efficiently mapping the communication line. The reward function can be designed as follows:

- Reward  $r_i(t) = 10$  when a drone visits a new cell, i.e.,  $M[h_i^x(t), h_i^y(t)] = 0$ .
- Reward  $r_i(t) = 0$  otherwise, i.e.,  $M[h_i^x(t), h_i^y(t)] = 1$ .

The total reward for UAV  $i$  at a given time step can be calculated as  $R_i = r_i(t)$ .



# Research Proposal

The use of a single UAV for detection, localization, mapping, and interfering with communication lines within the CEMA domain has been the initial approach [53]. However, this method has proven to be ineffective, particularly in complex RF environments. The primary objective of this thesis is to develop and evaluate a novel, bio-inspired swarm intelligence state-machine algorithm for a swarm of Unmanned Aerial Vehicles (UAVs) in the CEMA domain. The swarm-based approach aims to effectively detect, locate, map, and interfere with communication lines in complex RF environments, leveraging the collective intelligence and distributed capabilities of multiple UAVs.

This research aims to contribute new knowledge to the CEMA domain by investigating the use of UAV swarm intelligence to search for signals, map the area, and optimally interfere with communication lines using reinforcement learning. Additionally, a model will be developed to simulate various scenarios, providing insights into the planning and execution of such missions. This may result in a testing environment with proposed strategies, and a list of requirements to consider for real-world implementation.

Furthermore, this research will contribute to the field of swarm intelligence by identifying the most effective algorithms for signal recognition, search, mapping, and monitoring in the CEMA context. By exploring the applicability of existing algorithms and developing new ones, this thesis aims to advance the understanding of swarm intelligence techniques for communication interference missions. The outcomes of this research will provide valuable insights into the algorithms to use or avoid in the CEMA domain and demonstrate the potential of repurposing algorithms from other domains for this purpose.

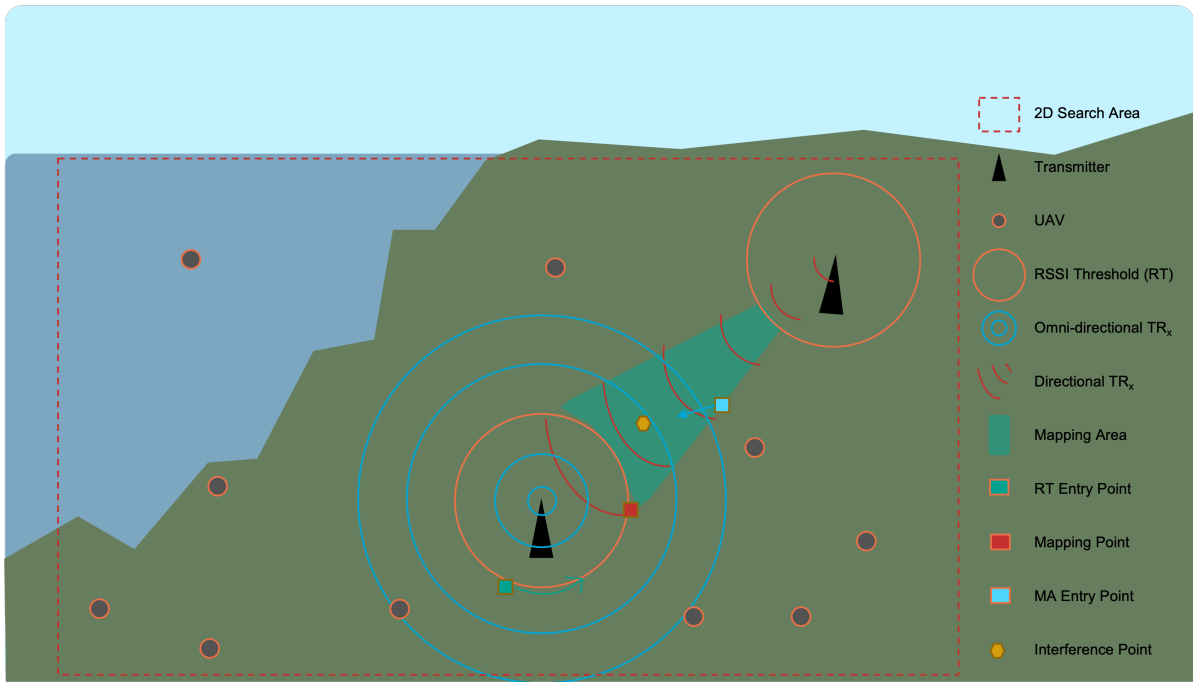


Figure 8.1: Visualisation of Mission Flow for UAV swarm

## 8.1. Research Questions

In order to address the challenges of employing a swarm of UAVs in complex RF environments, the following research questions have been formulated:

1. How can a state-machine swarm intelligence model be developed for UAV-based detection, localization, and mapping of communication lines in a base RF environment visualised in Figure 8.1, and how can the model be evaluated in terms of the evaluation metrics?
2. How can the swarm intelligence model adapt to various environment enhancements, such as increased transmitters, dynamic scenarios with moving transmitters, complex noise patterns, and frequency-hopping transmitters, and how do these enhancements affect the evaluation metrics?
3. How can the model address UAV performance enhancements, including different release points, flying time constraints, UAV fall-out, adaptive search areas, and unknown bandwidths, while maintaining effective detection, localization, and interference capabilities as measured by the evaluation metrics?
4. How can strategic algorithm enhancements, such as incorporating collaborative techniques (beamforming, phase difference, triangulation) and the FSPL formula, improve the UAV swarm's performance in complex RF environments, specifically in terms of direction-of-arrival estimation, distance estimation, and the evaluation metrics?

## 8.2. Methodology

### 8.2.1. Strategy

The proposed strategy for the swarm of UAVs consists of several steps, which are visualized in Figure 5.2. Each step is designed to contribute to the overall goal of detecting, locating, mapping, and interfering with communication lines in complex RF environments.

- Collaboratively find the RT, represented by the orange circle.
- Agent exploits RT to find the mapping point, denoted by the red square.
- Agent initiates the mapping of the communication line, depicted by the green area.



- Upon completion of mapping, the agent continues monitoring the communication line.
- Concurrently, a new agent crosses the MA Entry Point (blue square) and advances towards the interference point (yellow hexagon).

### 8.2.2. Algorithmic Approaches

Besides the high-level strategy, this study proposes a set of algorithms to execute the mission. The algorithms are centered around three key aspects: states, inputs, and actions. UAVs can assume different states, including searching, mapping, monitoring, and interfering, with each state representing a specific task being performed. The algorithms take into account several inputs, such as the position and velocity of the UAVs, RSSI and frequency, signal-to-noise ratio (SNR) thresholds for frequency detection, transmitter response times (RT), and phase-dependent environment information from other agents. These inputs are crucial for guiding the UAVs' behavior and ensuring efficient operation. The set of possible actions includes adjusting velocity and direction to navigate the search area, sharing information with other UAVs about received signal strength, position, and current tasks, and collaborating with other UAVs to map the communication line and locate signal sources. Collaboration can involve searching for signal sources, mapping, or searching for interference points. Finally, the UAVs can interfere with the signal by disrupting the communication line. By considering these states, inputs, and actions, the algorithms aim to provide a comprehensive framework for managing UAVs in detection, monitoring and interference of the communication line.

#### Searching Phase: Particle Swarm Optimization (PSO)

In the searching phase, the PSO algorithm is employed to guide the UAVs towards locations with stronger signals, potentially identifying the communication lines more efficiently. Each UAV represents a particle in the swarm, updating its position and velocity based on its own best position and the swarm's global best position found so far.

#### Mapping Phase: Grid Search, Multi-UAV Frontier-based Exploration, RL

During the mapping phase, the UAVs collaborate to find the optimal interference point by applying reinforcement learning, enhancing their efficiency and adaptability in the mission. By using RL, the UAVs can systematically explore the search space and evaluate the objective function at each point to find the best point of interference.

In scenarios with unknown area boundaries and multiple UAVs, reinforcement learning can also be employed to improve Frontier-based Exploration. This approach allows the UAVs to efficiently explore the environment by focusing on the frontiers, which are the boundaries between known and unknown areas. By incorporating RL, UAVs can better distribute the exploration task among themselves, reducing redundant efforts and increasing overall efficiency. The state, action, and reward components of the RL problem for multi-agent mapping of communication lines are defined as detailed in section 7.3.

#### Monitoring and Interfering Phase: RL

During the monitoring phase, the mapping agent continues performing the exhaustive grid search algorithm to track any changes in the environment and compare with the previously mapped data.

In the interfering phase, the UAV at the interference point utilizes reinforcement learning to adapt its speed vector, allowing it to approach and maintain its position at the interference point based on the previously obtained information about the communication line's location. By applying RL, the UAV can effectively employ signal disruption techniques while remaining undetected, enhancing its efficiency and adaptability in the mission. The state, action, and reward components of the RL problem for interference point finding are defined as detailed in section 7.2.

### 8.2.3. Model Environment

To evaluate and test the proposed swarm intelligence model, a Python-based simulation environment will be used. The simulation environment will comprise several time-stepped iterations, where each iteration represents a discrete time step. The following Python libraries may be employed to implement various aspects of the simulation:

- **Pygame:** A library that allows for the creation of graphical interfaces, enabling visualization of the UAVs and their environment. [55]

- **Stable Baselines:** A set of high-quality, state-of-the-art reinforcement learning algorithms, which can be used to train the UAVs to perform various tasks within the simulation. [60]
- **TensorFlow Keras:** A high-level neural networks API that can be used to develop and train machine learning models, such as those required for interference point finding and multi-agent mapping. [39]
- **OpenAI Gym:** A toolkit for developing and comparing reinforcement learning algorithms, providing a consistent interface for creating custom environments. [28]

The simulation will run iteratively, with each time step updating the UAVs' positions, actions, and internal states based on their interactions with the environment, their current policies, and the algorithms employed. The simulation will also maintain a record of the environment, including the communication lines, interference points, and any dynamic elements, such as moving transmitters. The model uses a time-stepped approach, where the simulation advances iteratively through discrete time steps. The step size is fixed in terms of time, meaning that each iteration corresponds to a constant time interval.

During each time step, the following events will occur:

1. Update the UAVs' positions based on their current velocities and actions. The spatial grid points, if used, would be determined by the chosen algorithm for updating the UAVs' positions and actions.
2. Update the UAVs' actions according to the algorithms used, such as Particle Swarm Optimization, Grid Search, Frontier-based Exploration, or Reinforcement Learning. These algorithms may involve spatial grid points, depending on their specific implementation, but the primary focus of the simulation is on time steps rather than spatial grids.
3. Update the UAVs' internal states, including the state representations and reward functions as required for reinforcement learning.
4. Update the environment, including any dynamic elements or changes in the signal properties.
5. Record the relevant data, such as UAV positions, actions, and performance metrics, for analysis and evaluation.

This iterative process will continue until a predefined termination condition is met, such as reaching a maximum number of time steps or achieving a specific performance threshold.

Further exploration into more sophisticated simulation environments, like Gazebo – ArduCopter/MAVRos [64], or pybullet [51], is also desired to enhance the realism and complexity of the simulation, facilitating more accurate assessments of the swarm intelligence model's performance in real-world scenarios.

### 8.3. Model Enhancements

Incorporating various enhancements into the model is crucial for improving its performance and adaptability in complex RF environments. The following model enhancements will be considered:

#### Environment Enhancements:

- Increased number of transmitters. (critical)
- Dynamic environment with moving transmitters. (critical)
- Vary number of UAVs. (critical)
- Complex noise patterns and interference from various sources. (desirable)
- Frequency-hopping transmitters to simulate agile communication systems. (desirable)

#### UAV Performance Enhancements:

- Single or multiple points of release for the UAVs, impacting their initial positioning and coordination. (critical)
- Constraints on maximum flying time for the UAVs, requiring efficient task execution and energy management. (desirable)
- Incorporating UAV fall-out in the model and assess task reallocation. (desirable)

- Actively defining and expanding the search area during the mission, adapting to the evolving RF landscape. (desirable)
- Unknown bandwidth of interest, requiring the UAVs to adapt their monitoring capabilities. The UAVs' bandwidth is set at 10 MHz, with adaptability to iterate over 40 bands (covering a total bandwidth of 400 MHz). In a spectrum of interest ranging from 2 to 4 GHz, UAVs listen to channels 1, 2, 3, 4, and 5. (desirable)

#### Strategic Algorithm Enhancements:

- Collaborative techniques like beamforming and phase difference can enhance DoA estimation. (desirable)
- Triangulation can be used to pinpoint the position of a signal source. (desirable)
- FSPL formula can improve distance estimation with the collaboration of at least two UAVs. (desirable)

## 8.4. Evaluation Metrics

The performance of the swarm intelligence model will be evaluated based on a set of carefully defined metrics, and the model will be enhanced to address various environment and UAV performance factors. This will ensure that the model remains effective and adaptable in different scenarios.

1. **Detection accuracy:** Measure the percentage of communication lines detected by the swarm of UAVs compared to the total number of communication lines present in the environment.
2. **Stealthiness:** Assess the ability of the swarm of UAVs to minimize their detection by a  $TR_x$ .
3. **Mapping accuracy:** Evaluate how accurately the swarm of UAVs can map the communication lines in the environment, considering factors like the number of correctly mapped lines, the completeness of the mapped area, and the coverage percentage of the overall area of interest.
4. **Interference effectiveness:** Quantify the success rate of the UAV swarm in disrupting communication lines by evaluating the optimality of the chosen interference point.
5. **Time efficiency:** Measure the average time taken by the swarm of UAVs to complete the tasks of detection, localization, mapping, and interference.
6. **Task complexity:** Incorporate a measure of the complexity or realism of the tasks, accounting for factors such as environmental dynamics, the number of communication lines, and the presence of disruptions.

## 8.5. Experiments

The experiments will be designed to test the model's performance and adaptability under various conditions, as well as to assess the impact of incorporating different enhancements. These experiments will be carried out in a simulation environment, allowing for controlled and repeatable experimentation. Some potential experiments include:

### 8.5.1. Baseline Performance

Evaluate the baseline performance of the swarm intelligence model in a simple RF environment with a limited number of static transmitters and without any environmental, UAV performance, or strategic algorithm enhancements. This will serve as a reference point for measuring the improvements gained from incorporating enhancements into the model.

### 8.5.2. Environmental, UAV Performance, and Strategic Algorithm Enhancements

To further evaluate and enhance the model's adaptability, various environmental, UAV performance, and strategic algorithm enhancements can be considered:

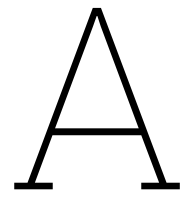
- **Environmental Enhancements:** Test the model's adaptability to different environmental enhancements, such as increased transmitters, dynamic scenarios with moving transmitters, complex noise patterns, and frequency-hopping transmitters. The evaluation metrics will be used to assess the impact of these enhancements on the model's performance.

- **UAV Performance Enhancements:** Evaluate the swarm intelligence model in scenarios with various UAV performance enhancements, including different release points, flying time constraints, adaptive search areas, and unknown bandwidths. This will help assess the model's ability to maintain effective detection, localization, and interference capabilities under different performance constraints.
- **Strategic Algorithm Enhancements:** Examine the impact of incorporating strategic algorithm enhancements, such as collaborative techniques (beamforming, phase difference, triangulation) and the FSPL formula, on the swarm's performance in complex RF environments. The evaluation metrics will be used to quantify the improvements in direction-of-arrival estimation, distance estimation, and overall model performance.

### 8.5.3. Comprehensive Scenario

Test the model's performance and adaptability in a comprehensive scenario, combining various environmental, UAV performance, and strategic algorithm enhancements. This will help evaluate the model's ability to handle complex and dynamic RF environments, as well as assess the overall effectiveness of the swarm intelligence approach.

The results of these experiments will be analyzed and compared using the evaluation metrics to determine the effectiveness of the swarm intelligence model and the benefits of incorporating various enhancements. This will provide valuable insights into the model's performance and adaptability in complex RF environments, informing future research and development efforts in the field of UAV-based CEMA applications.



# Project Planning

## Thesis Project Planning: UAV Swarm Intelligence for Wireless Communication Line Interception

MSc Student: Storm Holman  
 Supervisor TUD: Alexei Sharpanskykh  
 Supervisor NLR: Olivier den Ouden

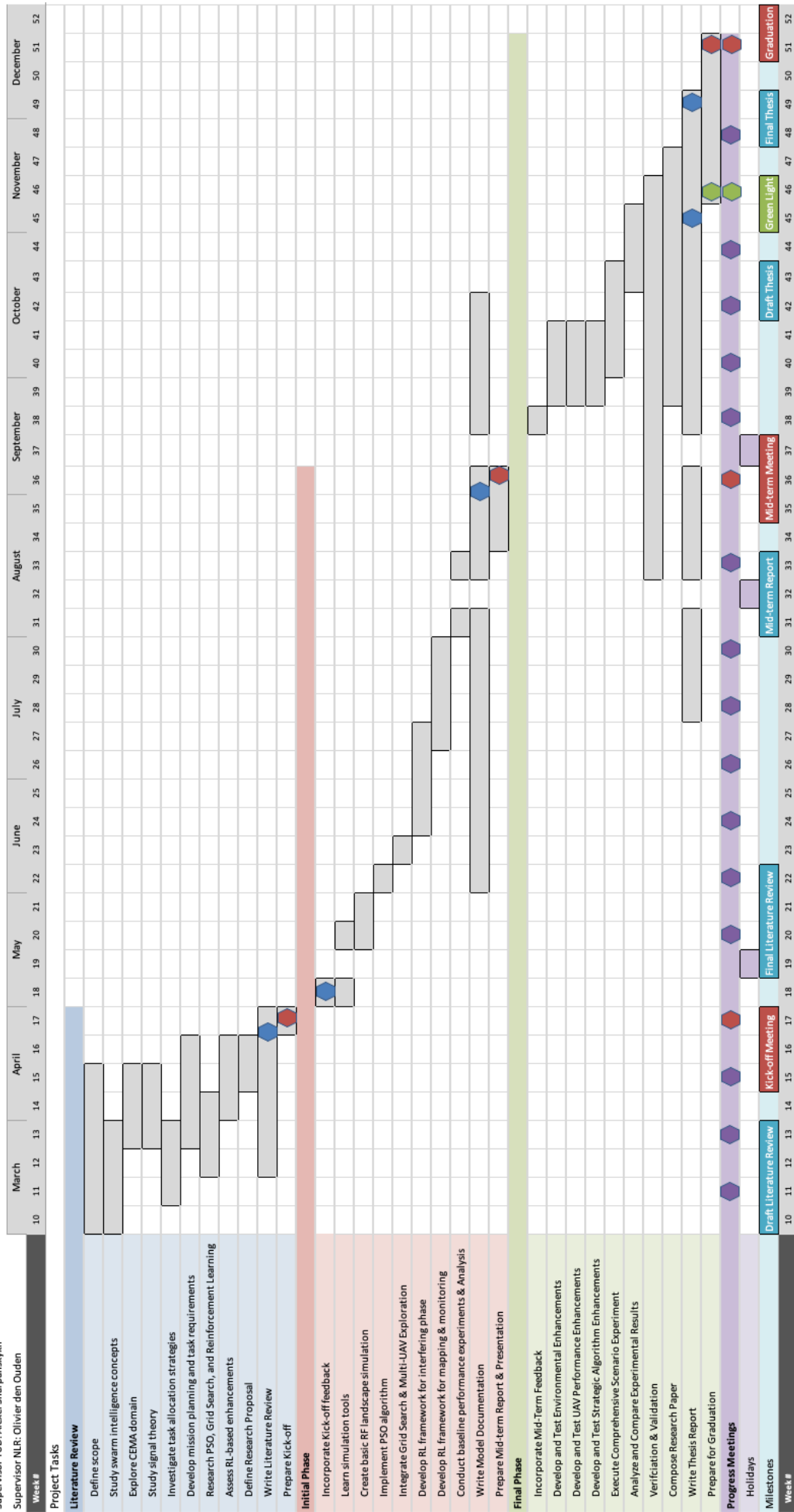


Figure A.1: Gantt chart of the project planning

# References

- [1] James Wlodarczyk (–catslash. *Own work of English Wikipedia user Catslash*. Wikimedia Commons. July 2006. URL: [https://en.wikipedia.org/wiki/Radiation\\_pattern#/media/File:Radiation-patterns-v.png](https://en.wikipedia.org/wiki/Radiation_pattern#/media/File:Radiation-patterns-v.png).
- [2] “A comparison of PSO and Reinforcement Learning for multi-robot obstacle avoidance”. In: 2013, pp. 149–156. ISBN: 9781479904549. DOI: [10.1109/CEC.2013.6557565](https://doi.org/10.1109/CEC.2013.6557565).
- [3] “A Distributed Collaborative Allocation Method of Reconnaissance and Strike Tasks for Heterogeneous UAVs”. In: *Drones* 7 (2 Feb. 2023).
- [4] “A non-dominated sorting particle swarm optimizer for multiobjective optimization”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2723 (2003), pp. 37–48. ISSN: 16113349. DOI: [10.1007/3-540-45105-6\\_4](https://doi.org/10.1007/3-540-45105-6_4).
- [5] “Aerial Swarms: Recent Applications and Challenges”. In: *Current Robotics Reports* 2 (3 Sept. 2021), pp. 309–320. DOI: [10.1007/s43154-021-00063-4](https://doi.org/10.1007/s43154-021-00063-4).
- [6] Susanne Albers. “Better bounds for online scheduling”. In: *SIAM Journal on Computing* 29.2 (1999), pp. 459–473.
- [7] Ross Arnold et al. *Heterogeneous UAV Multi-Role Swarming Behaviors for Search and Rescue; Heterogeneous UAV Multi-Role Swarming Behaviors for Search and Rescue*. 2020.
- [8] Ross D. Arnold, Hiroyuki Yamaguchi, and Toshiyuki Tanaka. “Search and rescue with autonomous flying robots through behavior-based cooperative intelligence”. In: *Journal of International Humanitarian Action* 3 (1 Dec. 2018). ISSN: 2364-3412. DOI: [10.1186/s41018-018-0045-4](https://doi.org/10.1186/s41018-018-0045-4).
- [9] Seyed Mohammad Asghari. *Deep Q-learning (DQN) for Multi-agent Reinforcement Learning We begin with reviewing single-agent and multi-agent reinforcement learning. 1.1 Single-Agent Reinforcement Learning*. 2019.
- [10] ATTIC. *BASIC ELECTRONIC WARFARE*. ATTIC, 2006, p. 230.
- [11] David Baldazo, Juan Parras, and Santiago Zazo. “Decentralized Multi-Agent Deep Reinforcement Learning in Swarms of Drones for Flood Monitoring”. In: *27th European Signal Processing Conference (EUSIPCO)978-9-0827-9703-9/19/©2019 IEEE* (2019).
- [12] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. online. Accessed: 2023-04-03. New York: Oxford University Press, 1999. DOI: [10.1093/oso/9780195131581.001.0001](https://doi.org/10.1093/oso/9780195131581.001.0001). URL: <https://doi.org/10.1093/oso/9780195131581.001.0001>.
- [13] Craig Boutilier. “Planning, learning and coordination in multiagent decision processes”. In: *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*. Morgan Kaufmann Publishers Inc. 1996, pp. 195–210.
- [14] Manuele Brambilla et al. “Swarm robotics: A review from the swarm engineering perspective”. In: *Swarm Intelligence* 7 (1 Mar. 2013), pp. 1–41. ISSN: 19353820. DOI: [10.1007/s11721-012-0075-2](https://doi.org/10.1007/s11721-012-0075-2).
- [15] Thameur Chaari et al. “Scheduling under uncertainty: Survey and research directions”. In: *International Conference on Advanced Logistics and Transport, ICALT*. IEEE. 2014, pp. 229–234.
- [16] Yu Chen et al. “Multi-UAV Autonomous Path Planning in Reconnaissance Missions Considering Incomplete Information: A Reinforcement Learning Method”. In: *Drones* 7 (1 Dec. 2022), p. 10. ISSN: 2504446X. DOI: [10.3390/drones7010010](https://doi.org/10.3390/drones7010010).
- [17] “Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming”. In: vol. 50. Elsevier B.V., July 2017, pp. 8011–8018.
- [18] “Cooperative Simultaneous Tracking and Jamming for Disabling a Rogue Drone”. In: (Feb. 2023). DOI: [10.1109/IR0S45743.2020.9340835](https://doi.org/10.1109/IR0S45743.2020.9340835). URL: <http://arxiv.org/abs/2302.07575%20http://dx.doi.org/10.1109/IR0S45743.2020.9340835>.

- [19] “Cooperative task allocation for unmanned vehicles with communication delays and conflict resolution”. In: vol. 13. American Institute of Aeronautics and Astronautics Inc., 2016, pp. 67–79. DOI: [10.2514/1.I010218](https://doi.org/10.2514/1.I010218).
- [20] “Decentralized Multi-Agent Pursuit Using Deep Reinforcement Learning”. In: *IEEE Robotics and Automation Letters* 6 (3 July 2021), pp. 4552–4559.
- [21] Michael L Dertouzos and Aloysius K Mok. “Multiprocessor online scheduling of hard-real-time tasks”. In: *IEEE Transactions on Software Engineering* 15.12 (1989), pp. 1497–1506.
- [22] Gianni Di Caro. *An Introduction to Swarm Intelligence Issues*. [http://staff.washington.edu/paymana/swarm/dicaro\\_lecture1.pdf](http://staff.washington.edu/paymana/swarm/dicaro_lecture1.pdf). Lugano, Switzerland: IDSIA, USI/SUPSI.
- [23] Matthew C Edwards, Renate Meyer, and Nelson Christensen. “Bayesian semiparametric power spectral density estimation with applications in gravitational wave data analysis”. In: (2015).
- [24] Ahmed Elmogy. *Market-based Framework for Mobile Surveillance Systems*. 2010.
- [25] European Space Agency. *Satellite frequency bands*. [https://www.esa.int/Applications/Telecommunications\\_Integrated\\_Applications/Satellite\\_frequency\\_bands](https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Satellite_frequency_bands). Accessed: 2023-04-15. n.d.
- [26] Everything RF. *ZigBee Frequency Bands*. Available at: <https://www.everythingrf.com/community/zigbee-frequency-bands>. 2021. URL: <https://www.everythingrf.com/community/zigbee-frequency-bands>.
- [27] Carmelo Di Franco and Giorgio Buttazzo. “Energy-aware coverage path planning of UAVs”. In: Institute of Electrical and Electronics Engineers Inc., May 2015, pp. 111–117. ISBN: 9781467369909. DOI: [10.1109/ICARSC.2015.17](https://doi.org/10.1109/ICARSC.2015.17).
- [28] *Gym: A toolkit for developing and comparing reinforcement learning algorithms*. <https://www.gymnasium.dev>. Accessed: 2023-04-18.
- [29] Il Kyu Ha. “A probabilistic target search algorithm based on hierarchical collaboration for improving rapidity of drones”. In: *Sensors (Switzerland)* 18 (8 Aug. 2018). ISSN: 14248220. DOI: [10.3390/s18082535](https://doi.org/10.3390/s18082535).
- [30] Abbas Haider. *5G vs 4G frequency spectrum comparison*. <https://electronics360.globalspan.com/article/15589/5g-vs-4g-frequency-spectrum-comparison>. Aug. 2020.
- [31] D.L. Herrick, P.K. Lee, and L.L. Ledlow. “Correlated frequency hopping-an improved approach to HF spread spectrum communications”. In: *Proceedings of the 1996 Tactical Communications Conference. Ensuring Joint Force Superiority in the Information Age*. 1996, pp. 319–324. DOI: [10.1109/TCC.1996.561099](https://doi.org/10.1109/TCC.1996.561099).
- [32] Hitoshi Iba. *Agent-Based Modeling and Simulation with Swarm*. Chapman & Hall, 2013.
- [33] National Instruments. *Frequency Domain vs. Time Domain*. [https://www.ni.com/docs/en-US/bundle/labview/page/lvanlscconcepts/freq\\_time\\_domain\\_diffs.html](https://www.ni.com/docs/en-US/bundle/labview/page/lvanlscconcepts/freq_time_domain_diffs.html). Accessed: 2023-04-23. 2023.
- [34] A Jamshidpey et al. *Centralization vs. decentralization in multi-robot coverage: Ground robots under UAV supervision*. 2021.
- [35] Lucas Hop June. *Efficient Swarm Robotic Persistent Surveillance by use of Stigmergy*. 2021.
- [36] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey”. In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 237–285.
- [37] Myron Kayton. *Avionics Navigation Systems, Second Edition*. except p. 187-189, §5.4.1, §5.4.2, §5.5.3, p. 213 (from ‘Signal Modulation’) - 218 (up to §5.5.6), p. 228 (from ‘Uncorrected ...’) - 229, §5.5.8, §5.6, p. 271-274, p. 277-278. John Wiley & Sons, Inc., 1997. Chap. 5.
- [38] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: *Proceedings of IEEE International Conference on Neural Networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.
- [39] *Keras: The Python Deep Learning Library*. <https://keras.io/about/>. Accessed: 2023-04-18.
- [40] David King and Gilbert Peterson. “The Emergence of Division of Labor in Multi-Agent Systems”. In: vol. 2019-June. IEEE Computer Society, June 2019, pp. 107–116. ISBN: 9781728127316. DOI: [10.1109/SASO.2019.00022](https://doi.org/10.1109/SASO.2019.00022).
- [41] Albert Krohn et al. “The implementation of non-coherent cooperative transmission for wireless sensor networks”. In: *Telecommunication Systems* 38.1-2 (2008), pp. 137–148.
- [42] Wenxin Le et al. “Coverage Path Planning Based on the Optimization Strategy of Multiple Solar Powered Unmanned Aerial Vehicles”. In: *Drones* 6 (8 Aug. 2022). ISSN: 2504446X. DOI: [10.3390/drones6080203](https://doi.org/10.3390/drones6080203).



- [43] Liang Lu, Carlos Redondo, and Pascual Campoy. "Optimal Frontier-Based Autonomous Exploration in Unconstructed Environment Using RGB-D Sensor". In: *Sensors* 20.22 (2020), p. 6507. DOI: [10.3390/s20226507](https://doi.org/10.3390/s20226507).
- [44] Cisco Meraki. *Signal-to-Noise Ratio (SNR) and Wireless Signal Strength*. [https://documentation.meraki.com/MR/Wi-Fi\\_Basics\\_and\\_Best\\_Practices/Signal-to-Noise\\_Ratio\\_\(SNR\)\\_and\\_Wireless\\_Signal\\_Strength](https://documentation.meraki.com/MR/Wi-Fi_Basics_and_Best_Practices/Signal-to-Noise_Ratio_(SNR)_and_Wireless_Signal_Strength). Accessed: 2023-04-25. 2021.
- [45] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv preprint arXiv:1312.5602* (2013).
- [46] Jonathan Mugan. *RLLib for Deep Hierarchical Multi-Agent Reinforcement Learning*. <https://deu.mbra.com/2022/08/rllib-for-deep-hierarchical-multiagent-reinforcement-learning/>. Aug. 2022.
- [47] "Multi-UAV coverage path planning based on hexagonal grid decomposition in maritime search and rescue". In: *Mathematics* 10 (1 Jan. 2022). ISSN: 22277390. DOI: [10.3390/math10010083](https://doi.org/10.3390/math10010083).
- [48] Daisuke Nakanishi, Gurpreet Singh, and Kshitij Chandna. "Developing Autonomous Drone Swarms with Multi-Agent Reinforcement Learning for Scalable Post-Disaster Damage Assessment". CUSP-GX-5006: Urban Science Intensive II (Fall 2021). 2021.
- [49] Daisuke Nakanishi, Gurpreet Singh, and Kshitij Chandna. *Developing Autonomous Drone Swarms with Multi-Agent Reinforcement Learning for Scalable Post-Disaster Damage Assessment Team Members*. 2021.
- [50] James O. Neel and Jeffrey H. Reed. "Performance of Distributed Dynamic Frequency Selection Schemes for Interference Reducing Networks". In: *MILCOM 2006 - 2006 IEEE Military Communications conference*. 2006, pp. 1–7. DOI: [10.1109/MILCOM.2006.302016](https://doi.org/10.1109/MILCOM.2006.302016).
- [51] Sean Nissim, Eric Hovland, and Timothy D. Barfoot. *gym-pybullet-drones*. <https://github.com/utiasDSL/gym-pybullet-drones>. 2021.
- [52] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals and Systems*. Upper Saddle River, NJ: Prentice Hall, 1997.
- [53] O.F.C. den Ouden et al. *CEMA Delivery systems for detecting, listening, and interfering with data communication lines*. Tech. rep. NLR-CR-2023-066. CUSTOMER: Ministry of Defence. Royal NLR - Netherlands Aerospace Centre, Mar. 2023.
- [54] Dennis V Perepelitsa. "Johnson Noise and Shot Noise". In: *MIT Department of Physics* (Nov. 2006).
- [55] *Pygame*. <https://www.pygame.org/news>. Accessed: 2023-04-18.
- [56] Remote Flyer. *How Fast Can a Drone Fly? With Examples*. Available at: <https://www.remoteflyer.com/how-fast-can-a-drone-fly-with-examples/>. 2021. URL: <https://www.remoteflyer.com/how-fast-can-a-drone-fly-with-examples/>.
- [57] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).
- [58] SimpleWiFi. *How it Works: WiFi Antenna & Extender*. Long range WiFi solutions: How to increase your coverage and get the most out of a connection. n.d. URL: <https://simplewifi.com/pages/how-it-works>.
- [59] *SINCGARS*. <https://www.hfunderground.com/wiki/index.php/SINCGARS>. Accessed: 2023-04-18. n.d.
- [60] *Stable Baselines 3*. <https://stable-baselines3.readthedocs.io/en/master/>. Accessed: 2023-04-18.
- [61] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT Press, 2018.
- [62] R. Swol. *Deinterleaving of Radar Pulses and PRI Modulation Type Recognition*. Tech. rep. Royal NLR, 2020.
- [63] S Tatar. *Flocking Algorithm for Formation Control of Non-Holonomic Networked Euler-Lagrange Multi-Robot Systems TOWARDS SWARM INTELLIGENT NETWORKED MOBILE MULTI-ROBOT SYSTEMS*. 2020.
- [64] The MAVLink Development Team. *MAVRos: MAVLink integration with ROS*. <https://github.com/mavlink/mavros>. accessed 2023.
- [65] "UAV-based multi-sensor data fusion and machine learning algorithm for yield prediction in wheat". In: *Precision Agriculture* 24 (1 Feb. 2023), pp. 187–212. ISSN: 15731618. DOI: [10.1007/s11119-022-09938-8](https://doi.org/10.1007/s11119-022-09938-8).

- [66] V.S. Varadharajan and Giovanni Beltrame. “Get Together! Multi-robot Systems: Bio-Inspired Concepts and Deployment Challenges”. In: *Foundations of Robotics*. Ed. by Diluka Herath and David St-Onge. Singapore: Springer, 2022. DOI: [10.1007/978-981-19-1983-1\\_11](https://doi.org/10.1007/978-981-19-1983-1_11). URL: [https://doi.org/10.1007/978-981-19-1983-1\\_11](https://doi.org/10.1007/978-981-19-1983-1_11).
- [67] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine Learning* 8.3-4 (1992), pp. 229–256.
- [68] Dan Xu, Eric Jung, and Xin Liu. “Optimal Bandwidth Selection in Multi-Channel Cognitive Radio Networks: How Much is Too Much?” In: *2008 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*. 2008, pp. 1–11. DOI: [10.1109/DYSPAN.2008.45](https://doi.org/10.1109/DYSPAN.2008.45).
- [69] Mi Yang et al. “A distributed task reassignment method in dynamic environment for multi-UAV system”. In: *Applied Intelligence* 52 (2 Jan. 2022), pp. 1582–1601. ISSN: 15737497. DOI: [10.1007/s10489-021-02502-3](https://doi.org/10.1007/s10489-021-02502-3).
- [70] Marco Zennaro and Ermanno Pietrosemoli. *Intro to Radio Propagation, Antennas and Link Budget*. Training materials for wireless trainers.
- [71] H. Zhu. *Free-space Path Loss*. <https://encyclopedia.pub/entry/31362>. Accessed: 2023-04-15.
- [72] Maciej Zurad et al. *Target Tracking Optimization of UAV Swarms based on Dual-Pheromone Clustering*. 2017. ISBN: 9781538622018.

# III

Supporting Work

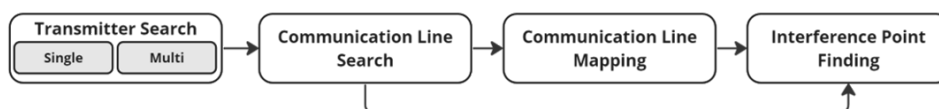


## Problem Setup

This appendix elaborates on the problem setup for the UAV swarm operations within Cyber Electromagnetic Activities (CEMA). It is structured into three sections: [section 1.1](#) breaks down the mission into its component tasks; Research Questions in [section 1.2](#); and [section 1.3](#) discusses the study's boundaries and the underlying assumptions.

### 1.1. Mission Tasks

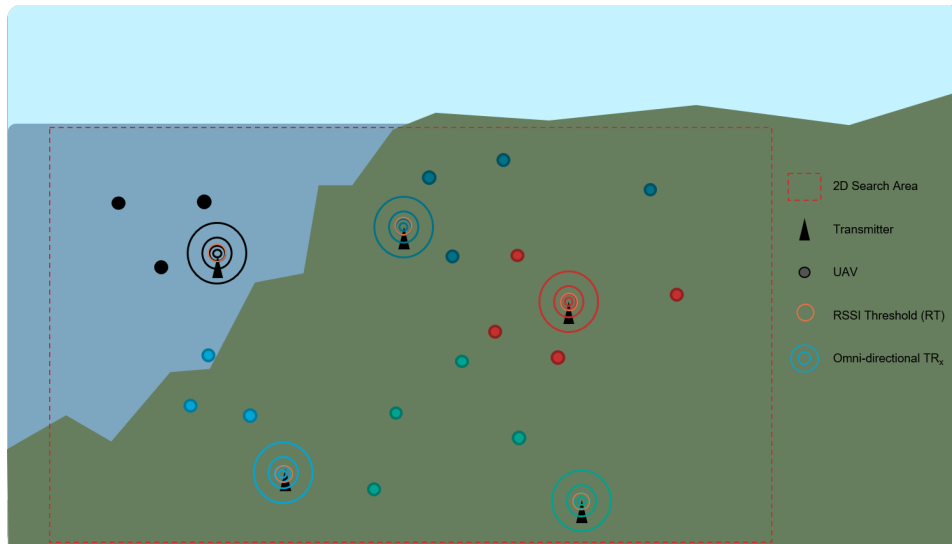
This section outlines the mission's structured approach, comprising four main tasks: Transmitter Search, Communication Line Search, Communication Line Mapping, and Interference Point Finding. [Figure 1.1](#) shows each task, illustrating their order. A high-level mission flow governs the transition between these tasks, ensuring a methodical progression of the operation.



**Figure 1.1:** Visualisation of Mission Flow for UAV swarm in CEMA

#### 1.1.1. Transmitter Search

[Figure 1.2](#) presents a schematic of the operational area for the Transmitter Search task. Agents coordinate to detect multiple omni-directional transmitters, each sending out unique signals. Effective search algorithms are employed to locate an omni-directional transmitter on a specific frequency. Coordination algorithms are also pivotal, enabling the agents to manage the search's multi transmitter aspect efficiently. As agents are restricted to one frequency band at a time, this multi transmitter strategy becomes essential. The RT, indicated by an orange circle in [Figure 1.2](#), marks a boundary that helps agents keep a safe operational distance from transmitters. Detection of the RT signifies the successful location of a transmitter.

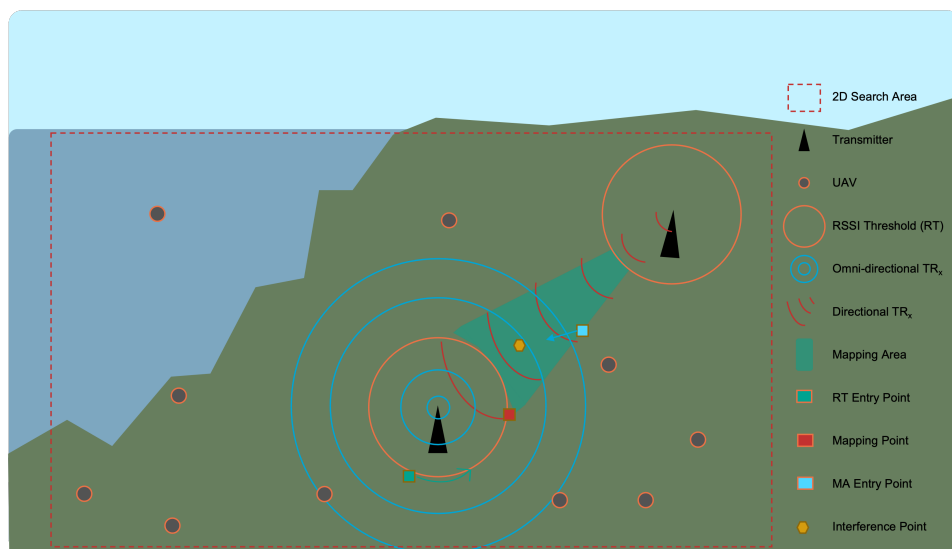


**Figure 1.2:** Schematic Representation - Global Setup

To select optimal velocity control algorithms, the search algorithms will initially be implemented in a single target environment as described by [Figure 1.3](#). Following this, the coordination required to resolve the multi transmitter scenario will be developed in [Figure 1.2](#)

### 1.1.2. Communication Line Search

In the progression of mission tasks, attention is directed towards the interaction between omni-directional and directional transmitters, as depicted in [Figure 1.3](#). A communication line is established between an omni-directional transmitter/receiver ( $TR_o$ ) and a directional transmitter/receiver ( $TR_d$ ). The rationale for employing both types lies in their distinct operational characteristics: omni-directional signals, known for their extensive coverage, are typically easier to detect, whereas directional signals, common in tactical communication systems, are advantageous for detailed mapping. This set-up requested by the NLR, involves targeted transmitters (which may be smaller or mobile) in communication with a more stationary omni-directional transmitter.



**Figure 1.3:** Schematic Representation - Base Setup

Upon an agent's arrival at the RT Entry Point (highlighted as an orange circle), the Communication Line Search task begins. The communication line is identified once the mapping point (shown as a red square in the figure) is established. Notably, this mapping point corresponds to the intersection between the RT and the periphery of the directional signal.

### 1.1.3. Communication Line Mapping

After the mapping point is identified, agents proceed to the Communication Line Mapping task. In this task, the communication line is mapped, represented as a green area in [Figure 1.1](#). This mapping serves dual purposes: it not only lays the groundwork for Interference Point Finding, but it also enhances situational awareness by providing detailed visual representation of the communication infrastructure, as requested by the NLR. This intelligence is crucial for effective strategic planning and execution of interference activities.

### 1.1.4. Interference Point Finding

Following the completion of mapping, the mission progresses to the Interference Point Finding task. In this phase, an agent navigates through the Mapping Arrival (MA) Entry Point, depicted as a blue square in [Figure 1.1](#), towards a predetermined interference point, marked by a yellow hexagon. The selection of this point is strategic, aimed at maximizing signal disruption. The ideal interference position is where the differential sum of the Received Signal Strength Indicator (RSS) levels from two signals is minimized, essentially where  $\min(RSS_1 - RSS_2)$  is achieved. This strategy ensures that the interference point chosen is the most effective for disrupting communication signals in both directions. By positioning at this specific location, the UAVs can exert maximum interference, ensuring optimal disruption of communication lines.

Organizing the mission into these tasks not only ensures a systematic approach but also promotes convergence to a solution, enhancing the likelihood of meeting the mission objective of finding, mapping and interfering with targeted communication lines.

## 1.2. Research Questions

**Main Question:** Which methods are most effective in developing a swarm intelligence model for the mission tasks in a specific CEMA environment?

1. How is the UAV swarm mission in CEMA defined, and what criteria determine its success?
2. What characteristics and parameters define a base and a global CEMA environment setup suitable for the application of a swarm intelligence model, and what challenges arise within such environments?
3. What methodologies and specific algorithms can the model most effectively adopt for executing each mission task?
  - (a) For Single Target Search: Which techniques can most effectively locate a single communication line, evaluated in terms of time to detect and success rate?
  - (b) For Multi Target Search: How can the swarm intelligence model adapt to environments with multiple communication lines to maximize both the speed of detection and the amount of transmitters detected?
  - (c) In Communication Line Mapping: What approaches can ensure the most precise and complete mapping of detected communication lines, measured by coverage and precision?
  - (d) For Interference Point Finding: What methodologies can be employed to locate interference points both precisely and efficiently, assessed by the time to determine the location and localization error?

## 1.3. Scope and Assumptions

The scope of this study is confined to a two-dimensional (2D) search area, focusing on the capabilities of UAVs to detect and manage communication lines within this restricted plane.

Assumptions integral to the model include:

- **Environment:**

- E1** Phenomena like refraction, reflection, and environmental interference are not considered in the model.
- E2** Atmospheric and antenna-induced noise are disregarded, making signal-to-noise ratio (SNR) thresholds irrelevant.
- E3** Environmental factors such as wind and temperature are assumed to have no impact on UAV performance.

- **UAV Performance:**

- U1** UAVs are modeled as simple entities, capable of movement only in the x and y directions, with a maximum velocity of 30 m/s.
- U2** Homogeneity in sensing and communication capabilities across all UAVs is assumed.
- U3** Agents can only listen to one frequency band at a time.
- U4** A frequency band is assumed to exactly match the signal of interest.
- U5** Onboard signal processing and transmission methods are not considered in this model.
- U6** Received Signal Strength Thresholds (RT) are set at -20 mdB to not further engage with a transmitter.
- U7** Only communication between a single omni-directional and a single directional signal is taken into account.

- **Simulation:**

- S1** Computation and data storage are considered available but minimal, drawing parallels to real-world small processors and memory cards onboard UAVs.
- S2** Communication between UAVs is instantaneous, without any time delay.
- S3** A sequential time-step model is used to study agent decisions, abstracting from real-world scenarios where decisions might be made asynchronously.
- S4** Communication between transmitters is represented by a continuous signal impulse.



# 2

## Model Elaboration

This appendix delves into the Agent-Based Model (ABM) for the UAV swarm operations, detailed in [section 2.1](#). Model architecture and simulation processes are elaborated in the Model Implementation in [section 2.2](#) leveraging Python and the NLR - EW Toolbox for efficiency and analysis. The software infrastructure and computational tools used in the model are outlined in the Technical Setup [section 2.3](#).

### 2.1. Agent-Based Model

This section is an explanation of the Agent-Based Model (ABM). It encapsulates environmental factors, agent attributes, behavioral and cognitive traits, as well as inter-agent interactions, to facilitate a comprehensive simulation of UAV swarm behavior.

#### Environment

The agent-based model's environment incorporates radio-frequency components such as transmitters, receivers, and reflectors. The environment operates in a 2D space with a known search area of  $x \times y$  km. Two types of transmitters are considered: one omni-directional (position  $(x_{TRO}, y_{TRO})$ ) and one directional (position  $(x_{TRD}, y_{TRD})$ ), both static. The environment is deterministic, meaning every action  $a \in A$  leads to a predictable outcome. It remains static unless acted upon by agents and is fully observable, allowing agents to acquire a complete state  $s$ .

#### Agent Properties

Agents in the model function as state machines and represent UAVs in a swarm of  $N_{UAV}$ . The UAVs in the swarm operate reactively, responding immediately to changes in environmental stimuli. Observations  $O$  encompass position, velocity, received signal strength (RSS), frequency, RSS Threshold of transmitters ensuring safe distance for the agents. Defined formally as  $O = \{o_{\text{position}}, o_{\text{velocity}}, o_{\text{RSS}}, \dots\}$ . Agents can perform actions  $A$ , which include velocity adjustment, information sharing, and task-based collaboration. These actions are denoted as  $A = \{a_{\text{adjust velocity}}, a_{\text{share information}}, \dots\}$ . Internal states  $I$  such as searching, mapping, and Interference Point Finding are regulated by perception function  $P: O \rightarrow I$ . Agents are constrained by a maximum velocity of 30 m/s and operate with bounded rationality. The decision-making process for these reactive agents is straightforward: upon receiving new observations, a UAV will update its velocity if the received signal strength indicates proximity to the signal threshold during a search operation. This ensures that the UAVs' movements are continually adjusted to optimize the search pattern.

#### Inter-Agent Interactions

Agents communicate findings, influencing their actions  $A$  based on observed data  $O$  and internal states  $I$ . In cooperative contexts, interaction protocols facilitate collective goal achievement. Communication enables task or state switching, and collaboration is possible for activities like signal source gradient determination.

## Dynamics & Time Modeling

The simulation employs a time-stepped approach, conducting a series of iterative steps to continuously update UAV positions, actions, and internal states. The time-stepped simulation operates in a sequential manner as outlined:

1. UAV positions are updated based on current velocities and actions taken, in compliance with the agent's state-machine structure and the deterministic nature of the environment.
2. UAV actions are adjusted according to deployed algorithms, affecting the set of actions  $A$  based on observations  $O$  and internal states  $I$ .
3. Internal states of UAVs, denoted by  $I$ , are updated.
4. The swarm state is updated based on the actions and attributes of the agents, influencing the set of observations  $O$  available to each agent.
5. Relevant data including UAV positions, actions, and performance metrics are logged for subsequent analysis and evaluation.

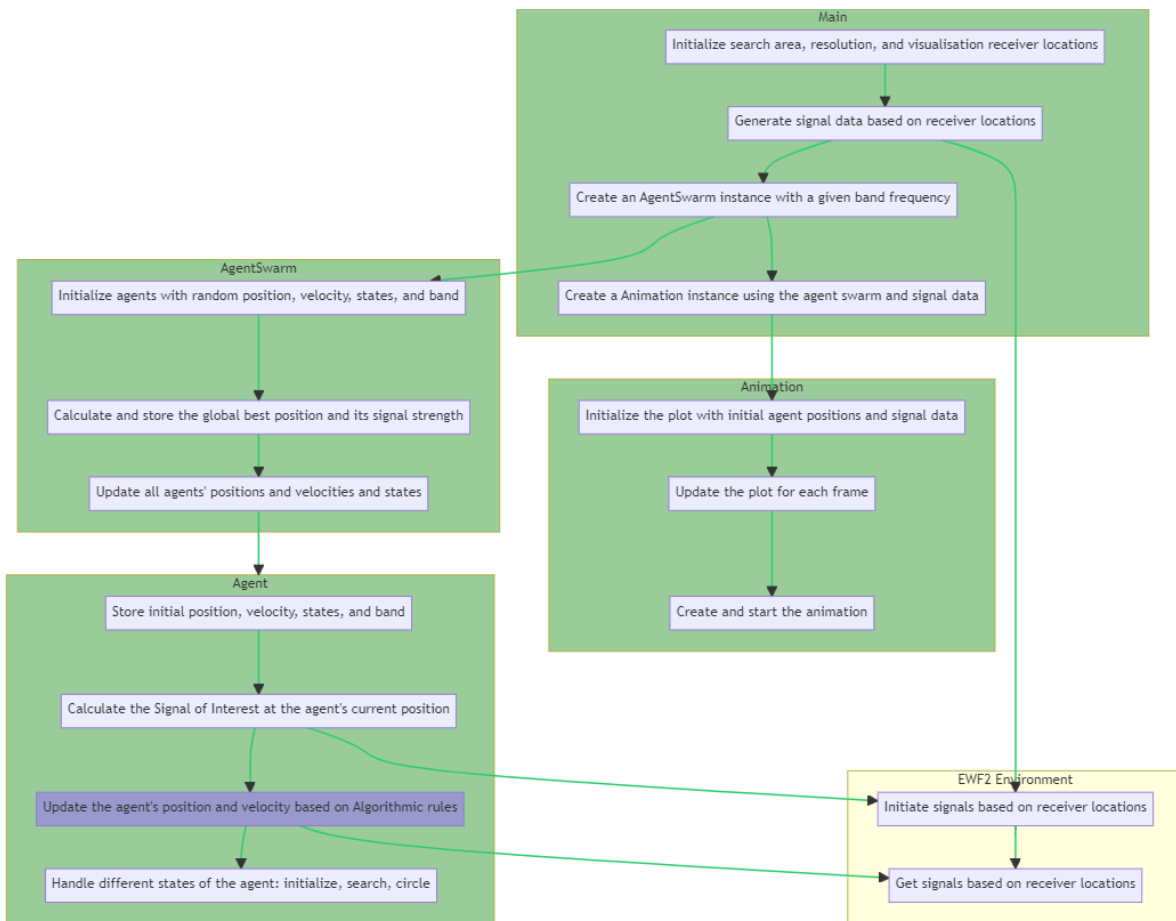
The simulation iterates through these steps until a predefined termination condition is met, such as reaching a maximum number of time steps or achieving a specified performance threshold.

## Simulation Objectives and Challenges

The primary objectives of employing this ABM are to identify algorithms and their parameters affecting UAV swarm operations. Key performance indicators (KPIs) like mission time, swarm efficiency, and trajectory coverage are used to evaluate algorithmic performance. Challenges may include evaluating and verifying known patterns, identifying new patterns, and interpreting causality within the simulation.

## 2.2. Model Implementation

This section outlines the model's algorithmic structure, as depicted in [Figure 2.1](#) and detailed in [Algorithm 1](#). Central to the model are two classes: the Agent class, directing individual UAV behaviors, and the AgentSwarm class, coordinating the collective actions of the swarm. Key operations include the initialization of agents, updating their positions, and retrieving signal strengths from the NLR - EW environment, detailed in [subsection 2.3.1](#). Visual representations, crucial for analysis, are handled by the Animation instance, which utilizes signal data alongside agent swarm movements. The algorithmic block, 'Update the agents' position and velocity based on algorithmic rules', forms the essence of the swarm's intelligence and is the primary focus of this thesis.



**Figure 2.1:** High-level Model Architecture Flow

## 2.3. Technical Setup

This section outlines the technical components essential for implementing and running the agent-based model. Components are categorized into specialized software, programming libraries, and hardware infrastructure.

### 2.3.1. Specialized Software: NLR - EW Toolbox

NLR - EW Toolbox is specialized for controlling and manipulating radio frequency elements such as transmitters, receivers, reflectors, and countermeasures. User-adjustable attributes include position, velocity, attitude, and signature lobe features. Signal processing parameters like bandwidth, frequency, power, and pulse width can be manipulated.

### 2.3.2. Python Libraries

Several Python libraries are employed in the simulation, each selected for specific advantages:

- **SciPy/Numpy:** Responsible for numerical operations and data manipulation, contributing to algorithmic performance.
- **Matplotlib:** Utilized for data visualization, aiding in result interpretation.
- **Stable Baselines:** Provides reinforcement learning algorithms for training the UAVs in the simulation.
- **OpenAI Gym:** Offers a standard interface for custom reinforcement learning environment creation.

### 2.3.3. Hardware Infrastructure

The model is implemented in a local Python virtual environment on two computing systems to cater to varying computational demands. For regular simulation tasks, a high-performance personal computer is used, equipped with an Intel Core i5-8400H CPU at 2.50GHz, 16 GB of installed RAM, and a 64-bit operating system, which supports efficient data management and algorithmic execution.

For heavy computational tasks, such as running **MCTS** and reinforcement learning training for algorithms like QL and PPO, a more robust system is utilized, an Alienware Aurora R13. This system boasts a 12th Gen Intel(R) Core(TM) i9-12900KF processor with 24 CPUs at 3.2GHz and 131072MB of RAM, providing substantial computational power and memory capacity to handle intensive simulations and data processing with ease.

# 3

## Elaboration on Methods

This appendix offers a detailed explanation of the methods primarily focused on controlling velocity updates. It starts with detailed insights into the Transmitter Search methods in [section 3.1](#). The discussion then progresses to the Communication Line Search & Other Specialized Algorithms in [section 3.2](#), highlighting the techniques used for establishing and identifying communication lines. Subsequently, [section 3.3](#) elucidates the Communication Line Mapping process, focusing on the strategies used for accurately charting these lines. Lastly, the appendix explores the Interference Point Finding methods in [section 3.4](#), detailing the approaches for locating optimal interference points within communication lines.

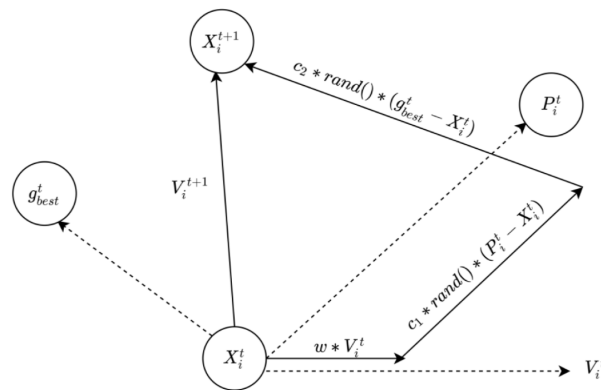
### 3.1. Transmitter Search (Single)

In the Single Transmitter Search, two algorithms are evaluated and deployed in the Base Setup: Particle Swarm Optimisation (PSO) and Pair-based Gradient Search.

#### 3.1.1. Particle Swarm Optimisation Search

##### Algorithm Description

- **Overview:** Particle Swarm Optimisation (PSO) [\[7\]](#) is an optimization technique inspired by the collective behavior observed in bird flocks and fish schools. When applied to robotics, each robot, or specifically UAV, acts as a particle in the swarm. These UAVs, beyond being abstract particles in an optimization algorithm, have physical constraints and operate within the real environment. The method has been employed in guiding UAVs toward locations with stronger Received Signal Strength (RSS), optimizing their positions based on this signal. Such an application enables efficient search operations in UAV swarm systems, as demonstrated in prior work [\[4\]](#).
- **Symbols and Definitions:**
  - **Agent Properties:** [Figure 3.1](#) visualizes the key components each UAV uses to determine its new velocity vector. The UAV updates its trajectory by adding up its current velocity ( $v_i^{(t)}$ ) with cognitive ( $c_1$ ) and social ( $c_2$ ) components, both influenced by random factors ( $r_1$  and  $r_2$ ). The cognitive component involves the UAV's personal best position ( $p_i^{(t)}$ ), while the social component relates to the swarm's global best position ( $g^{(t)}$ ). This update mechanism is governed by the velocity update equation  $v_i^{(t+1)} = wv_i^{(t)} + c_1r_1(p_i^{(t)} - x_i^{(t)}) + c_2r_2(g^{(t)} - x_i^{(t)})$ . Following the velocity update, the UAV's position is then updated according to the equation  $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$ , integrating the new velocity into the UAV's current location.



**Figure 3.1:** Visualization of the key PSO agent components: current velocity, social influence, cognitive drive, and random factors  $r_1$  and  $r_2$ .

- **Observations  $O$ :** Observations encompass the Received Signal Strength  $RSS$  detected by a UAV at its current coordinates  $x, y_i(t)$  and the global best position known within the swarm. These data points are critical for guiding the UAV's trajectory adjustments.
- **Actions  $A$ :** The primary action undertaken by a UAV is the iterative adjustment of its velocity vector. This adjustment is influenced by both its personal best and the global best positions.
- **Internal States  $I$ :** These include the UAV's current velocity, its personal best position (determined by the strongest  $RSS_d$  it has encountered), and its knowledge of the swarm's global best position. The UAV continuously measures and compares these positions, updating them as necessary during its search.
- **Inter-Agent Interactions:** UAVs constantly communicate, sharing their most recent observations. Through this, they collectively guide the swarm towards areas with the strongest signal strengths.
- **Procedure:** The PSO's central mechanism involves iterative updates to a UAV's velocity and position, aligning with the principles of the algorithm. A detailed breakdown is presented in pseudocode, as shown in [Algorithm 2](#).

### Parameters and Tuning

- **Key Parameters:** The success of the PSO algorithm hinges on understanding and correctly setting its parameters. Important parameters include:
  - **Weights for cognitive and social components ( $c_1, c_2$ ):** These weights determine how much emphasis an agent places on its own experiences versus the experiences of its neighbors.
  - **Random factors  $r_1$  and  $r_2$ :** These introduce stochasticity into the adjustment of the velocity, ensuring diversity in exploration.
  - **Agent Initialization:** The starting positions and velocities of UAVs can influence the search's trajectory and speed. Different initialization methods can lead to varied results.
- **Tuning:** Fine-tuning these parameters is crucial for task-specific and environmental optimization. The PSO algorithm's parameters, including an inertia weight of 0.8, a cognitive coefficient of 0.1, and a social coefficient of 0.05, are chosen based on established swarm intelligence behaviors [\[14\]](#) to ensure effective convergence.

### Hypothesis for Testing and Prospects

- **Hypothesis:** It is predicted that the PSO method, when implemented on UAVs, will perform efficiently in guiding them towards optimized positions. However, its performance is expected to

be sensitive to parameter tuning, and there might be a risk of the algorithm getting trapped due to specific agent initialization settings.

- **Advantages:**

- Enables cooperative and distributed search, enhancing the swarm's ability to cover vast areas effectively.
- Provides robustness against individual UAV failures due to its inherent swarm nature.
- Easily scalable, allowing for the incorporation of large numbers of robots without significant changes to the algorithm.

- **Limitations:**

- Demands a consistent communication range among robots, making it challenging in environments where communication might be unreliable.
- Requires careful tuning to achieve the right balance between exploration (searching new areas) and exploitation (optimizing based on known data).
- Performance can be heavily dependent on the method and settings used for agent initialization.

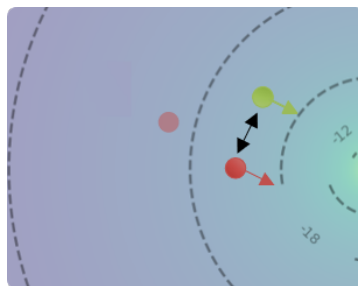
### 3.1.2. Pair-based Gradient Search

#### Algorithm Description

- **Overview:** The pair-based gradient search builds upon traditional gradient search algorithms. Ghosh et al. previously developed an algorithm called Gradient Direction Turn Switching Strategy for Source Localization [10]. In the current study, the approach emphasizes agents operating in pairs, which minimizes the reliance on historical data. Triangulation is essential in this approach, as sample points should not be colinear to avoid inaccurate gradient computations. One agent moves to an offset position at 90 degrees, aiding in this triangulation. The pairing problem solution employs a Gale-Shapley Algorithm, designed to achieve a stable match between two sets of elements, with inherent biases based on the order of proposing, which might introduce suboptimal results [5]. However, gradient determination can be challenging due to the non-linear behavior of signal propagation.

- **Symbols and Definitions:**

- **Agent Properties:** Agents in this model are specifically designed to operate in pairs. Within each pair, there is a hierarchical structure: a leader and a follower. This hierarchy is established based on the signal strength recorded at their current positions. In the figure below, the red agent represents the leader agent and its previous position, while the green agent signifies the follower agent.



**Figure 3.2:** The figure illustrates the triangulation method used in the pair-based gradient search. As shown, the follower agent (P2, depicted in green) is positioned at a 90° offset from the trajectory of the leader agent (from  $P1_{last}$  to  $P1$ , depicted in red). This setup ensures non-colinear positioning of sample points, facilitating accurate gradient determination.

Introducing the gradient at position  $P1$ , it is described by the following equation:

$$\nabla_{P1} = f(P1, RSS1, P1_{last}, RSS1_{last}, P2, RSS2) \quad (3.1)$$

Where:  $P1$  - Current position of the leader,  $RSS1$  - Signal strength at  $P1$ ,  $P1_{last}$  - Previous position of the leader,  $RSS1_{last}$  - Signal strength at  $P1_{last}$ ,  $P2$  - Position of the follower,  $RSS2$  - Signal strength at  $P2$ .

The gradient at position  $P1$  with respect to other positions is defined as:

$$\nabla_i = (RSS_i - RSS1) \times (P_i - P1) \quad \text{for } i = 1', 2$$

The average direction pointing towards the signal source, based on these gradients, is:

$$\text{src\_dir} = \frac{\nabla_1' + \nabla_2}{2}$$

Following this, the normalized direction becomes:

$$\text{norm\_dir} = \frac{\text{src\_dir}}{\|\text{src\_dir}\|}$$

The desired velocity for circling is then described by:

$$v = [\text{norm\_dir}_x, \text{norm\_dir}_y]$$

- **Observations  $O$ :** Observations encapsulate the signal strength values collected by the agents at distinct locations. These values are labeled as  $P1$ ,  $RSS1$ ,  $P1_{last}$ ,  $RSS1_{last}$ ,  $P2$ , and  $RSS2$ .
- **Actions  $A$ :** Agents can either move towards a partner's position, adjust velocity based on gradient measurements, or remain stationary, depending on their hierarchy and relative positions.
- **Internal States  $I$ :** These states encompass the current position, velocity, and role (leader or follower) of an agent. The perception function evaluates the agent's environment to decide on the next action.
- **Inter-Agent Interactions:** Agents constantly communicate with their designated partners to coordinate movement and direction, ensuring optimal gradient-based search.
- **Procedure:** Agents initiate the search by finding a partner. Once paired, leaders determine the direction based on signal strength gradients while followers adjust their position relative to their leaders. The process continues until termination criteria, such as reaching a gradient threshold, are met. Refer to [Algorithm 3](#) for detailed procedures.

### Parameters and Tuning

- **Key Parameters:**
  - **Agent Initialisation:** Initially, agents are positioned uniformly within the search area. Their roles as leaders or followers are determined by comparing the signal strengths they receive.
  - **Follower Offset:** The perpendicular distance the follower should maintain from the leader's trajectory.
  - **Follower Velocity:** The velocity of the follower agent towards the target.
- **Tuning:** Proper calibration of the parameters is essential for optimal performance. The follower offset is critical; a large offset might disrupt gradient readings due to non-linear signal propagation, whereas a smaller one could risk triangulation due to overshooting. Similarly, the follower velocity needs careful adjustment to prevent overreaching or lagging behind the target position.



### Hypothesis for Testing and Prospects

- **Hypothesis:** The pair-based gradient search is anticipated to be effective, particularly in scenarios where fewer agents are required. However, the inclusion of pairs may introduce delays. Additionally, non-linear signal propagation may disrupt the gradient-direction determination process.
- **Advantages:**
  - Feasibility of smaller UAV teams in optimal scenarios.
  - High probability of algorithm convergence due to continuous gradient reassessment.
- **Limitations:**
  - Sensitivity to local disruptions which may lead to inefficient search paths.
  - Potential instability arising from non-linear signal propagation.

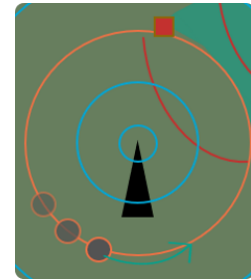
## 3.2. Communication Line Search & Other Specialized Algorithms

The methodology then progresses to the Communication Line Search, as outlined in [subsection 1.1.2](#), which involves establishing communication lines between omni-directional and directional transmitters. This phase's method is applied to the Base Setup. Next to this, some other practical algorithms are discussed in this section.

### 3.2.1. Communication Line Search

#### Symbols and Definitions:

- **Agent Properties:** The communication line search is done by a single agent. In [Figure 3.3](#), the agent's trajectory is visualized. The full dot depicts the agent at the current position,  $\mathbf{p}$ . The two vague dots trailing behind the agent represent the historical positions. The closer dot is the agent's last position,  $\mathbf{p}_{\text{last}}$  (also referred to as the last received signal strength,  $RSS'$ ), and the farther dot signifies the agent's position before the last,  $\mathbf{p}_{\text{pre-last}}$  (or the signal strength two steps back,  $RSS''$ ). Using these historical positions in combination with the signal strengths, the agent adjusts its velocity to circle towards the red square, which acts as the mapping point.
- **Observations  $O$ :**
  - $\mathbf{p}$ : Current position of the agent.
  - $\mathbf{p}_{\text{last}}$  or  $RSS'$ : Agent's last position.
  - $\mathbf{p}_{\text{pre-last}}$  or  $RSS''$ : Agent's position before the last.
- **Actions  $A$ :** Actions or the trajectory adjustments based on velocities and signal strengths.
- **Internal States  $I$ :** Internal States including previous signal strengths, highest signal strength (recorded as  $\mathbf{I}_{\text{max}}$ ), and a counter ( $\mathbf{I}_{\text{count}}$ ) for consecutive signal strength decreases.



**Figure 3.3:** Trajectory of the agent using historical positions to circle towards the mapping point.

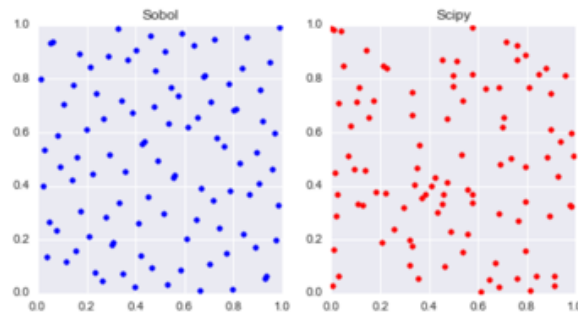
**Inter-Agent Interactions:** A single agent leverages historical observations,  $\mathbf{O}$ , for making motion decisions.

**Procedure:** The procedures detail how an agent searches a communication line and decides when to stop at the mapping point. In 'circle\_velocity' (see [Algorithm 4](#)), the agent calculates its next move using its recent positions and observed signal strengths. In 'SignalStop' (see [Algorithm 5](#)), the agent decides whether to continue moving or to halt, based on the current and past signals.

### 3.2.2. Sobol Sequence

The Sobol sequence, a type of low-discrepancy sequence, is frequently utilized in quasi-Monte Carlo methods designed for numerical integration. Unlike traditional random numbers that are inherently scattered, Sobol sequences are specifically designed to uniformly fill the space. In the realm of Transmitter Search algorithms, ensuring a diverse and efficient search is essential. To this end, the Sobol sequence can be relevant. A new technique named WELL was introduced, leveraging a low-discrepancy sequence, like the Sobol sequence, to improve population diversity and convergence in PSO. [2]

Figure 3.4 provides a visual comparison between points generated using the Sobol sequence and those using a typical random number generator from the Scipy library. The Sobol sequence, as shown in the left subplot, produces points that more uniformly fill the space. In contrast, the Scipy-generated points on the right are scattered and may cluster in certain areas, potentially leading to gaps or oversampling in specific regions.

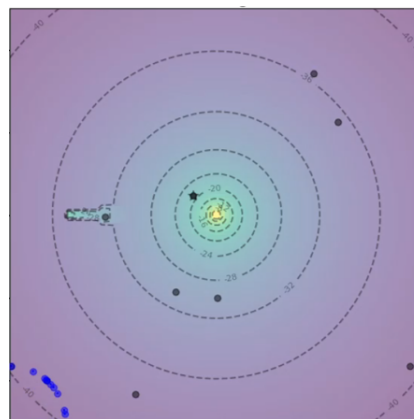


**Figure 3.4:** Comparison between Sobol sequence (left) and Scipy's random number generation (right). Adapted from [8].

The algorithm provided in Algorithm 6 illustrates the generation process for a Sobol sequence. It takes into consideration the desired dimension and the number of points, ensuring that the output sequence uniformly spans the space.

### 3.2.3. One Point Departure Strategy

In practical scenarios, it's common for agents to initiate their search from a single departure point. Despite starting from this unified location, it remains imperative for agents to venture towards points distributed randomly across the search space, ensuring that no area is overlooked. Adopting a "one point departure" approach can potentially expedite the location of the transmitter. In Figure 3.5, the one point departure strategy is depicted. Agents commence their search from a single corner point, represented by the blue dots. The grey dots serve as intermediate destinations for the agents, ensuring a comprehensive coverage of the search area.



**Figure 3.5:** One Point Departure Strategy with Intermediate Destinations.

The pseudocode detailed in [Algorithm 0](#) provides a systematic method for initializing agents in this manner, setting their course from a common origin towards random target points within the search area.

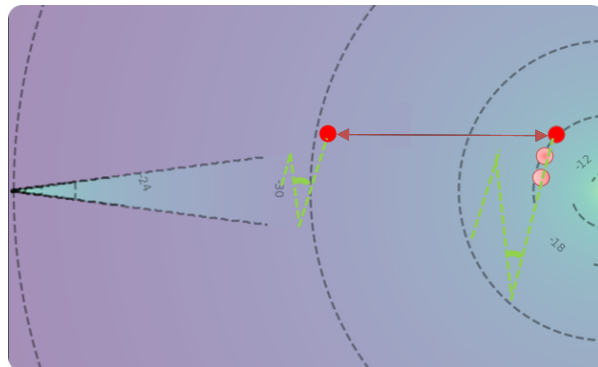
### 3.3. Communication Line Mapping

The methodology transitions to Communication Line Mapping, described in [subsection 1.1.3](#) where agents map the established communication lines. The application of these methods is integrated with the Base Setup.

#### 3.3.1. Frontier Boustrophedon Mapping

##### Algorithm Description

- **Overview:** The Frontier Boustrophedon Mapping focuses on cooperative area coverage. Agents leverage area partitioning or frontier-based methods to divide the area they cover. The mapping boundary is set by the lobe borders of the directional signal. Upon entering this area, agents adhere to a systematic pattern. In the presented algorithm, the trajectory of the agents is based on the boustrophedon pattern. They traverse back and forth across the width, making pronounced turns at the boundaries.
- **Symbols and Definitions:**
  - **Agent Properties:** [Figure 3.6](#) illustrates key components associated with the mapping agents. The history points of a circling agent play an instrumental role in defining initial velocity and establishing the helper agent's offset. The offset distance is represented by the red arrow in the figure, while the turn angles are shown using green arcs. The dotted green trajectories mark the agent's path, which adheres to the boustrophedon pattern.



**Figure 3.6:** Visualization of the Frontier Boustrophedon Mapping. Highlighting offset distance, turn angles, and the agent trajectory following boustrophedon pattern.

- **Observations  $O$ :** Observations include the main mapper's previous velocity ( $last\_velocity$ ), the position before the last known position ( $pre\_last\_position$ ), areas mapped by the helper mapper ( $mapped\_areas$ ), the location where interference is noted ( $interference\_point$ ), and the reference transmission point ( $RT_x$ ).
- **Actions  $A$ :** Key actions in the algorithm encompass adjusting the velocity, updating the position, initiating the mapping process, tracking data, relocating agents, and checking proximity conditions for  $RT_x$ .
- **Internal States  $I$ :** These states are critical for the mapping process. They comprise the negated last velocity ( $v_{neg}$ ), the newly assigned velocity to the main mapper ( $v_{new}$ ), the computed new position for the helper mapper ( $p_{new}$ ), and the main mapper's present velocity ( $v_{current}$ ).
- **Inter-Agent Interactions:** The `main_mapper` and `helper_mapper` have defined roles and interactions during the mapping process to ensure efficient area coverage.



center, and the anticipated trajectory of the main agent, which is defined by the spiral borders.

- **Actions A:** Significant actions include altering the agent’s state, adjusting the velocity of the agent, pinpointing the subsequent position on the spiral, and accumulating data for the purpose of mapping.
- **Internal States I:** Crucial states related to this algorithm are the current leg of the spiral the agent is traversing, the central reference point of the spiral, and the current velocity of the agent.
- **Inter-Agent Interactions:** Clear distinctions are drawn between the roles and engagements of the main\_mapper and the helper\_mapper. The primary function of the helper agent is to aid in charting the trajectory of the spiral, whereas the main mapper adheres to the spiral pattern for optimized mapping.
- **Procedure:** The primary objective of the algorithm is that the agents are tracing the inward trajectory of the spiral. Consistent adjustments to their positions and velocities are carried out in line with the demands of the spiral. An expansive breakdown of this is captured in the pseudocode as depicted in [Algorithm 9](#).

### Parameters and Tuning

- **Key Parameters:** The efficacy of the spiral mapping algorithm is heavily influenced by its key parameters:
  - **Helper agent offset:** This determines the distance between the main agent and the helper agent, crucial for determining the trajectory of the spiral.
  - **Turn angle:** A small adjustment, crucial for the agent’s rotation and, thus, the shape and precision of the spiral.
  - **Velocity:** The speed at which the agent moves can influence the mapping rate and accuracy, especially in environments with varying attributes.
- **Tuning:** Tuning these parameters is critical based on the environment and the specific requirements of the task. This might involve iterative adjustments or using optimization techniques to find the best parameter values.

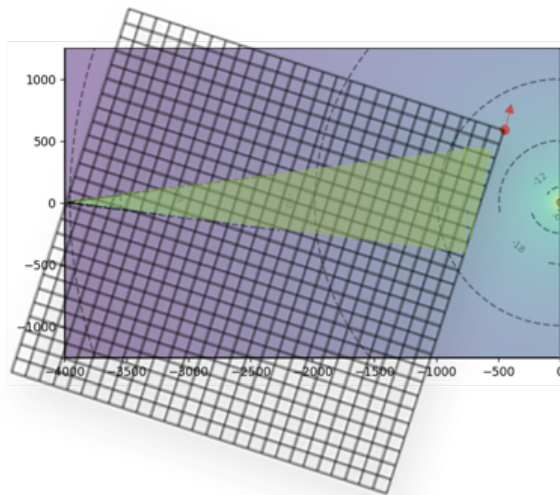
### Hypothesis for Testing and Prospects

- **Hypothesis:** The spiral mapping approach is anticipated to deliver good performance, especially in environments requiring detailed mapping. The performance, however, could be contingent on careful parameter tuning, and there’s a dependency on prior knowledge, specifically the cone angle.
- **Advantages:**
  - Provides a systematic approach for efficient spatial mapping, especially in complex environments.
  - The helper agent’s involvement aids in refining the spiral trajectory, enhancing mapping precision.
  - Allows for adaptability, where agents can dynamically adjust based on the spiral’s requirements.
- **Limitations:**
  - Dependency on prior information, like the cone angle, can restrict its application in completely unknown terrains.
  - Sensitive to parameter adjustments, particularly the turn angle and helper agent offset.
  - Might require frequent recalibrations based on environmental changes or agent dynamics.

### 3.3.3. Multi-agent Q-learning

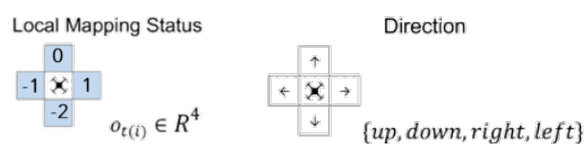
#### Algorithm Description

- **Overview:** A study from NYU demonstrated the effective deployment of multiple agents in a square environment using tabular Q-learning, achieving substantial coverage [9]. This approach found that utilizing mapping status within an agent's Field of View as a state offers advantages in coverage and scalability over using their positions. Q-learning, consistent with other reinforcement learning algorithms, operates within the Markov Decision Processes (MDPs) framework. In this framework, the transition probability to a particular state is solely influenced by the current state and action, reflecting the Markov property.
- **Symbols and Definitions:**
  - **Agent Properties:** In this algorithm, both state and actions are discretized, necessitating the initialization of a grid for agent maneuvers. The first mapping agent establishes a grid starting from the mapping point, extending in the direction opposite to its historical point with cells of 30m x 30m, as depicted in Figure 3.8. The amount of cells in x' and y' direction or chosen to be large such that the grid covers the whole communication line. This grid establishes a discrete framework for state observations and action executions to map the communication line, depicted as the green area in the figure.



**Figure 3.8:** Illustration of the initialized grid by the first mapping agent, starting from the mapping point and extending opposite to its historical point. The green area indicates the communication line being mapped.

- **Observations  $O$ :** Agents operate within a shared environment where their observations of local mapping status are interdependent. The observation for agent  $i$  consists of the mapping status of neighboring cells, characterized by values 0 (undiscovered), 1 (mapped), -1 (obstacle - not considered), and -2 (boundary), as depicted on the left side of Figure 3.9.
- **Actions  $A$ :** Agents can move in four possible directions: up, down, left, and right. These actions are represented as  $a_i \in \{a_{up}, a_{down}, a_{left}, a_{right}\}$ , depicted on the right side of Figure 3.9.



**Figure 3.9:** Illustration showing local mapping statuses (left) and potential actions (right).

- **Internal States  $I$ :** For agent  $i$ , its internal state is solely described by  $o_i(t)$ , which is the observation of the mapping status of its adjacent cells.
- **Inter-Agent Interactions:** Co-existence in a shared space, where agents' actions affect one another's states and rewards, leading to potential collaboration or competition.
- **Procedure:** The model employs a Centralized Training with Decentralized Execution (CTDE) approach. During the training phase, resources are shared among agents. Once training concludes, each agent operates based on its independent policy. The learning process is further enhanced through curriculum learning. This involves a gradual increase in task complexity, ensuring agents first instill foundational behavior. As training progresses, both the diversity and complexity of training scenarios adapt to better meet the requirements. Pre-training takes place in a modified base environment inspired by [9]. This environment is triangular with borders that are not pre-defined; agents determine these borders based on signal strength. The specific training algorithm used is outlined in Algorithm 10. The pre-training phase consists of 200,000 episodes, following the successful strategies employed in the square 10 x 10 environment as documented in [9].

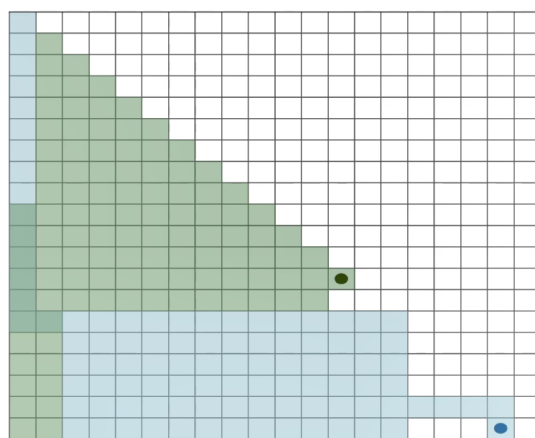


Figure 3.10: Triangular environment used for pre-training.

Subsequent to pre-training, agents undergo an additional 30,000 episodes of training in the NLR - EW environment, as visualized in Figure 3.8.

### Parameters and Tuning

- **Key Parameters:** For the Q-learning algorithm, as presented in Algorithm 10, understanding and tuning critical parameters is paramount for achieving optimal outcomes. These include:
  - **Learning Rate ( $\alpha$ ):** Sets how much new information affects learned information. ( $\alpha = 0.01$ )
  - **Exploration-Exploitation Balance:** Managed by the epsilon parameter ( $\epsilon$ ), it helps the agent decide when to explore new actions versus exploit known actions. ( $\epsilon$  start = 1,  $\epsilon$  end = 0)
  - **Discount Factor ( $\gamma$ ):** Shows the weight of future rewards versus immediate ones. ( $\gamma = 0.5$ )
  - **Reward Decay ( $\beta$ ):** The rate at which the reward value decreases over time. ( $\beta = 0.9999$ )
  - **Reward Structure:** Encourages certain actions in the agent. (Reward = +1 for each new cell discovered)
  - **State Space:** Defines how the agent sees and understands its environment. (Field of View: 4 cells)



- **Number of Training Episodes:** Influences how much the agent learns from its environment. (200,000 episodes)
- **Tuning:** Given the complexities introduced, especially in a multi-agent environment, it's crucial to adapt these parameters effectively. The tuning for both pre-training and the NLR - EW environment was inspired by [9]. Reward structure, state space, and the number of training episodes are also important to tune as they highly determine the algorithm's performance. This fine-tuning can be approached through iterative evaluation over multiple training episodes or utilizing a validation set for more precise calibration.

### Hypothesis for Testing and Prospects

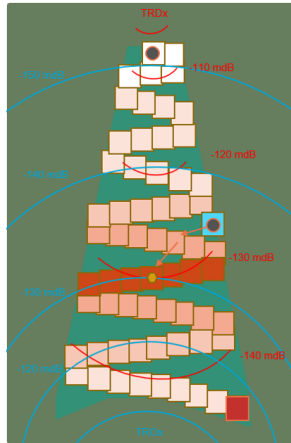
- **Hypothesis:** Curriculum learning, which incorporates pre-training, is expected to ground agents in foundational mapping behaviors before they navigate more complex and scaled environments. Multi-agent Q-learning, in such adaptive scenarios, should foster cooperation. Nevertheless, there may be challenges where agents occasionally get stuck, but introducing a random action policy can likely address this. A constrained field of view may result in overlooking certain encapsulated regions.
- **Advantages:**
  - Enables agents to grasp basic mapping behavior before facing larger challenges.
  - Encourages cooperative learning and mapping in shared, dynamic environments.
  - Offers adaptability to changes in environments and agent interactions.
  - Presents scalability for accommodating a greater number of agents.
- **Limitations:**
  - Susceptible to becoming trapped in particular states.
  - Might miss out on certain areas due to a restricted field of view.
  - Performance can be significantly influenced by the efficacy of parameter adjustments and the design of the reward mechanism.

### 3.4. Interference Point Finding

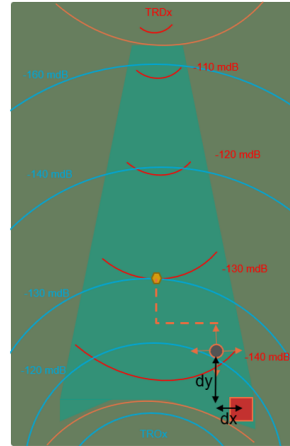
Lastly, the methodology addresses the Interference Point Finding phase, elaborated in [subsection 1.1.3](#), focusing on identifying optimal points for signal interference. This phase's methods are also applied within the framework of the Base Setup.

There are two methods for finding the interference point. The first approach is predicated on the mapping of the communication line. Following this, a UAV is directed to the interference point, which is defined as the location where the agent records the optimal objective measure, as depicted in [Figure 3.11](#). The objective function in this case is the minimisation of the difference in RSS levels of the two communication signals.





**Figure 3.11:**  
Communication Line  
Mapping



**Figure 3.12:** Reinforcement  
Learning Method

In the second approach, Reinforcement Learning (RL) is used to identify the interference point. The method involves training an agent through the Proximal Policy Optimization (PPO) algorithm in the environment presented in [Figure 3.12](#). PPO, which is an advanced actor-critic method as referenced by [\[12\]](#), merges the benefits of both value-based and policy-gradient techniques by implementing separate networks for the policy, known as the actor, and the value function, referred to as the critic.

The RL framework is characterized by defined state representations, action spaces, and a reward structure to facilitate the agent's learning process and execution.

**Algorithm Description** The application of the PPO algorithm for training an agent to locate an interference point is structured as follows:

- **Overview:** The PPO algorithm is implemented to train the agent. The agent learns to identify the interference point using feedback from its actions within the environment.
- **Symbols and Definitions:**
  - **Agent Properties:**
  - **Observations  $O$ :** The agent's observations include its relative position  $(d_x, d_y)$  and the Received Signal Strength (RSS) values, which are essential in determining its actions. These observations are collectively represented as  $s = (d_x, d_y, RSS_o, RSS_d)$ . The agent's available actions ( $a$ ) encompass a set of discrete choices, including staying stationary or moving at predefined speeds, expressed as  $a = \{\text{stay}, \text{move } 10 \text{ m/s}, \text{move } 30 \text{ m/s}\}$ . The reward function, defined as  $R = -\frac{|RSS_d - RSS_o|}{\text{max\_diff}}$ , where  $\text{max\_diff}$  is equivalent to the RSS at the RT (orange circle in [Figure 3.12](#)), is designed to effectively guide the agent's actions. This function penalizes the difference in signal strengths ( $RSS_d$  and  $RSS_o$ ), incentivizing the agent to reduce this disparity to enhance efficiency in navigation and positioning for optimal interference.
  - **Actions  $A$ :** The actions available to the agent are discrete and include staying still, moving at set speeds in the x or y directions, and identifying the interference point. The action set is  $a = \{\text{stay}, \text{move } 10 \text{ m/s}, \text{move } 30 \text{ m/s}\}$ .
  - **Internal States  $I$ :** The agent's state is represented by its relative position and the signal strength readings, formulated as  $s = (\Delta x, \Delta y, RSS_{\text{current}}, RSS_{\text{previous}})$ .
- **Inter-Agent Interactions:** Since only one agent is used, there are no interactions with other agents.
- **Procedure:** The agent is placed within the environment and employs the PPO algorithm to refine its policy in response to the reward signals obtained. This iterative process persists until

the agent demonstrates a reliable ability to identify the interference point, achieving a favorable reward score. For a detailed explanation of the PPO algorithm, refer to [Algorithm 14](#).

### Parameters and Tuning

- **Key Parameters:** The primary parameters consist of coefficients for the reward function that motivate the agent's objectives, variables related to the agent's ability to sense and act within its environment, and PPO-specific hyperparameters that control the learning process. Critical PPO hyperparameters include the learning rate, discount factor ( $\gamma$ ), clip range, and batch size. For example, the learning rate determines the step size in updating weights, while  $\gamma$  affects how future rewards are discounted.
  - **Learning Rate:** Affects the magnitude of updates to the policy network. A typical value might be 0.0002.
  - **Gamma ( $\gamma$ ):** The discount factor which balances immediate and future rewards, commonly set around 0.95.
  - **Clip Range:** Used in the PPO clipping objective, usually set to 0.2 to limit the policy update step.
  - **Batch Size:** Determines how many experiences are used in one update, with a common size being 64 experiences.
- **Tuning:** Parameter tuning involves making systematic changes to the key parameters to optimize the learning process of the agent. The process includes varying the number of episodes, adjusting the discount factor ( $\gamma$ ), refining the learning rate, modifying the state space representation, and improving the reward function. This iterative adjustment helps the agent better adapt to various environmental conditions and interference point locations.

### Hypothesis for Testing and Prospects

- **Hypothesis:** It is posited that while Reinforcement Learning (RL) may achieve faster interference point localization than mapping-based methods, it could yield greater deviation from the actual interference point due to potential dynamic instabilities.
- **Advantages:** The primary advantage of RL is its speed in progressing toward the interference point, as it does not require the preliminary step of data collection that mapping entails. Conversely, the mapping-based approach is grounded in data accuracy, ensuring that the interference point identified is based on comprehensive signal analysis.
- **Limitations:** A key limitation of the RL approach is its reliance on a reward function, which may not capture the complexity of the environment fully, leading to less reliable localization. Mapping-based methods, while data-rich, are constrained by the time and resources required to collect and analyze data before localization can commence.

# 4

## Verification & Validation

This appendix presents an evaluation of the system’s performance, encompassing both verification steps and a discussion on the validation. The verification process, as outlined in [section 4.1](#), verifies various components of the system, including the EW environment functions in [subsection 4.1.1](#) and mission tasks specific funtions in (autorefsec: verification PSO, [subsection 4.1.3](#), [subsection 4.1.4](#) and [subsection 4.1.5](#). Following verification, [section 4.3](#) provides an overview of how the model’s behavior corresponds with real-world phenomena.

### 4.1. Verification

#### 4.1.1. EW Environment Functions Testing

To evaluate the EW integration’s functionality, we conducted tests to verify the system’s signal detection and processing capabilities within a controlled simulation setting. For clarity in the test setup, it is important to note that in the used coordinate system, positive values in the x, y, and z directions correspond to west, north, and down, respectively. This coordinate system explains the negative z-values used to represent positive elevation. A directional transmitter was placed at coordinates (-4000, 0, -100), implying a location 4000 meters west and 100 meters above the origin. Concurrently, an omni-directional transmitter was positioned at (0, 0, -100), as depicted in [Figure 1.3](#). The elevation of both transmitters was chosen to acquire effective signal propagation.

Receivers, or UAVs, were strategically positioned at the transmitters’ exact locations and at points 100 meters to the east, west, and directly above at ground level. The objective was to record signal strengths at these points to determine the transmitters’ coverage and effectiveness. The signal strength was measured in decibels (dB), considering the different wavelength characteristics associated with the directional and omnidirectional transmitters.

**Table 4.1:** Signal Strength and Wavelength Measurements

Receiver Location	Signal Power (dBm) for $\lambda_{dir}$	Signal Power (dBm) for $\lambda_{omni-dir}$
(-4000, 0, -100)	nan	-36.08
(-3900, 0, -100)	-4.95	-35.86
(-4100, 0, -100)	nan	-36.30
(-4000, 0, 0)	nan	-36.08
(0, 0, -100)	15.96	-36.95
(100, 0, -100)	-4.09	-37.17
(-100, 0, -100)	-4.09	-36.73
(0, 0, 0)	15.96	-36.95

The results in [Table 4.1](#) reveal distinct propagation characteristics for directional and omnidirectional transmitters. The directional transmitter, with a 15-degree lobe, shows no signal reception (nan) directly above it, which verifies the expected lobing pattern. Reception is confirmed at 100 meters in the signal’s direction, indicating effective lobing. No signal was detected on the opposite side, highlighting the

unidirectional nature of the transmission.

For the omnidirectional transmitter, significant signal strength is observed directly above, consistent with its design for z-axis propagation. Measurements to the west and east are equal, reflecting the expected isotropic radiation pattern. Additionally, the signal strength for both transmitters remains unaffected at ground level, suggesting negligible impact from vertical displacement in the simulation environment.

#### 4.1.2. Verification of PSO Algorithm Convergence

The exploration of the convergence properties of the Particle Swarm Optimization (PSO), described in [subsection 3.1.1](#) is critical for confirming its effectiveness in search strategies within a multi-agent system. Convergence is indicative of the algorithm's ability to guide agents to collaborate effectively and efficiently towards an optimal solution. The PSO algorithm's parameters are selected based on established behaviors in swarm intelligence [\[14\]](#), aiming to promote convergence.

In this study, the convergence behavior of the Particle Swarm Optimization (PSO) algorithm was carefully analyzed in an isolated environment through a series of unit tests. These tests involved eight agents operating within a confined 10 km x 10 km area. The objective was to observe how these agents, each allowed a maximum of 200 steps and capable of reaching velocities up to 30 meters per second, navigate towards a collective target. This setup, with a potential travel distance of up to 6000 meters for each agent, provided a comprehensive scope to assess their convergence capabilities.

The tests were conducted 100 times, under conditions that included a convergence threshold of 50 meters. Influencing the agents' movement were key parameters: an inertia weight set at 0.8, a cognitive coefficient of 0.1, and a social coefficient of 0.05 as described in [subsection 3.1.1](#). Agents were initially positioned at random points within the designated boundary. This structured approach in an isolated setting allowed for a clear observation of the PSO algorithm's effectiveness in achieving convergence.

The testing framework revealed that, across 100 independent runs, the agents consistently converged toward each other within the stipulated 50-meter threshold. This consistent convergence within the range of 200 iterations provides compelling evidence that the PSO algorithm, with the chosen parameters, is adept at facilitating a cohesive search strategy among the agents.

Reflecting a more realistic scenario where UAVs often start from a single location, this setup was also simulated and explained in [subsection 3.2.3](#). It was observed that locating the transmitter occasionally happened quicker due to the initial spread of the agents, enhancing their chances of encountering the transmitter.

#### 4.1.3. Verification of Gradient Determination Algorithm

The verification of the gradient and source direction determination described in [subsection 3.1.2](#) was conducted through tests within a simulated 10 km x 10 km environment. The objective was to verify the algorithm's ability to accurately identify the direction of the signal and is verified by the ground truth.

In each test, three agents were strategically placed: agent A at a randomly selected location, and agents B and C at points 5 meters east and north of agent A, respectively. The signal strength at each agent's location was determined using a logarithmic function of their distance from the origin (0,0), thus emulating a signal source and its propagation characteristics.

The primary focus was to evaluate the gradient calculation method. The algorithm calculated gradients based on the variations in signal strength at the locations of agents B and C relative to agent A. These gradients were averaged to ascertain the source direction, further normalized to ensure it is a unit vector for precise directional analysis.

A tolerance of  $1e-2$  was adopted for these tests, striking a balance between accuracy and the realities of computational and environmental variations. The success of each test was measured by comparing the computed source direction with the true direction towards the origin, employing a tolerance parameter to accommodate minor deviations.

The setup involved an offset of 5 meters for agents B and C from agent A and evaluating the algorithm's

response to this spatial arrangement. The results demonstrated that the algorithm consistently aligned the computed source direction with the actual direction towards the origin, falling within the acceptable tolerance level.

#### 4.1.4. Verification of Mapping in Mapping Area

The test aims to verify that the mapping occurs within the designated mapping area. This area is delineated by a triangle, determined by the positioning of the directional transmitter, its lobe angle, RSS thresholds, and the location of the omni-directional transmitter. The verification focused on ensuring that the agents' mapping activities are confined to this specified area.

To evaluate this, a series of tests were conducted with Boustrophedon mapping described in [subsection 3.3.1](#), analyzing the agents' mapping data. The primary indicators used for evaluation were the fraction of the agents' positions falling outside the defined triangular mapping area and the average distance by which these positions deviated from the boundaries of this area.

The results from ten independent runs showed an average fraction of 0.112 for positions outside the triangular area. This indicates that, on average, around 11.2% of the mapping occurred outside the predefined area. Additionally, the average distance of deviation for these outlying positions was 24.723 meters, suggesting a relatively close proximity to the intended area.

Furthermore, the standard deviation in the fraction of positions outside the area was 0.0102, and the standard deviation in the average deviation distance was 5.042 meters. These deviations underline a consistent performance across different test runs, reinforcing the reliability of the mapping strategy within the set boundaries.

Overall, these results affirm that the majority of the mapping activities are effectively concentrated within the designated area, with only a minor fraction slightly deviating. This deviation was expected as there is a certain overshoot of the agents with their maximum velocity of 30m/s.

#### 4.1.5. Verification of Updates in the MCTS Tree

This section focuses on verifying the updates in the Monte Carlo Tree Search (MCTS) tree, shown in [Figure 4.1](#). The primary goal is to examine the values of the chosen decision node and assess whether they exhibit expected behavior.

The tree, depicted in [Figure 4.1](#), illustrates the decision nodes within the MCTS framework. This figure was generated using a uniform seed of 23, with 20 agents and 200,000 steps per layer, and a constant  $C$  value of 1.

Level	(u, v)_1	(u, v)_2	(u, v)_3	(u, v)_4	(u, v)_5	(u, v)_6	(u, v)_7	(u, v)_8	(u, v)_9	(u, v)_10	(u, v)_11	(u, v)_12	(u, v)_13	(u, v)_14	(u, v)_15	(u, v)_16	(u, v)_17	(u, v)_18	(u, v)_19	(u, v)_20	(u, v)_21
1	<b>3.3, 1</b>																				
2	-0.0, 55	<b>3.0, 1</b>																			
3	0.4, 112	0.8, 58	<b>3.0, 2</b>																		
4	0.7, 172	1.1, 118	1.3, 62	<b>3.5, 1</b>																	
5	0.8, 231	1.0, 177	1.1, 121	1.0, 60	<b>2.3, 1</b>																
6	0.8, 293	1.0, 239	1.1, 183	0.9, 122	0.9, 63	<b>2.4, 5</b>															
7	0.8, 357	1.0, 303	1.1, 247	1.0, 186	1.0, 127	1.2, 69	<b>2.5, 3</b>														
8	1.0, 428	1.1, 374	1.2, 318	1.1, 257	1.2, 198	1.3, 140	1.6, 74	<b>2.7, 4</b>													
9	1.2, 507	1.3, 453	1.4, 397	1.4, 336	1.5, 277	1.7, 219	1.9, 153	2.3, 83	<b>3.2, 4</b>												
10	1.3, 587	1.4, 533	1.5, 477	1.5, 416	1.6, 357	1.7, 299	1.9, 233	2.1, 163	2.0, 84	<b>2.7, 5</b>											
11	1.4, 670	1.5, 616	1.6, 560	1.6, 499	1.7, 440	1.8, 382	2.0, 316	2.1, 246	2.1, 167	2.2, 88	<b>3.0, 4</b>										
12	1.5, 759	1.6, 705	1.7, 649	1.7, 588	1.8, 529	1.9, 471	2.0, 405	2.1, 335	2.1, 256	2.2, 177	2.2, 93	<b>3.1, 4</b>									
13	1.6, 861	1.7, 807	1.8, 751	1.8, 690	1.9, 631	2.0, 573	2.1, 507	2.2, 437	2.2, 358	2.3, 279	2.4, 195	2.6, 106	<b>3.2, 4</b>								
14	1.7, 969	1.8, 915	1.9, 859	1.9, 798	2.0, 739	2.1, 681	2.2, 615	2.3, 545	2.3, 466	2.4, 387	2.4, 303	2.5, 214	2.5, 112	<b>3.5, 4</b>							
15	1.9, 1106	2.0, 1052	2.0, 996	2.1, 935	2.1, 876	2.2, 818	2.3, 752	2.4, 682	2.5, 603	2.5, 524	2.6, 440	2.7, 351	2.8, 249	3.0, 141	<b>3.4, 8</b>						
16	2.0, 1264	2.1, 1210	2.2, 1154	2.2, 1093	2.3, 1034	2.4, 976	2.5, 910	2.6, 840	2.6, 761	2.7, 682	2.7, 598	2.9, 509	2.9, 407	3.1, 299	3.2, 166	<b>3.4, 10</b>					
17	2.2, 1465	2.3, 1411	2.4, 1355	2.4, 1294	2.5, 1235	2.6, 1177	2.6, 1111	2.7, 1041	2.8, 962	2.8, 883	2.9, 799	3.0, 710	3.1, 608	3.2, 500	3.3, 367	3.4, 211	<b>3.6, 12</b>				
18	2.3, 1663	2.4, 1609	2.5, 1553	2.5, 1492	2.6, 1433	2.7, 1375	2.7, 1309	2.8, 1239	2.8, 1160	2.9, 1081	3.0, 997	3.0, 908	3.1, 806	3.2, 698	3.3, 565	3.3, 409	3.3, 210	<b>3.8, 12</b>			
19	2.7, 2305	2.8, 2251	2.9, 2195	2.9, 2134	3.0, 2075	3.0, 2017	3.1, 1951	3.2, 1881	3.2, 1802	3.2, 1723	3.3, 1639	3.4, 1550	3.4, 1448	3.5, 1340	3.6, 1207	3.6, 1051	3.7, 852	3.8, 654	<b>3.8, 38</b>		
20	3.1, 3305	3.1, 3251	3.2, 3195	3.2, 3134	3.2, 3075	3.3, 3017	3.3, 2951	3.4, 2881	3.4, 2802	3.5, 2723	3.5, 2639	3.6, 2550	3.6, 2448	3.6, 2340	3.7, 2207	3.7, 2051	3.8, 1852	3.8, 1654	3.8, 1038	<b>3.8, 1000</b>	
21	3.4, 5305	3.4, 5251	3.4, 5195	3.5, 5134	3.5, 5075	3.5, 5017	3.5, 4951	3.6, 4881	3.6, 4802	3.6, 4723	3.7, 4639	3.7, 4550	3.7, 4448	3.7, 4340	3.8, 4207	3.8, 4051	3.8, 3852	3.8, 3654	3.8, 3038	3.8, 3000	<b>3.8, 2000</b>

**Figure 4.1:** MCTS Decision Nodes  $u$ (value),  $v$ (visits) - ( $U$ \_seed: 23, Agents: 20, Steps per layer: 200000,  $C$ : 1)

In the analysis of the MCTS tree patterns emerge. Firstly, there is an increasing trend in the number of nodes up to level 21, correlating with the final average score of 3.8. This indicates the presence of 21 decision moments essential for locating all transmitters in under 2200 steps. Additionally, the average value ( $u$ ) of newly added nodes in each layer initially decreases, a result of additional runs lowering the average. However, this trend reverses over time, showing more stable and increasing values, suggesting

that the algorithm is converging towards a viable solution. Finally, there is an increase in the amount of visits ( $\nu$ ) and average values for selected nodes increase upstream for each search layer, demonstrating the effective role of backpropagation in the process. This trend indicates a systematic refinement and improvement in decision-making as the MCTS algorithm progresses through each layer.

## 4.2. Discussion on Validation

In order to determine whether the to-be-developed simulation model represents realistic scenarios at an acceptable level, validation steps are required. Although the model environment is based on expert knowledge, assumptions and the absence of data regarding model input and output are absent. Validating the model, described in [chapter 2](#), with real-world environments poses challenges due to necessary assumptions aimed at reducing experimental workload and enhancing algorithm discovery. The Royal Netherlands Aerospace Center (NLR) is actively seeking methods to transition from simulated environments to real-world applications. This involves incorporating practical aspects into more sophisticated simulation platforms like Unity or Gazebo, which include sensor observations and processing. The discussion here focuses on the implications of these assumptions for real-world applicability.

## 4.3. Discussion on Validation

This section discusses the necessary steps for validating the simulation model, as mentioned in [chapter 2](#) to ensure its representation of realistic scenarios is at an acceptable level. The model's environment, while informed by expert knowledge, encounters challenges due to assumptions made and the lack of specific data concerning model inputs and outputs. The process of validating the model against real-world environments is not straightforward, largely because of the assumptions in [section 1.3](#) that have been implemented. These assumptions are essential for reducing the experimental workload and facilitating the discovery of effective algorithms. However, they also present potential limitations in terms of the model's fidelity to actual scenarios. The Royal Netherlands Aerospace Center (NLR) is actively engaged in efforts to transition from these simulated environments to practical real-world applications. This transition includes enhancing simulation platforms such as Unity or Gazebo with more realistic features, including sensor observations and data processing capabilities. The primary focus of this discussion is on understanding the impact of these necessary assumptions and assessing how they might influence the model's applicability and effectiveness in real-world settings.

### 4.3.1. Environment

The environmental assumptions in the model also bring notable discrepancies. **E1** neglects phenomena like refraction, reflection, and environmental interference, which can significantly influence signal propagation and detection accuracy. **E2**'s disregard for atmospheric and antenna-induced noise, along with SNR thresholds, simplifies the signal environment and may not accurately represent real-world noisy or cluttered conditions. Furthermore, **E3** assumes that environmental factors such as wind and temperature do not impact UAV performance, omitting critical variables that can affect UAV stability and energy consumption in real-world operations.

### 4.3.2. Operations

In terms of UAV performance, the model introduces several assumptions. **U1** presents UAVs as entities moving solely in x and y directions with a maximum velocity of 30 m/s, disregarding the complex dynamics encountered in varied terrains such as urban or mountainous areas. **U2** assumes uniform sensing and communication capabilities across all UAVs, which may not reflect the diversity in actual UAV systems, potentially impacting operations in different scenarios. **U3** limits UAVs to listening to a single frequency band at a time, a constraint that may not align with the multifrequency capabilities of modern UAVs in complex signal environments. **U4**'s assumption about frequency band matching oversimplifies real-world challenges in signal detection and discrimination. **U5**'s exclusion of onboard signal processing and transmission overlooks key UAV operation aspects. **U6** sets a static stealth maintenance threshold, not accounting for the variable stealth requirements in real operations. Lastly, **U7** restricts communication to a simplistic interaction between an omni-directional and a directional signal, simplifying the intricate nature of real-world signal interactions.

### 4.3.3. Simulation

Regarding simulation, the model incorporates several assumptions for computational feasibility. **S1** parallels real-world constraints with minimal computation and data storage capabilities, yet this may restrict the complexity of scenarios that can be effectively simulated. **S2** assumes instantaneous communication between UAVs, overlooking potential delays and disruptions in real-world communication networks. **S3**'s use of a sequential time-step model abstracts away from the asynchronous decision-making processes in real scenarios, potentially impacting the model's capacity to simulate real-time dynamic interactions. Finally, **S4**'s representation of communication between transmitters as a continuous signal impulse simplifies the complexity of signal interactions, which may involve intermittent or variable signals in the real world.

### 4.3.4. Self-confidence

In scenarios involving environmental noise (**E2**), UAV positioning (**U1**), and frequency hopping (**U3**), the potential benefits of integrating self-awareness or self-confidence measures into agent systems are promising yet uncertain. For **E2**, these measures could enable agents to adapt more effectively to noise variations, possibly enhancing signal processing and decision-making, but the actual improvement is contingent on the noise's characteristics and the agent's design. In **U1**, self-confidence metrics might increase UAV positional accuracy, particularly in GPS-challenged areas or when using dead-reckoning, but the real-world effectiveness of these metrics across different operational contexts is not confirmed. Regarding **U3**, such measures could assist in anticipating and adapting to frequency changes, potentially improving the reliability of secure communications; however, the true impact on frequency hopping depends on the communication environment's complexity and the sophistication of the agent's algorithms. Furthermore, adaptive mapping, as outlined by Hollinger and Sukhatme [6], benefits from self-awareness. Agents equipped with these metrics can better assess the value of their collected data, leading to more effective path planning, especially in areas where the model's data confidence is low.

Within the framework of self-awareness in agents, the process begins with measuring confidence, which includes both quantitative and qualitative assessments. The quantitative aspect involves normalizing metrics like signal quality to a standardized scale, while the qualitative aspect assesses data integrity, such as the success rate of error checks, also normalized to a comparable scale. These two measures are then integrated to form a composite confidence score. Decision-making confidence is further refined by setting threshold values for data reliability, using ensemble methods to build a more robust confidence profile, and potentially employing predictive models to estimate decision reliability. In this context, [Figure 4.2](#) illustrates how agents could use a similar concept, a confidence interval, to gauge the trustworthiness of their sensory inputs and decisions, thereby enhancing their operational effectiveness.



**Figure 4.2:** 95% Confidence Interval

Acknowledging the necessary assumptions and incorporating self-awareness in these models is crucial for accurate interpretations and guiding future enhancements towards more realistic simulations.



# 5

## Specification of Experimental Setups & KPIs

This appendix provides a detailed report on the experimental setups and Key Performance Indicators (KPIs) used in the UAV-based CEMA operations. It encompasses several operational phases: Transmitter Search, divided into Single Transmitter Search (section 5.1) and Multi Transmitter Search (section 5.2), focusing primarily on the detection of transmitters. It also includes a comprehensive discussion on Communication Line Mapping & Interference Point Finding (section 5.3), which involves the mapping of communication lines and the identification of interference points.

### 5.1. Single Transmitter Search

#### 5.1.1. Experimental Setup: E1

In our experimental procedure, both algorithms underwent 100 runs with different initialisation seeds. Each run terminated upon locating TR\_o or after reaching a limit of 10000 steps. Runs that did not successfully locate TR\_o were excluded from the primary analysis. Including non-converging solutions could distort the representative performance metrics, although they are essential for understanding the reliability and robustness of each algorithm.

**Experimental Configuration:** The setup provides a foundational comparison between the PSO and Pair-Gradient approaches.

- Search Area:  $10km^2$  - A sufficiently vast domain ensures varied initial positions and trajectory paths while maintaining consistency between both algorithms.
- Initialization: Uniform and random distribution ensures unbiased and comparable initial conditions for both algorithms.
- Number of Agents: 8 - Designed to balance computational overhead with search efficiency, ensuring consistency in agent density across both methods.
- Cognitive Coefficient (PSO): 0.1 - Dictates the influence of an agent's individual experiences.
- Social Coefficient (PSO): 0.05 - Provides a modest weight to group consensus to prevent premature convergence.
- Inertia Weight (PSO): 0.8 - Sustains search momentum and minimizes erratic agent movements.
- Maximum Velocity (PSO): 30 m/s - Restricts agents, allowing for speedy exploration without overlooking possible solutions.
- Follower Offset (Pair-Gradient): 5 m - Defines a proximate yet separate distance for followers, fostering gradient-dependent motion.
- Follower Velocity (Pair-Gradient): 20 m/s - Facilitates rapid convergence while upholding the gradient-oriented movement.



### 5.1.2. Key Performance Indicators

To evaluate the efficacy, efficiency, and reliability of swarm-based algorithms, a set of Key Performance Indicators (KPIs) have been selected for their relevance to the specific objectives of this mission task. Some have multi-purpose and can be used to analyse the algorithms for communication line mapping as well. Each KPI offers distinct metrics for assessing algorithmic performance under various conditions.

- **Time (Steps):** Speed is essential, especially in time-sensitive tasks such as emergency response or minimizing the swarm's visibility. The total step time serves as a KPI for assessing the algorithm's agility and responsiveness. This metric starts counting when the agents are released from their random initial positions and stops when one agent finds the RT. The total step time is cumulative across all agents.
- **Swarm Mean Distance to RT (Meter):** This KPI evaluates the swarm's clustering by calculating the mean distance to the RSSI Threshold (RT). It offers insights into the swarm's spatial distribution, allowing for the potential assignment of other tasks to agents in different areas. The mean distance is measured when one agent has reached the RT.
- **Trajectory Coverage (Area):** Coverage shows how widely the surveillance is done of the search area. This KPI is calculated as the volume of the Convex Hull formed by the swarm's positions, denoted by  $CH$ . Formally,

$$CH = \text{Volume of Convex Hull}$$

Understanding coverage is essential for assessing the algorithm's capability to explore and map large areas comprehensively. The value of  $CH$  can yield various insights into the swarm's behavior:

- High Coverage: Indicates that the agents are spread out over a large area, maximizing the ground that gets covered.
- Low Coverage: Suggests that the agents are mostly clustered in a small area and have not covered much ground.
- **Trajectory Density ( $m^{-1}$ ):** Density complements coverage by focusing on the depth of the search. It is calculated using Kernel Density Estimation (KDE) with a Gaussian kernel [15]. Specifically,

$$f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)$$

where  $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$ . The mean density is then given by

$$\text{Mean Density} = \text{Mean}(f(x_1), f(x_2), \dots, f(x_n))$$

High density values suggest that the agents are clustered in specific high-density regions within the covered area, which is crucial for specialized tasks like environmental monitoring. The value of the density can be categorized as:

- High Density: Indicates that the agents are clustered in specific high-density regions within the covered area.
- Low Density: Indicates that the agents are evenly distributed across the covered area.

**Additional Insights:** The combination of Coverage and Density KPIs can provide nuanced insights into the swarm's behavior, such as:

- High Coverage, High Density: The agents are covering a large area but are also clustered in specific high-density regions within that large area. This could resemble a forest patrolling scenario.
- High Coverage, Low Density: The agents are spread out over a large area, and they are fairly evenly distributed across this area. This is ideal for search and rescue missions.
- Low Coverage, High Density: The agents are mostly clustered in a small area and have not covered much ground. This could resemble a small research facility.

- Low Coverage, Low Density: The agents are in a small area and are even within that small area. This might indicate a few guards patrolling a small art gallery.

Each of these combinations provides different insights into how the agents are distributed in space, which can be highly useful depending on the specific application or question at hand.

**Energy Usage (Velocity):** This Key Performance Indicator (KPI) assesses the energy efficiency of drones, especially important for long missions or those far from recharging stations. We estimate the total energy  $E$  used by the drone swarm with the simplified formula:

$$E = \sum_{i=1}^n \|v_i\|^2$$

Here,  $v_i$  is the velocity of each drone  $i$ . This formula assumes that energy consumption is mainly driven by the velocity, with the square of velocity representing the energy use. It's a basic approximation focusing on velocity as a key factor, but it doesn't account for other complexities like aerodynamic effects or motor efficiency variations.

A high value for  $E$  usually correlates with high velocities among the agents. While fast motion may be advantageous for rapid response or emergency scenarios, it comes at the cost of depleting energy reserves quickly. In such cases, the mission duration or the distance from a recharging station becomes a critical factor in real-world scenarios. A low  $E$  value generally implies lower velocities and, hence, more sustainable energy usage. This would be ideal for long-duration missions where energy efficiency is prioritized over speed.

- **Compute Time (Seconds):** In applications requiring real-time decisions, low computational time is crucial. This KPI measures the algorithm's efficiency by summing the step times for the entire mission. It is particularly relevant for drones with limited processing power. Time is measured from start of mission task until end of mission task. Only agent updates are considered in determining the compute time.
- **Failure Rate (Percentage):** Reliability is a critical aspect, particularly for mission-critical scenarios where failure is not acceptable. This KPI tracks the number of failed attempts, defined as instances when the swarm doesn't find the RT within 10,000 steps.

Where the steps to RT and failure rate are the most important parameters for this specific mission task, the other KPIs can serve as an indication on how these algorithms perform when the objective is different or the methods are applied in a different field.

## 5.2. Multi Transmitter Search

### 5.2.1. Experimental Setup: E2, E5

**Agent-Based Model Configuration** The agent-based model was set up as follows:

- **Transmitters:** Utilized were six transmitters, with operational frequencies between 3.7 GHz and 4.3 GHz.
- **Positional Seed:** A consistent transmitter positioning was ensured by using a fixed seed value of 10.
- **Search Area:** The model's search area spanned a  $10 \times 10 \text{ km}^2$  grid.
- **Initialization:** Agent placement at the outset was determined by a Sobol sequence, as described in [subsection 3.2.2](#) with a seed value of 0.
- **Agent Count:** The simulation comprised 20 agents.
- **Agent Dynamics:** Directed by a Particle Swarm Optimization (PSO) algorithm, the agents' movements were parameterized with  $c_1$ ,  $c_2$ , and inertia weight  $w$  set to 0.1, 0.05, and 0.8, respectively as described in [subsection 3.1.1](#).

**E2: Optimization of MCTS Parameters** The MCTS parameters were adjusted in the following manner:

- **Steps per Layer:** Trials were conducted with the number of steps per layer varying from 200,000 to 1,000,000 to determine the optimal quantity.
- **Exploration Factor:** The UCB1 algorithm's exploration constant 'C' was tested at values of 0.5, 0.8, and 1.0 to find the most effective parameter.

**E5: Comparative Tests between MUC and MCTS in Multi Transmitter Search** The final experiment focused on comparing the **MUC** and **MCTS** methods in the context of Multi Transmitter Search, with modifications applied to the Global Setups. The experiment entailed two primary variations:

- **Agent Initialization Variation:** Different Sobol initialization seeds were employed to position the 20 agents, with a total of 100 runs conducted for both **MUC** and **MCTS** under each variation.
- **Transmitter Positioning Variation:** Transmitters were placed within a central 8x8 km area, avoiding border locations based on previous findings. A similar set of 100 runs for each method was executed to evaluate this variation.

### 5.2.2. Key Performance Indicators (KPIs)

This section outlines the Key Performance Indicators (KPIs) crucial for assessing the performance of Maximum Utility Coalition (MUC) and Monte Carlo Tree Search (MCTS) solutions in the context of Transmitter Search problems. These KPIs gauge efficiency, speed, and adaptability.

- **Transmitters Found:** This KPI directly measures the success rate of the search algorithm by indicating the number of transmitters located.
- **Total Time Steps Required:** This KPI captures how many steps the algorithm took to achieve its result.
- **Value Score:** An overall score that incorporates both the number of transmitters found and the total steps needed. The formula is:

$$\text{Value Score} = \text{Transmitters Found} - \frac{\text{Total Time Steps Required}}{1000}$$

- **Stationary and Altering Decision Moments:** These metrics provide valuable insights into the dynamics of the coalition-forming process, specifically quantifying the frequency and extent of changes in decision-making throughout the operation.

**Qualitative Analysis:** Trajectory maps can be employed to visualize the effectiveness and movement patterns of the algorithms.

#### Additional metrics for MCTS

For Monte Carlo Tree Search, the following metrics offer additional insights:

- **Amount of Visits per Layer:** Captures the frequency of node visits at each layer of the tree, providing information on the search focus.
- **Average Reward per Layer:** This metric shows the average rewards collected at each tree layer, offering a measure of utility optimization.
- **Amount of Layers:** Indicates the depth of the tree, which can provide information on how far ahead the algorithm is planning.

These KPIs serve as metrics for methodological insights and evaluating the methods' performance.

## 5.3. Communication Line Mapping & Interference Point Finding

### 5.3.1. Experimental Setup: E6, E7

The experimental design entailed 100 runs, with various initialization, per algorithm, with each run ending when the mapped area ceased to grow for a set duration. Runs failing either to map the area or resulting in algorithm crashes were excluded from the primary data analysis. Such exclusion aims to minimize data distortion while remaining informative for assessing algorithmic robustness and reliability.

**Base Configuration:** A common setup is used to ensure unbiased comparison across Boustrophedon, Spiral, and Tab-QL algorithms.

- Base Configuration: Pre-Mapping Algorithm: PSO, Search Area:  $10\text{km}^2$ , Initialization: Uniformly random, Agents: 8
- Helper Offset: 1000 m (Boustrophedon & Spiral) - Helps distribute agents well without causing overshoot.
- Turn Angle:  $5^\circ$  (Boustrophedon) - Enables detailed mapping without high density. Easy to adjust.
- Shrink Factor: 0.97 (Spiral) - Balances mapping density throughout trajectories.
- Border Angle:  $0^\circ$  (Spiral) - Matches the shape of the directional transmitter cone.
- Velocity: 30 m/s - Maximum velocity for the UAVs.
- Pre-Training: 200,000 episodes (Tab-QL) - Allows the model to learn the environment well enough for robust mapping. The used parameters are described in [subsection 3.3.3](#).
- Training Episodes: 30,000 (Tab-QL) - Deployed in the EW environment where episodes are significantly longer than in the setup of [\[9\]](#). The used parameters are described in [subsection 3.3.3](#).

### 5.3.2. Key Performance Indicators (KPIs)

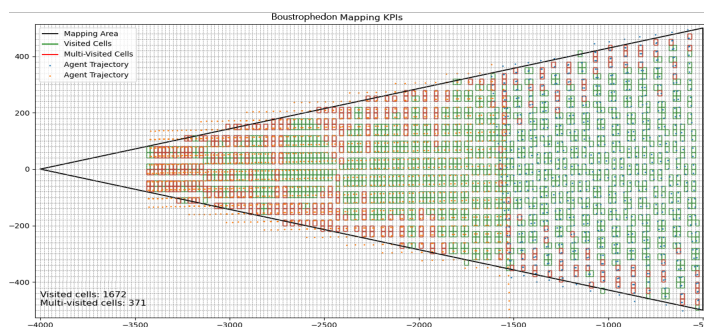
This section describes Key Performance Indicators (KPIs) for evaluating three mapping algorithms: Boustrophedon, Spiral, and Tabular Q-Learning. These KPIs quantify algorithm performance in terms of speed, accuracy, and energy use.

- **Mapping Step Time:** Time is measured from the agent's arrival at the starting point to when the mapped area stops growing for a set time. This KPI assesses how quickly the algorithm completes the mapping task.
- **Localization Error:** This measures the difference between the real interference point and its estimate using RSSI measurements. The formula for error calculation is:

$$\text{Chosen Point} = \text{position with smallest RSSI difference}$$

$$\text{Localization Error} = \text{Distance between true point (X=-2100, Y=0) and chosen point}$$

- **Visited Cells Count:** A  $20 \times 20$  grid is placed over the mapping area as illustrated in [Figure 5.1](#). This KPI counts the cells that the agent crosses once or more, represented by green or red cells in the figure.



**Figure 5.1:**  $20 \times 20$  grid cells (Frontier Boustrophedon)

- **Multiple Visits to Cells:** This KPI counts the cells in the  $20 \times 20$  grid that the agent traverses more than once. Such cells are marked red in the figure.

- **Energy Use:** This sums up the energy spent during the 'Mapping Step Time'. The formula is:

$$E = \sum_{i=1}^n \|v_i\|^2$$

- **Compute Time (Seconds):** Time is measured from start of mission task until end of mission task. Only agent updates are considered in determining the compute time.

These KPIs help in evaluating each algorithm's strong and weak points. They aid in selecting the most fitting algorithm for different needs such as fast mapping, precise interference point finding, or low energy use.

# 6

## Statistical Elaboration

This appendix provides the statistical analyses conducted to assure the significance of the study's results. The methodologies employed include the Shapiro-Wilk test and QQ-plots for data normality assessment, the Coefficient of Variation (CV) for evaluating data consistency and reliability, and the Wilcoxon Signed-Rank Test for comparing paired results. The analyses are done for single target search: E1 in [section 6.1](#), multi target search: E5 in [section 6.2](#) and communication line mapping: E6 in [section 6.3](#), each employing a tailored approach to investigate the distinct aspects of these experimental setups.

### 6.1. Single Target Search: E1

To verify the statistical significance of the results, a detailed statistical analysis is conducted. This analysis starts with the Shapiro-Wilk test [\[13\]](#) and analyzing QQ-plots, for assessing data normality, which determines the selection of appropriate statistical tests. Following this, the stability of the Coefficient of Variation (CV) is evaluated to gauge the consistency and to sense if the system is well captured. Finally, the Wilcoxon Signed-Rank Test [\[11\]](#), a non-parametric method ideal for small or non-normally distributed samples, is used for comparing paired samples. This test identifies statistically significant differences between sets of observations.

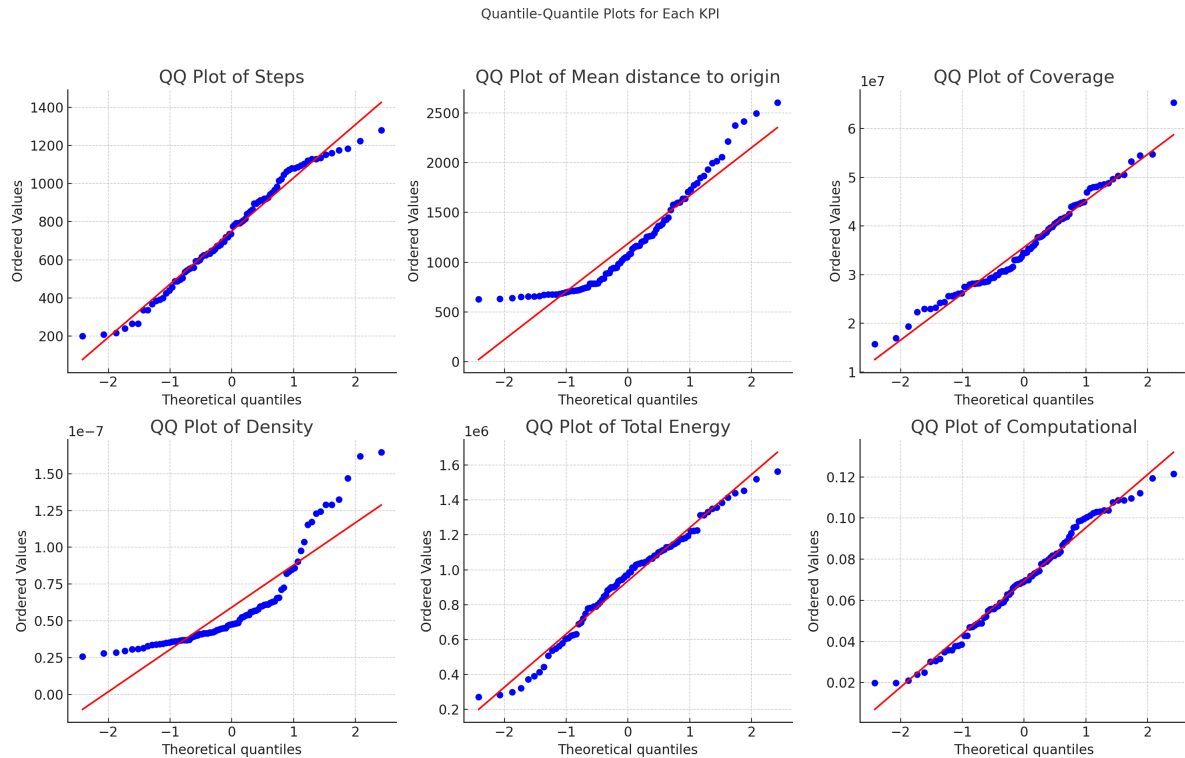
#### Data Distribution

In the statistical evaluation of Key Performance Indicators (KPIs), the Shapiro-Wilk test [\[13\]](#) was employed to determine the normality of the data distributions for each KPI. This test is a widely used method for assessing whether a sample comes from a normally distributed population. It calculates a statistic that indicates how closely a set of values follows a normal distribution; the closer the statistic is to 1, the more normal the distribution.

The findings from the Shapiro-Wilk test are summarized in [Table 6.1](#), providing a clear indication of whether the KPI data conform to a normal distribution. To supplement and visually corroborate these results, Quantile-Quantile (QQ) plots were also utilized. QQ plots are graphical tools for comparing data quantiles to the quantiles of a theoretical distribution – in this case, the normal distribution. By plotting the observed data quantiles against the expected quantiles of a normal distribution, QQ plots help in visually assessing the normality. Deviations from a straight line in the plot indicate departures from normality. Together, the Shapiro-Wilk test and QQ plots offer a comprehensive approach to evaluating the normality of the KPI data, an essential step in further statistical analyses.

**Table 6.1:** Results of Shapiro-Wilk Test and QQ Plot References

KPI	Test Statistic	$p$ -value	QQ Plot Reference
Steps	0.975	0.080	Figure 6.1 Top Left
Mean distance to origin	0.896	$3.46 \times 10^{-6}$	Figure 6.1 Top Middle
Coverage	0.978	0.135	Figure 6.1 Top Right
Density	0.788	$6.28 \times 10^{-10}$	Figure 6.1 Bottom Left
Total Energy	0.979	0.156	Figure 6.1 Bottom Middle
Computational	0.980	0.195	Figure 6.1 Bottom Right

**Figure 6.1:** Quantile-Quantile Plots for each KPI

In summary, a  $p$ -value less than 0.05 indicates non-normality. This criterion is met by **Mean distance to origin** and **Density**, as evidenced by their low  $p$ -values and confirmed by their QQ plots.

### Analyzing Data Variability Through the Coefficient of Variation

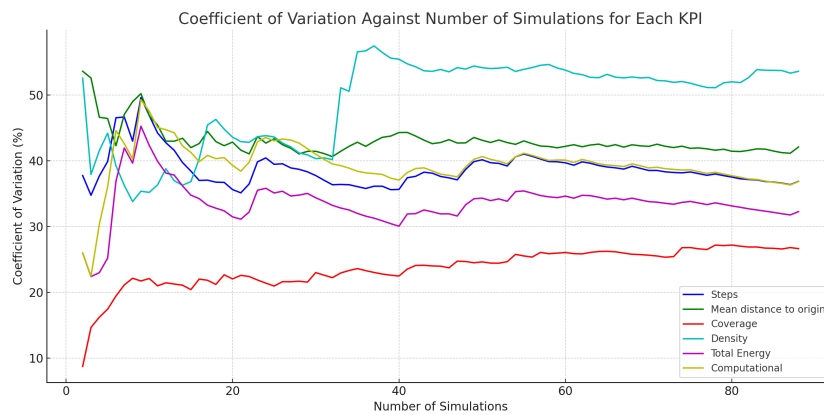
The Coefficient of Variation (CV) serves as a key statistical tool in this analysis, providing insights into the relative variability of each Key Performance Indicator (KPI). The CV, defined as the ratio of the standard deviation to the mean and expressed as a percentage, offers a measure of relative variability, independent of the unit of measurement. It is calculated using the following formula:

$$CV = \left( \frac{\text{Standard Deviation}}{\text{Mean}} \right) \times 100$$

The importance of assessing the CV lies in its ability to indicate the degree of dispersion around the mean value. High CV values suggest a large degree of variability relative to the mean, signaling potential inconsistencies or fluctuations in the data. Conversely, low CV values denote greater stability and consistency in the dataset.

In this study, the CV was calculated for each KPI to examine the reliability and consistency of the findings. As depicted in [Figure 6.2](#), most KPIs demonstrate CV stabilization, implying that the sample

size may be sufficient to draw definitive conclusions about the data's distribution and reliability.



**Figure 6.2:** Coefficient of Variation Against Number of Simulations for each KPI

Considering the mixed outcomes from the Shapiro-Wilk tests and the indication of CV stabilization, coupled with the observations from the QQ plots, a conservative stance in statistical inference is advisable. This analysis suggests treating all KPIs as non-normally distributed. Consequently, this leads to the preference for non-parametric statistical tests in further analyses, as these tests do not rely heavily on assumptions about the data's distribution, thus providing a more robust framework for evaluating the data under the given conditions.

### Statistical Test

The Wilcoxon Signed-Rank Test [11] is an alternative to the paired Student's t-test for cases where data do not follow a normal distribution. This robust test is suitable for evaluating matched or paired samples when the assumption of normality is not met. It evaluates the differences between pairs using ranks rather than raw data, making it less sensitive to outliers.

1. Non-Parametric: It is used when data do not need to be normally distributed.
2. Two Related Samples: It applies to situations involving two sets of related observations.
3. Rank-Based: It focuses on the relative magnitudes of the data by using ranks.

### Test Steps

1. Calculate Differences: Compute the difference  $d = x_i - y_i$  for each paired observation.
2. Ranking: Assign ranks to the absolute differences  $|d|$ . Disregard any differences that are zero.
3. Calculate W statistics:
  - Sum the ranks for positive differences:  $W^+$ .
  - Sum the ranks for negative differences:  $W^-$ .
  - The W statistic is the smaller of the two sums:  $W = \min(W^+, W^-)$ .
4. Significance Testing: Determine if the observed W statistic is significant by comparing it to a critical value or by assessing the p-value.

### Effect Size Calculation

The effect size for the Wilcoxon Signed-Rank Test can be quantified using the rank-biserial correlation, which provides a measure of the strength of the relationship between the paired observations. It is calculated as follows:

$$\text{effect\_size} = 1 - \frac{2W}{n(n+1)}$$



where  $n$  is the number of non-zero differences between pairs. This effect size metric is useful for quantifying the magnitude of an effect independently from the sample size.

**Table 6.2:** Wilcoxon Signed-Rank Test on PSO (1) against Pair-Gradient (2)

KPI	Stat	P-value	Effect Size	PSO Median	Pair-Gradient Median
Steps	226.5	$5.81 \times 10^{-13}$ *	0.9422	756	1204
Mean Distance to Origin	184.0	$1.57 \times 10^{-13}$ *	0.9530	1065.63	2073.96
Coverage	1669.0	0.2292	0.5738	$3.45 \times 10^7$	$3.41 \times 10^7$
Density	1787.0	0.4768	0.5437	$4.77 \times 10^{-8}$	$5.42 \times 10^{-8}$
Total Energy	241.0	$9.05 \times 10^{-13}$ *	0.9385	977927	637050
Computational Time	132.0	$3.01 \times 10^{-14}$ *	0.9663	0.0693	0.1321

\* denotes  $p < 0.05$ , indicating statistical significance.

Table 6.2 shows a comparison between Particle Swarm Optimization (PSO) and Pair-Gradient algorithms. The test results indicate significant differences in Steps, Mean Distance to Origin, Total Energy, and Computational Time with large effect sizes ranging from 0.9422 to 0.9663, highlighting substantial improvements when using PSO. Although Coverage and Density metrics did not reveal statistical significance, their moderate effect sizes of 0.5738 and 0.5437, respectively, suggest that PSO and Pair-Gradient differ in these aspects as well. Overall, the results demonstrate that PSO is more efficient than Pair-Gradient in terms of step count, energy consumption, and computational demands, while maintaining a closer average distance to the origin.

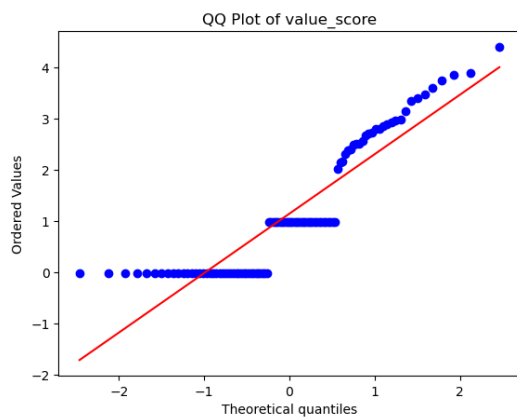
## 6.2. Multi Target Search: E5

### Data Distribution

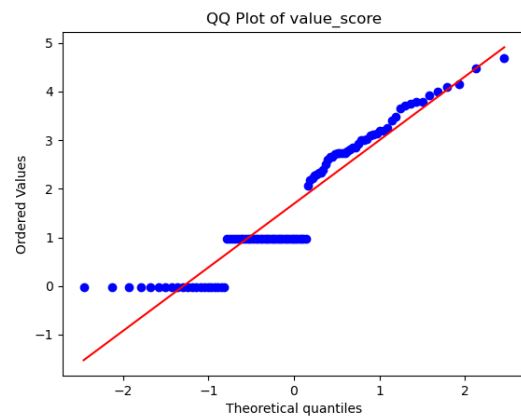
To assess the normality of the data, the Shapiro-Wilk test was employed. This analysis focused on the value scores under two different conditions: variations in transmitter positioning and agent initialization. The results are tabulated in Table 6.3, and the corresponding data distributions are visually represented through Quantile-Quantile (QQ) plots.

**Table 6.3:** Shapiro-Wilk Test Results and QQ Plot References

Metric	Test Statistic	$p$ -value	QQ Plot Reference
value score (transmitter variation)	0.8951	$8.35 \times 10^{-7}$	Figure 6.4
value score (initiation variation)	0.8245	$1.50 \times 10^{-9}$	Figure 6.3



**Figure 6.3:** QQ Plot for value score with initiation variation.



**Figure 6.4:** QQ Plot for value score with transmitter variation.

Both metrics, value score for transmitter and initiation variations, show  $p$ -values significantly lower than

0.05. The QQ plots seem to be a bit off, this has to do with the time-cap of 4000 for unsuccessful runs, which are also reflected in the value score. The lower and middle points coincide with the unsuccessful runs with consistent values. This result, combined with the QQ plots, suggests non-normal distribution of the data in both scenarios.

### Coefficient of Variation

The Coefficient of Variation (CV) was analyzed to determine the stability and reliability of the model output across 100 simulation runs. The CV, which is the ratio of the standard deviation to the mean, provides insights into the variability relative to the mean of the dataset. The CV plots for both scenarios are depicted below.

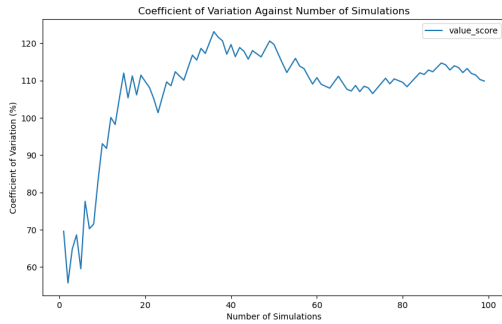


Figure 6.5: CV Plot for varying agent initialization.

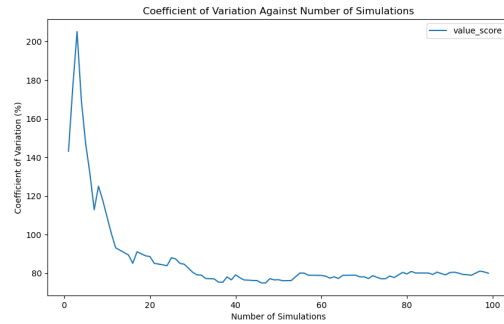


Figure 6.6: CV Plot for varying transmitter positioning.

Both CV plots depicted in [Figure 6.5](#) and [Figure 6.6](#) seem to stabilize, although the latter more than the first one.

### Statistical Test

Statistical significance of the comparison between the utility-based [MUC](#) and the benchmarking [MCTS](#) was evaluated using the non-parametric paired Wilcoxon Signed-Rank Test, which has been previously described in [section 6.1](#). The comparative analysis, encompassing 100 runs for each scenario, investigated the effects of variations in agent initialization and transmitter positioning. The results are presented in [Table 6.4](#).

Table 6.4: Analysis of agent initialization and Transmitter Positioning

Scenario	Stat	P-value	Effect Size	M1	M2
Varying Agent initialization	1.0	$4.00 \times 10^{-18}$	0.9998	0.98	4.2
Varying Transmitter Positioning	3.0	$4.25 \times 10^{-18}$	0.9994	0.98	4.25

In the varying agent initialization scenario, the test statistic of 1.0, combined with a p-value of  $4.00 \times 10^{-18}$ , indicates an almost uniform direction of effect across the dataset. The near-perfect effect size of 0.9998 corroborates the profound impact of swarm initiation variation on value scores.

For the varying transmitter positioning, the test statistic of 3.0 suggests a strong but slightly less uniform effect than in the swarm initiation case. The p-value of  $4.25 \times 10^{-18}$  and an effect size of 0.9994 still indicate a significant and substantial effect on value scores due to transmitter position variations.

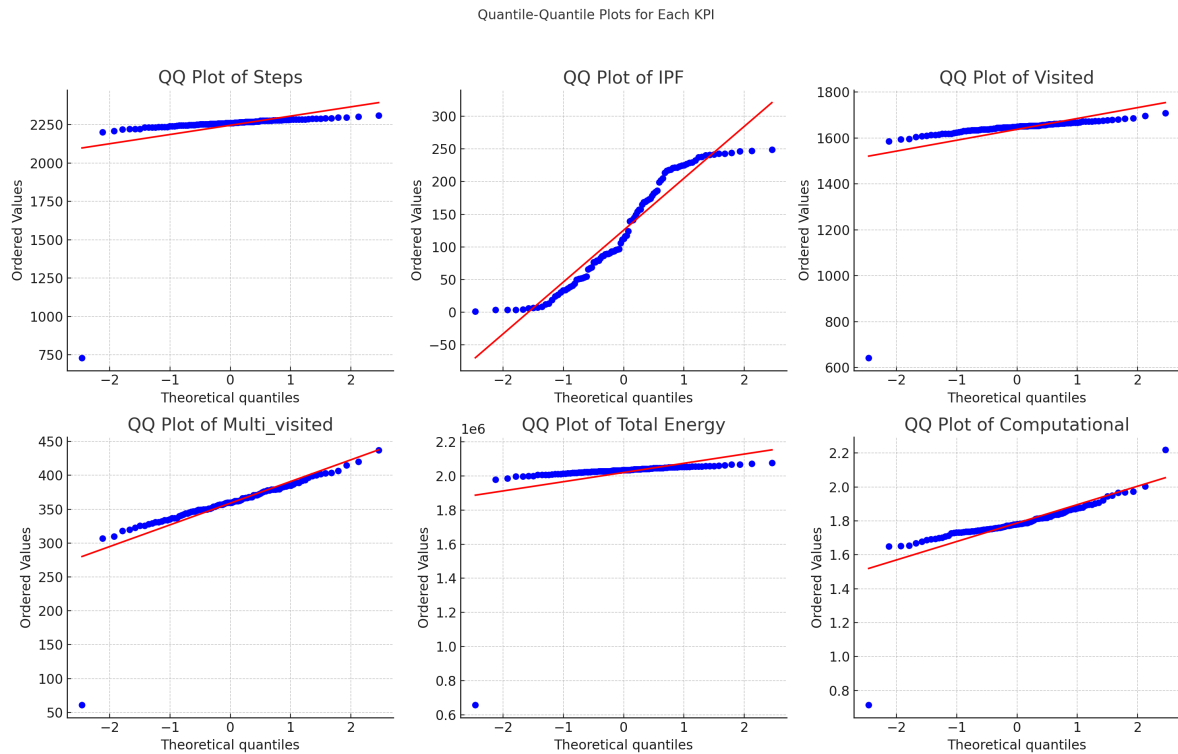
## 6.3. Communication Line Mapping: E6

### Data Distribution

Statistical quantification for each Key Performance Indicator (KPI) was carried out using the Shapiro-Wilk test for normality. The results are summarized in [Table 6.5](#) and further corroborated by their respective Quantile-Quantile (QQ) plots.

**Table 6.5:** Results of Shapiro-Wilk Test and QQ Plot References

KPI	Test Statistic	$p$ -value	QQ Plot Reference
Steps	0.1615	$3.41 \times 10^{-21}$	Figure 6.7 Top Left
IPF	0.9223	$2.05 \times 10^{-5}$	Figure 6.7 Top Middle
Visited	0.2224	$1.64 \times 10^{-20}$	Figure 6.7 Top Left
Multi_visited	0.6736	$1.62 \times 10^{-13}$	Figure 6.7 Bottom Left
Total Energy	0.1615	$3.41 \times 10^{-21}$	Figure 6.7 Bottom Middle
Computational	0.6221	$1.38 \times 10^{-14}$	Figure 6.7 Bottom Right

**Figure 6.7:** Quantile-Quantile Plots for each KPI

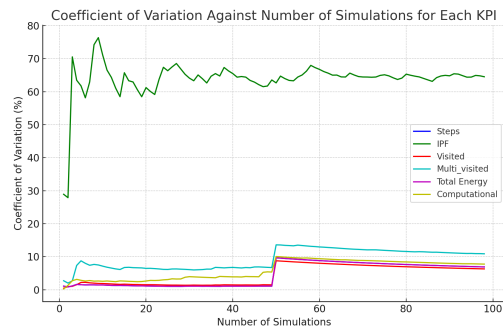
All KPIs have  $p$ -values less than 0.05, which means the data for these KPIs can not be assumed to be normally distributed. This conclusion is backed up by the QQ plots, not aligning well with the red line.

### Coefficient of Variation

The reliability of these findings is further assessed by examining the coefficient of variation (CV) for each KPI. The CV is calculated as:

$$CV = \left( \frac{\text{Standard Deviation}}{\text{Mean}} \right) \times 100$$

The line graph in [Figure 6.8](#) shows the CV against the number of simulations for each KPI. All CV values tend to stabilize over the number of simulations taken.



**Figure 6.8:** Coefficient of Variation Against Number of Simulations for each KPI

Given the results from the Shapiro-Wilk tests and the QQ plots, a conservative approach was taken and the data was not assumed to be normally distributed. This guides the choice towards non-parametric statistical tests for further analysis.

### Statistical Test

Statistical robustness of the mapping algorithms was evaluated using the non-parametric paired Wilcoxon Signed-Rank Test, which has been previously described in [section 6.1](#).

**Table 6.6:** Wilcoxon Signed-Rank Test on Boustrophedon (1) and Spiral mapping (2)

KPI	Stat	P-value	Effect Size	M1	M2
Steps	0	$1.78 \times 10^{-17}$ *	1.0	2261	3113.5
IPF	1975	0.1971	0.5758	111.82	97.69
Visited Cells	0	$1.78 \times 10^{-17}$ *	1.0	1649	2189
Multi-Visited Cells	0	$1.78 \times 10^{-17}$ *	1.0	360	437
Total Energy	0	$1.78 \times 10^{-17}$ *	1.0	2034000	2779200
Computational Time	10	$2.44 \times 10^{-17}$ *	0.9979	1.7791	2.2239

\* denotes  $p < 0.05$ , indicating statistical significance.

[Table 6.6](#) provides the Wilcoxon Signed-Rank Test statistics for comparing the Boustrophedon and Spiral mapping algorithms. The results show that Boustrophedon and Spiral have statistically significant differences in Steps, Visited Cells, Multi-Visited Cells, Total Energy, and Computational Time with large effect sizes of 1.0 or close to 1.0, indicating that the differences in these KPIs are not only statistically significant but also of high practical significance. For the IPF KPI, while the p-value does not indicate a statistically significant difference, the effect size suggests a moderate practical difference in performance. These results highlight Boustrophedon's greater efficiency in terms of mapping density and coverage, despite a larger number of time steps required.

**Table 6.7:** Wilcoxon Signed-Rank Test on Boustrophedon (1) and tab-QL mapping (2)

KPI	Stat	P-value	Effect Size	M1	M2
Steps	822.0	0.6344	0.5356	2261.0	2228.0
IPF	49.0	$2.79 \times 10^{-10}$ *	0.9723	116.29	13.47
Visited Cells	3.0	$2.79 \times 10^{-11}$ *	0.9983	1648.0	1399.0
Multi-Visited Cells	0.0	$2.38 \times 10^{-11}$ *	1.0	360.0	231.0
Total Energy	665.0	0.0968	0.6243	$2.034 \times 10^6$	$1.941 \times 10^6$
Computational Time	0.0	$2.39 \times 10^{-11}$ *	1.0	1.78	37.85

From [Table 6.7](#), we observe that the Boustrophedon (1) and tab-QL (2) mapping algorithms exhibit

statistically significant differences in several KPIs, especially IPF, Visited Cells, and Multi-Visited Cells. While the p-values for Steps and Total Energy suggest no significant difference, the effect sizes reveal a moderate to large practical difference in performance between the two algorithms. The large effect sizes for IPF, Visited Cells, Multi-Visited Cells, and Computational Time (all close to 1.0) indicate that one algorithm consistently outperforms the other across the majority of runs.

**Table 6.8:** Wilcoxon Signed-Rank Test on Spiral (1) and tab-QL mapping (2)

KPI	Stat	P-value	Effect Size	M1	M2
Steps	32.0	$1.21 \times 10^{-10}$ *	0.9819	3118.0	2262.0
IPF	0.0	$2.39 \times 10^{-11}$ *	1.0	100.56	13.47
Visited Cells	0.0	$2.39 \times 10^{-11}$ *	1.0	2188.0	1404.0
Multi-Visited Cells	0.0	$2.39 \times 10^{-11}$ *	1.0	438.0	231.0
Total Energy	1.0	$2.52 \times 10^{-11}$ *	0.9994	$2.779 \times 10^6$	$1.953 \times 10^6$
Computational Time	0.0	$2.39 \times 10^{-11}$ *	1.0	2.209	38.18

**Table 6.8** presents a comparison between the Spiral (1) and tab-QL (2) mapping algorithms using the Wilcoxon Signed-Rank Test. The test results reveal significant differences in all KPIs, most notably in the number of steps and total energy consumed during mapping. The effect sizes, close to or equal to 1.0, indicate that the differences are not just statistically significant but also have a substantial practical impact, demonstrating consistent superiority of one algorithm over the other across all runs.

# 7

## Supplementary Experiments & Analysis

This appendix presents an overview of the supplementary experiments and analyses. To better understand the methodologies and their effectiveness, various analyses were performed, which are detailed in [Table 7.1](#). This table provides details of each experiment, categorizing them by ID, the type of experiment conducted, the specific property being examined, the method used, and the experimental setup. The experiments included examining the sensitivity of algorithm parameters, a deeper dive into the utility function in for coalition formation in Multi Transmitter Search.

**Table 7.1:** Overview of Supplementary Experiments and Analyses

ID	Experiment Type	Property	Method	Setup
Se1	Sensitivity Analysis	Search Area $20km^2$	<a href="#">PSO</a> Pair-gradient	Modified Base Setup
Se2	Sensitivity Analysis	Sobol Initialization	<a href="#">PSO</a> Pair-gradient	Base Setup
Se3	Sensitivity Analysis	Agent Count Variations	<a href="#">PSO</a> Pair-gradient	Modified Base Setup
Se4	Sensitivity Analysis	Coefficients Adjustment	<a href="#">PSO</a>	Base Setup
Se5	Sensitivity Analysis	Follower Offset and Velocity	Pair-gradient	Base Setup
Se6	Sensitivity Analysis	Helper Agent Offset	Boustrophedon	Base Setup
Se7	Sensitivity Analysis	Turn Angle Variation	Boustrophedon	Base Setup
Se8	Sensitivity Analysis	Agent Velocity	Boustrophedon, Spiral	Base Setup
Se9	Sensitivity Analysis	Shrink Factor	Spiral	Base Setup
Se10	Sensitivity Analysis	Border Angle Variation	Spiral	Base Setup
Se11	Sensitivity Analysis	Pre-train	Tab-QL	Simple Triangle Grid
Se12	Sensitivity Analysis	EW Train	Tab-QL	Base Setup
Ma1	Methodological Analysis	Utility Function Refinement	MUC	Global Setup
Ma2	Methodological Analysis	Metric Exploration for <a href="#">PSO</a> subswarming	MUC	Global Setup

### 7.1. Single Transmitter Search

This section presents supplementary experiments and a sensitivity analysis for both the Particle Swarm Optimisation (PSO) and Pair-based Gradient Search algorithms. The aim is to investigate their performance and robustness across varied configurations.

The configurations are derived from analyses in the results, where PSO exhibited potential weaknesses such as high energy consumption and occasional non-converging solutions. For the Pair-based Gradient Search, where it showcased certain advantages like energy efficiency, it lagged in terms of speed in reaching the RSSI Threshold (RT). This delay is attributed to its intrinsic pairing mechanism, which could be influencing the overall gradient determination and hence the search efficacy. Additional runs experiments are done to investigate what drives the gradient determination. The experiments have the following alternations:

- **Se1: Search Area  $20km^2$ :** This setting increases the search area to explore if a larger area affects the step time to RT or the rate of unsuccessful searches.
- **Se2: Sobol initialisation:** Due to the noted issue of non-convergence in the base setting, this configuration employs Sobol sequences for initial positioning of agents to provide a more uniform distribution. This technique is explained in [subsection 3.2.2](#)

- **Se3: 5 Agents:** The number of agents is reduced to check how this affects energy consumption and non-convergence rates.
- **Se3: 10 Agents:** The swarm size is increased to see if collective capabilities affect the time to reach the RT and trajectory density.
- **Se4:  $c1(2.8), c2(1.3), w(1)$  (PSO):** Taking cues from the study by A. Carlisle and G. Dozier [3], these coefficients are modified to refine the agents' search behavior in the PSO algorithm. The revised coefficients aim to balance better convergence and exploration by generating higher velocities for agents, particularly in scenarios with diminished social, cognitive, and inertia influences, which may assist the swarm in evading premature convergence.
- **Se5: 35 m Follower offset (Pair-Gradient):** By adjusting the follower agents' offset to 35 m, this configuration aims to explore the offset's role in gradient determination. The relative positioning between the leader and follower, affected by the offset, is important for effective triangulation, which in turn plays a critical role in gradient ascertainment and impacts the overall search pattern and efficiency.
- **Se5: 30 m/s Follower velocity (Pair-Gradient):** Altering the velocity parameter of the follower agents to 30 m/s is intended to analyze the impact of speed on gradient determination. The relative motion between the leader and follower, dictated by this velocity, influences their triangulation capability. This could further affect the gradient's orientation and magnitude, altering the search dynamics.

**Table 7.2:** Mean Value KPIs for Sensitivity Analysis in Particle Swarm Optimisation

Algorithm	# Runs	Step time to RT	Mean distance to RT	Trajectory Coverage	Trajectory Density	Swarm Energy usage	Compute	Unsuccessful searches
Base PSO	88	751	1187	$3.57 \times 10^7$	$5.93 \times 10^{-8}$	936643	0.0695	0.12
Search Area $20km^2$	78	1668	1686.6	$1.42 \times 10^8$ *	$2.04 \times 10^{-8}$ *	2044510	0.164	0.22
Sobol initialisation	100	659	1413.8	$5.06 \times 10^7$	$3.44 \times 10^{-8}$	899800	0.0611	0*
5 Agents	70	552	904.1	$2.44 \times 10^7$	$8.62 \times 10^{-8}$	646771*	0.0538	0.3
10 Agents	100	1421	1268.6	$4.22 \times 10^7$	$8.26 \times 10^{-7}$	1138177	0.1304	0.06
$c1(2.8), c2(1.3), w(1)$	100	525*	1715.0*	$3.47 \times 10^7$	$4.45 \times 10^{-8}$	848171	0.0526*	0*

Upon a detailed analysis of the results presented in [Table 7.2](#), several key observations are described regarding the performance of Particle Swarm Optimisation under different configurations:

Starting with the Base PSO, a step time to RT of 751 and an energy usage of 936643 is reported. When the search area is increased to  $20km^2$ , both the step time to RT and energy usage surge to 1668 steps and  $2044510 (m/s)^2$  respectively. This increase indicates that a larger search area, potentially improving coverage, also implies slower and more resource-intensive performance.

Sobol initialisation, described in [subsection 3.2.2](#), impacts the performance significantly. It entirely eliminates unsuccessful searches, leading to a 0 %. Additionally, energy consumption drops to 899800  $(m/s)^2$ , less than the Base PSO. These results advocate for the efficiency of Sobol initialisation and underscore the importance of robust initialization strategies.

Regarding agent count, when reduced to 5 agents, the step time to RT is at its quickest, recorded at 552 steps. Yet, this faster performance comes at the expense of a higher rate of unsuccessful searches, which increases to 30 %. Such a short step time is a consequence of the reduced number of agents.

Conversely, a configuration using 10 agents results in a step time to RT of 1422 steps, nearly three times longer than with 5 agents. However, the rate of unsuccessful searches drops significantly to 6 %. This data suggests a trade-off between speed and reliability, where a larger swarm may lead to more consistent results but not necessarily faster outcomes.

When custom coefficients  $c1(2.8), c2(1.3), w(1)$  are applied, the algorithm achieves an optimal step time to RT at 525 steps, with no unsuccessful searches recorded. At the same time, the mean distance to RT increases to 1715 steps. This configuration appears to balance search speed, convergence quality, and agent distribution, possibly due to the enhanced cognitive component.

In summary, varying the PSO’s configurations presents both advantages and challenges. Recognizing and tailoring these configurations is crucial for subsequent attempts to improve the algorithm.

**Table 7.3:** Mean Value KPIs for Sensitivity Analysis in Pair-based Gradient Search

Algorithm	Step time to RT	Mean distance to RT	dis- to Trajectory Coverage	Trajectory Density	Swarm Energy usage	Compute	Unsuccessful searches
Base	1218	2055	34678424	$5.75 \times 10^{-8}$	658970	0.135	0
Search Area $20km^2$	2643	3733*	137298858*	$1.75 \times 10^{-8}$ *	1438729	0.309	0
Sobol initialisation	1312	2033	50908040	$4.16 \times 10^{-8}$	678783	0.145	0
5 Agents	965*	2034	22024909	$9.40 \times 10^{-8}$	422905*	0.105*	0
10 Agents	1376	2149	42369537	$4.69 \times 10^{-8}$	744577	0.159	0
35 m Follower offset	1399	2059	34725447	$6.23 \times 10^{-8}$	772544	0.165	0
30 m/s Follower velocity	1969	2131	37642040	$4.87 \times 10^{-8}$	585448	0.226	0

Upon analyzing the results displayed in [Table 7.3](#), several insights are revealed about the Pair-based Gradient Search’s behavior under varied configurations:

The Base configuration shows a step time to RT of 1218 steps with an energy consumption of 658970. Expanding the search area to  $20km^2$  causes the step time to RT and energy usage to rise to 2643 and 1438729 respectively. This sharp increase implies that although a broader search area might offer better coverage, it demands more resources and takes longer.

The introduction of Sobol initialisation results in a step time to RT of 1312 steps, which is slightly higher than the Base. Energy consumption is also marginally increased to 678783  $(m/s)^2$ . These findings contrast the earlier hypothesis and imply that while Sobol initialisation might offer benefits in other scenarios or metrics, its contribution on step time to RT and energy usage in this context is negative. This can be explained by the pairing which might take longer when agents are distributed well over the area.

Changing the agent count yields intriguing outcomes. With only 5 agents, the step time to RT decreases to 965 steps, the lowest among all configurations. Conversely, using 10 agents takes the step time to RT to 1376 steps, while also seeing an energy increase. This illustrates the inherent trade-offs between search speed, total energy consumption, and swarm size.

Configurations involving follower adjustments, both in offset and velocity, display nuanced impacts on the gradient determination. With a 35 m follower offset, the step time to RT stands at 1399 steps, and energy usage is slightly higher than the base, emphasizing the offset’s role in effective triangulation. On the other hand, a 30 m/s follower velocity leads to a prolonged step time to RT at 1969 steps, showcasing the interplay between speed, triangulation, and gradient orientation.

In conclusion, adjusting configurations in Pair-based Gradient Search introduces a series of performance trade-offs. Understanding and fine-tuning these parameters is important for optimizing the search algorithm’s efficiency and efficacy.

## 7.2. Multi Transmitter Search

### 7.2.1. Ma1: Refinement of Utility-function

Reflecting back on the results, a particular decision involving the switching of agents from the yellow frequency to blue and orange was identified as suboptimal. This early decision impacted the ability to discover one of the yellow transmitters. It was observed that in successful MCTS runs, retaining an outlier agent within the initial coalition increased search effectivity.

To address this, an additional experiment was conducted that introduced a penalty in the utility function if an outer agent within a cluster attempted a frequency switch. The hypothesis was that such a penalty might prevent outer agents from prematurely departing their clusters, which in turn, would improve the search outcomes on the outer edges of the search area.

The element added to the utility function can be expressed as follows:



$$\Delta\text{Utility} = -(\text{Penalty}_{\text{switch}} \times N_{\text{outer agents switching in cluster}})$$

The outcomes after implementing this revised utility function were as follows:

**Table 7.4:** Experimental Results with Revised Utility Function

Run	Transmitters Found	Total Steps	Stationary	Alternations	Value Score	Energy Used
0 (Modified Utility)	6	3220	25	7	2.78	3188242

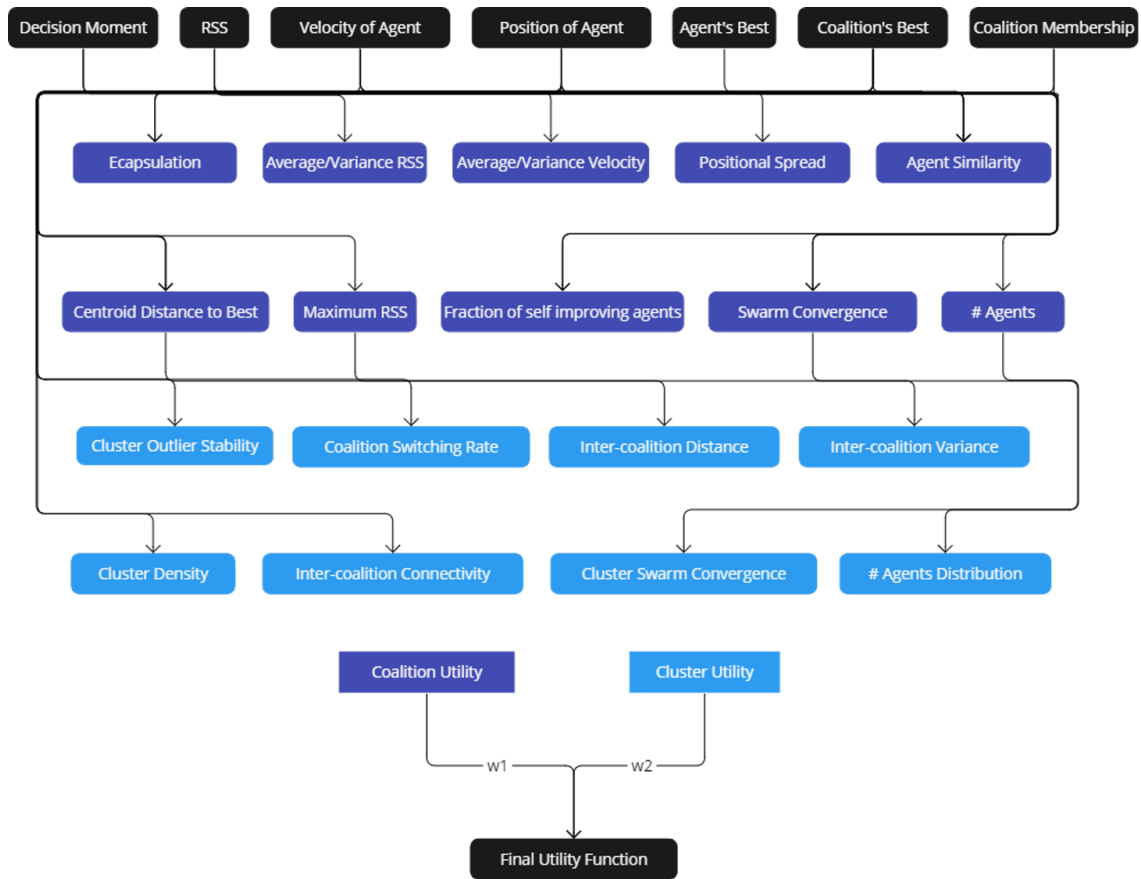
The experiment output indicates all six transmitters were located, now with a reduced number of steps. This strategy successfully convinced outer agents to stay stationary, decreasing unnecessary changes, and ultimately enhancing the value score and energy usage by the swarm.

### 7.2.2. Ma2: Further Metric Exploration

In examining the varied outcomes of other initialisations and understanding why some were not effective in their missions, the method of the HURMS framework can be used, specifically transitioning from block 6 to 2b and repeating the evaluation process. To facilitate the refinement of the utility function in the MUC approach, a qualitative analysis was done to point out potential metrics that could prove beneficial for further optimization of the utility function.

Figure 7.1 presents the assessment of potential metrics deemed valuable for optimizing subswarming behavior within Particle Swarm Optimization (PSO) contexts. The analysis delves into agent-specific properties and develops coalition-based metrics, which can serve as the groundwork for constructing cluster-based metrics. This structured framework aids the second step (2b) within the HURMS framework.

Once a metric's desired impact is established, the subsequent phase involves integrating both coalition and cluster-based metrics, shaping a composite utility function. This integration forms is described by block 2d in the framework, which the current exploration aims to enhance. The development of these metrics and their incorporation into the utility function aim to provide clear insights into the outcomes of specific metric implementation. Furthermore, this process is intended to inform tactical modifications to the MUC approach, potentially paving the way for more consistently successful simulations.



**Figure 7.1:** A qualitative analysis of potential valuable metrics for subswarming with PSO

### 7.2.3. Comparative Results: E5 - exclusion unsuccessful runs

To accurately compare the speed of mission completion between the two methods, it is necessary to exclude the **MUC** runs where 4 to 5 transmitters were located. This is due to the 4000-time cap on unsuccessful runs, which could otherwise distort the analysis. After removing these runs, 44% of the runs with varying transmitter positions and 29% with varying agent initialization remain, as detailed in **Table 7.5**. This filtered data indicates that the MCTS method is about 45% faster than the MUC method for swarm initialization variation, and around 46% faster for transmitter position variation in successful runs.

**Table 7.5:** Mean Performance Overview of **MUC** and **MCTS** for various agent initializations and transmitter positioning (successful runs: 29 & 44)

Experiment	Transmitters Found	Total Steps	Stationary	Alternations	Value Score
Agent Initialisation: MUC (29 Runs)	6.00	3086	23.41	7.14	2.91
Agent Initialisation: MCTS (29 Runs)	6.00	1658	0.93	15.17	4.34
Transmitter Initialisation: MUC (44 Runs)	6.00	2924	22.61	6.25	3.08
Transmitter Initialisation: MCTS (44 Runs)	6.00	1606	0.80	14.80	4.39

## 7.3. Communication Line Mapping & Interference Point Finding

Following the insights gathered from the results, it is evident that while each algorithm possesses distinct strengths, they also showcase areas for potential improvement. The supplementary experiments aim to investigate the effects of certain parameter adjustments on the overall performance and efficiency of the algorithms. By running these experiments, it is anticipated that a deeper understanding of the algorithms' behaviors under different configurations will be achieved, providing a foundation

for optimizing their efficiency and accuracy. The experiments have the following adjustments to the experimental setup of the main results:

- **Se6: Helper agent offset 1.25 km (Boustrophedon):** Introducing an offset between agents might enable broader area coverage in parallel, possibly leading to improved efficiency in mapping or searching.
- **Se7: Turn angle variation +2 degrees (Boustrophedon):** Altering the turning angle by 2 degrees aims to determine how wider angles influence the agent’s traversal pattern. Such adjustments might affect both the thoroughness of exploration and the density of the mapping.
- **Se8: Agent velocity 20 m/s (Boustrophedon):** By setting the agent’s speed to 20 m/s, this variation examines the balance between swift area coverage and data quality. A boost in velocity could speed up exploration, but might diminish mapping detail due to reduced dwell time in areas.
- **Se9: Shrink Factor 0.95 (Spiral):** Adjusting the spiral’s shrink rate to 0.95 tests the impact of a tighter spiral on the mapping’s density. A denser spiral might improve mapping granularity but at the potential cost of reduced overall coverage.
- **Se10: Border Angle variation +5 degrees (Spiral):** Modifying the approach angle to boundaries by 5 degrees seeks to discern its effect on agent-border interactions. This change could alter boundary navigation behaviors, possibly resulting in less efficient edge data capture.
- **Se8: Agent velocity 20 m/s (Spiral):** Altering the agent’s velocity in the spiral pattern offers insights into how speed variations can influence spiral traversal, potentially affecting the mapping’s resolution and efficiency.
- **Se11: Pre-train (Tab-QL):** This experiment confines agent training to a smaller environment for 200,000 episodes, aiming to gauge if focused pre-training optimizes the agent’s capability in more intricate environments. Training parameters are kept constant. After training, the agents are deployed in the EW environment.
- **Se12: EW train (Tab-QL):** This experiment confines agent training to the expansive EW environment for 30,000 episodes. Training parameters are kept constant. After training, the agents are deployed in the EW environment.

**Table 7.6:** Mean Value KPIs for Sensitivity Analysis in Boustrophedon Mapping

Variation	Steps	IPF error	Visited Cells	Multi-Visited Cells	Total Energy	Computational Time
Base	2246	126	1638	359	2020255	1.79
Helper agent offset 1.25 km	2245	121	1637	356	2019456	1.65
Turn angle variation +2°	1580	126	1245	158	1420882	1.25
Agent velocity 20 m/s	3322	110	2436	658	1341993	2.67

Upon analyzing the results outlined in [Table 7.6](#), several crucial observations are documented regarding the performance of the Boustrophedon Mapping under distinct configurations:

Commencing with the Base configuration, the system executes 2246 steps, consumes an energy of 2020255, and demands a computational time of 1.79. However, the introduction of a helper agent with an offset of 1.25 km demonstrates minimal variations from the base model, showing a slight reduction in steps to 2245, a marginal decrease in energy to 2019456, and a reduced computational time of 1.65. These results suggest that the helper agent’s offset facilitates a more efficient energy consumption and computational time, albeit with a nearly identical step count.

Incorporating a turn angle variation of +2° shows a significant change in behavior. The steps decrease markedly to 1580, and the energy consumption drops to 1420882. Additionally, the computational time shortens to 1.25. Importantly, both the visited cells reduce to 1245, and the multi-visited cells decline to 158, indicating enhanced exploration efficiency. These findings imply that by tweaking the turn angle, the agent can potentially refine its movement pattern, resulting in fewer steps, reduced revisits to cells, and lower energy use.

When the agent's velocity is increased to 20 m/s, there is a noticeable change. The agent undertakes more steps, totaling 3322, and visits more cells, with 2436 visited cells. However, the energy consumption decreases to 1341993. On the flip side, the computational time rises to 2.67, indicating that the increased speed might result in extended computation times.

In summary, adjusting parameters in the Boustrophedon Mapping method introduces varying outcomes. These might lead to either improved efficiency or heightened computational requirements. This emphasizes the importance of careful parameter selection to meet specific mapping goals.

**Table 7.7:** Mean Value KPIs for Sensitivity Analysis in Spiral Mapping

Variation	Steps	IPF error	Visited Cells	Multi-Visited Cells	Total Energy	Computational Time
Base	3118	110	2185	435	2778077	2.35
Helper agent offset 1.25 km	3119	112	2183	435	2779096	2.27
Shrink Factor 0.92	3814	114	2475	663	3404213	2.73
Agent velocity 20 m/s	4442	91	2949	698	1893995	3.17

Upon examining the results presented in [Table 7.7](#), several noteworthy observations are made regarding the performance of the Spiral Mapping under various configurations:

Starting with the Base configuration, the system performs 3118 steps, consumes an energy of 2778077, and requires a computational time of 2.35. Introducing the helper agent offset of 1.25 km results in almost identical outcomes compared to the base. The system completes 3119 steps, has a marginally increased IPF error of 112, and shows a slight increase in energy consumption to 2779096. However, the computational time is somewhat reduced to 2.27. These findings suggest that the helper agent's offset may slightly influence the system's efficiency, but overall, the impact is minimal. A visual analysis posits that the predominant limiting factor is inherent to the helper agent itself, and a lateral shift does not substantially mitigate this constraint.

Implementing a shrink factor of 0.92 yields a considerable change in the system's performance. The total steps increase to 3814, and there is a rise in the number of visited cells to 2475. Furthermore, energy consumption increases to 3404213, and the computational time extends to 2.73. These results indicate that by adjusting the spiral's shrink factor, the system might increase its mapping density, but this comes at the cost of higher energy consumption and computational time.

Adjusting the agent's velocity to 20 m/s reveals significant changes in the system's dynamics. There is a notable increase in steps to 4442 and visited cells rise to 2949. Interestingly, even with the higher step count, the energy consumed drops significantly to 1893995. However, the computational time grows to 3.17. These figures suggest that the agents map more densely, and become more energy-efficient but demands a longer computational duration.

In summary, the Spiral Mapping technique's performance can vary widely based on the selected configurations. These insights emphasize the importance of careful parameter selection to attain the desired mapping goals while maintaining efficiency.

**Table 7.8:** Mean Value KPIs for Sensitivity Analysis in Tabular Q-Learning

Variant	Steps	IPF error	Visited Cells	Multi-Visited Cells	Total Energy	Computational Time
EW-train	918	86	461	36	704950	42
Pre-train	2268	13	1292	221	1912082	38
EW + Pre-train	2346	14	1327	221	1951424	41

The data presented in [Table 7.8](#) shows the effects of different training scenarios on the performance of Tabular Q-Learning, specifically looking at the number of steps and the count of visited cells as indicators of the system's thoroughness in mapping an area.

In the "EW-train" case, the system takes 918 steps and visits 461 cells, setting a baseline for comparison. This indicates the system's basic mapping capability without further training enhancements.

For the "Pre-train" configuration, there is a noteworthy increase in activity. Steps rise to 2268, which is expected as more steps can lead to more thorough mapping. Correspondingly, the visited cells increase

significantly to 1292. This nearly triples the mapping coverage in comparison to the "EW-train" and suggests that the system is more effectively exploring its environment. The increase in energy to 1912082 shows a higher demand, but it is compensated by the greater number of visited cells. Additionally, this configuration is more efficient in terms of computational time, decreasing to 38 seconds.

When combining "EW" and pre-training ("EW + Pre-train"), steps increase slightly to 2346, with visited cells also rising to 1327. The energy usage is somewhat higher at 1951424, and computational time is a bit longer at 41 seconds. These figures imply that while the combination leads to more explored cells, the improvements are marginal compared to pre-training alone.

In essence, pre-training markedly increases the thoroughness of mapping, as reflected by the higher visited cell count, without substantially increasing the computational time. The step count is also higher, but this is not a downside since more steps can contribute to more comprehensive mapping. The sequential approach of the training was to first apply "EW-train," then "Pre-train," and lastly to combine both, indicating an incremental strategy to enhance mapping effectiveness.

# 8

## Pseudocode

This appendix provides a detailed examination of the methodologies, algorithms, and procedural steps for the Unmanned Aerial Vehicle (UAV) swarm operations.

---

**Algorithm 1** High Level Model Architecture

---

```
1: procedure main
2:   Initialize search area (SA), resolution, and visualization receiver locations
3:   Generate signal data based on visualization receiver locations
4:   Create a AgentSwarm instance with a given band frequency
5:   Create a Animation instance using the agent swarm and signal data
6:   Start the animation
7: end procedure
8: procedure AgentSwarm(band)
9:   Initialize agents (each with a random position, velocity, band)
10:  Define a method to calculate and store the global best position and its signal strength (based
    on the fitness of each agent)
11:  Define a method to update all agents' positions and velocities based on algorithmic rules (con-
    sidering personal best, global best)
12: end procedure
13: procedure Agent(position, velocity, band)
14:  Store initial position, velocity, and band
15:  Define a method to calculate the Signal of Interest (SOI) at the agent's current position
16:  Define a method to update the agent's position and velocity based on algorithmic rules
17:  Define methods to handle different states of the agent:
18:  if initialize then
19:    The agent has random movement to search for signal
20:  else if search then
21:    The agent moves towards stronger signal (uses algorithmic rules)
22:  else if circle then
23:    The agent circles around the detected signal source
24:  end if
25: end procedure
26: procedure Animation(swarm, x, y, z)
27:  Initialize the plot with initial agent positions and signal data
28:  Define a method to update the plot for each frame (move the agents, update the color according
    to signal strength)
29:  Define a method to create and start the animation
30: end procedure
```

---

---

**Algorithm 2** PSO Algorithm applied for UAVs [\[7\]](#)

---

- 1: Initialize swarm (UAVs) with random positions and velocities
  - 2: Assess the signal strength for each UAV
  - 3: Assign each UAV's initial position as its personal best
  - 4: Designate the strongest initial signal as the global best position
  - 5: **while** termination criteria are unsatisfied **do**
  - 6:     **for** each UAV  $i$  **do**
  - 7:         Recalculate UAV velocity based on:  $v_i^{(t+1)} = wv_i^{(t)} + c_1r_1(p_i^{(t)} - x_i^{(t)}) + c_2r_2(g^{(t)} - x_i^{(t)})$
  - 8:         Adjust UAV position using:  $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$
  - 9:         Re-evaluate signal strength at the new position
  - 10:        **if** new signal strength is superior to  $p_i$  **then**
  - 11:            Update the UAV's personal best position
  - 12:        **end if**
  - 13:     **end for**
  - 14:     Adjust the global best position based on the strongest signal among all UAVs
  - 15: **end while**
-

**Algorithm 3** Pair-based Gradient Search

---

```

1: procedure grad_search
2:   find_partner()
3:   if agent is follower then
4:     if leader hasn't moved then
5:       Move follower towards  $P_2$ 
6:     else
7:       Compute trajectory between  $P_{1_{last}}$  and  $P_1$ 
8:       Position follower at a  $90^\circ$  offset and  $x$  meters from this trajectory
9:     end if
10:  else if agent is leader then
11:    if  $P_2$  is near  $P_1$  then
12:      Update leader's velocity using  $\text{grad\_velocity}(P_1, P_{1_{last}}, P_2)$ 
13:    else
14:      Keep leader stationary until follower is near
15:    end if
16:  else
17:    Keep agent stationary ▷ Agent without a partner
18:  end if
19: end procedure
20: procedure find_partner
21:   List agents without partners (excluding the current agent)
22:   if current agent is unpaired then
23:     if list is empty then
24:       Exit
25:     else
26:       Pair current agent with the nearest in list
27:       Establish roles as leader or follower based on  $RSS_1$  and  $RSS_2$ 
28:     end if
29:   end if
30: end procedure
31: procedure grad_velocity( $P_1, P_{1_{last}}, P_2$ )
32:   Measure signal strengths:  $RSS_1$ ,  $RSS_2$ , and between  $P_1$  and  $P_{1_{last}}$ 
33:   Determine gradient using these  $RSS$  values
34:   Calculate velocity direction as average gradient, pointing towards the signal source
35:   Adjust the velocity's magnitude
36:   Return the new velocity
37: end procedure

```

---

**Algorithm 4** Calculate Desired Velocity for Circling a Source

---

```

1: procedure circle_velocity( $\mathbf{p}$ ,  $\mathbf{p}_{last}$ ,  $\mathbf{p}_{pre-last}$ ,  $RSS$ , velocity_multiplier=20, alpha=0.4)
2:   Let positions be [ $\mathbf{p}$ ,  $\mathbf{p}_{last}$ ,  $\mathbf{p}_{pre-last}$ ]
3:    $\mathbf{O} \leftarrow [RSS(\text{pos}) \mid \text{pos} \in \mathbf{positions}]$ 
4:   Calculate gradients using  $\mathbf{O}$  and positions
5:   Compute source direction from gradients
6:   Calculate desired velocity tangential to source direction
7:   Determine previous velocity using  $\mathbf{p}_{last}$  and  $\mathbf{p}_{pre-last}$ 
8:   Compute new and previous velocity directions
9:   Determine average direction, avg_dir
10:  Compute new velocity as avg_dir  $\times$  Norm(desired_velocity)
11:  return new velocity
12: end procedure

```

---



**Algorithm 5** Signal-Based Stop Algorithm

---

```

1: procedure SignalStop
2:   Compute new velocity using circle_velocity procedure
3:   Update current observation  $O_t$  based on position  $\mathbf{p}$ 
4:   Call handle_rx_signal procedure with  $O_t$ 
5: end procedure
6: procedure handle_rx_signal( $O_t$ )
7:   if  $O_t$  is the highest so far then
8:     Update  $\mathbf{I}_{\max}$ 
9:     Reset  $\mathbf{I}_{\text{count}}$ 
10:  else if  $O_t <$  previous  $O$  then
11:    Increment  $\mathbf{I}_{\text{count}}$ 
12:    Call check_decreasing_signal procedure
13:  else
14:    Reset  $\mathbf{I}_{\text{count}}$ 
15:  end if
16:  Store  $O_t$  for the next timestep
17: end procedure
18: procedure check_decreasing_signal
19:  if  $\mathbf{I}_{\text{count}} \geq 2$  and current  $O_t < 0.5 \mathbf{I}_{\max}$  then
20:    Stop the agent by setting its velocity to zero
21:    Switch to the "mapping" state
22:  end if
23: end procedure

```

---

**Algorithm 6** Generate Sobol Sequence

---

```

1: procedure SobolSequence(dimension, point_number)
2:   Initialize an array  $V$  to store the direction numbers. Each row represents a dimension.
3:   Initialize an array  $X$  to store the intermediate results.
4:   Compute the first direction number for each dimension,  $V[i, 1] = 1$  for  $i = 1$  to dimension.
5:   for  $j = 2$  to point_number do
6:     Compute the gray code of  $j$ ,  $g = j \oplus (j/2)$ .
7:     Compute the position of the rightmost zero bit in  $g$ , represented as  $p$ .
8:     for  $i = 1$  to dimension do
9:       if  $p > \text{length}(V[i])$  then
10:        Calculate new direction numbers  $V[i, p]$  for this dimension.
11:      end if
12:       $X[i, j] = X[i, j - 1] \oplus V[i, p]$ .
13:    end for
14:  end for
15:  Scale  $X$  to the range of  $[0, 1]$  by dividing each number by  $2^{\text{point\_number}}$ .
16:  return  $X$ .
17: end procedure

```

---

---

**Algorithm 7** One Point Departure

---

```

1: procedure initialize_agents(agents, search_area)
2:   Initialize the agents' positions with a random number generator for 2 dimensions.
3:   for each agent in agents do
4:     Set the agent's initial position to a common starting point.
5:     Generate a point from the random number generator scaled to the search area.
6:     Set this point as the agent's target position.
7:     Calculate the direction from the agent's current position to its target.
8:     Normalize this direction and scale it by a velocity constant to determine the agent's initial
     velocity.
9:     if the distance from the agent to its target is less than a specified threshold then
10:      Set the agent's state to "search".
11:     end if
12:   end for
13: end procedure

```

---



---

**Algorithm 8** Frontier Boustrophedon Mapping

---

```

1: procedure VelocityAdjustment(main_mapper, last_velocity)
2:   Compute  $v_{neg} = -last\_velocity$ .
3:   Rotate  $v_{neg}$  by angle  $\theta$ .
4:   Set  $v_{new} = v_{neg}$  as the new velocity for main_mapper.
5: end procedure
6: procedure PositionUpdate(main_mapper, helper_mapper, pre_last_position)
7:   Calculate  $p_{new}$  such that  $\angle(p_{new}, pre\_last\_position) = 90^\circ$  and  $\|p_{new} - pre\_last\_position\| = d$ .
8:   Command helper_mapper to move to  $p_{new}$ .
9: end procedure
10: procedure MappingStart(main_mapper, helper_mapper)
11:   On main_mapper's arrival at  $p_{new}$ , helper_mapper begins mapping.
12:   Use velocity  $v_{current}$  of main_mapper and increment by  $\Delta v$  for mapping.
13: end procedure
14: procedure DataTracking(main_mapper, helper_mapper, mapped_areas)
15:   if main_mapper is near an area in mapped_areas previously covered by helper_mapper then
16:     Stop main_mapper.
17:   end if
18: end procedure
19: procedure Relocation(main_mapper, helper_mapper, interference_point)
20:   Move both main_mapper and helper_mapper towards interference_point.
21: end procedure
22: procedure RTxCondition( $RT_x$ , main_mapper, helper_mapper)
23:   if  $\|RT_x - main\_mapper\|$  or  $\|RT_x - helper\_mapper\|$  is less than  $d_{threshold}$  then
24:     Only one agent continues mapping.
25:   end if
26: end procedure

```

---

**Algorithm 9** Spiral Mapping Algorithm

---

```

1: procedure update_state
2:   Set state to "mapping".
3:   Determine helper target using position_helper_mapper.
4:   Identify  $agent_{closest}$  to helper target.
5:   Assign helper target to  $agent_{closest}$ 's target and set state to 'help_map'.
6:   Add  $agent_{current\_position}$  to spiral borders.
7: end procedure
8: procedure helper_mapping
9:   if State = 'help_map' then
10:    Measure field strength  $s$ .
11:    if  $\Delta s$  (signal change) exceeds  $TR_d$  border then
12:       $leg \leftarrow 0$  and  $helper\_arrived \leftarrow position$ .
13:    else if  $leg = 0$  then
14:       $leg \leftarrow 1$ ; Add position to spiral borders.
15:    else if  $leg = 1$  and  $s > threshold$  then
16:       $leg \leftarrow 2$ ; Add position to spiral borders.
17:    else if  $leg = 2$  then
18:      Rotate velocity by  $75^\circ$ ; Add position to spiral borders.
19:    else if  $leg = 3$  then
20:      Rotate velocity by  $165^\circ$ ; Add position to spiral borders. If position crosses initial line,
     $leg \leftarrow 4$ .
21:    else if  $leg = 4$  then
22:      Set  $velocity \leftarrow 0$ ;  $leg \leftarrow 5$ .
23:    else if  $leg = 5$  then
24:      Navigate to subsequent spiral border point; Condense spiral borders upon arrival.
25:    end if
26:  end if
27: end procedure
28: procedure mapping
29:   if State = 'mapping' then
30:    Measure field strength  $s$ .
31:    if Helper agent's position is set then
32:      Reference  $\leftarrow$  helper position.
33:    if  $leg = 1$  then
34:      Rotate velocity by  $75^\circ$ ; Add position to spiral borders.
35:    else if  $leg = 2$  and  $s = 0$  then
36:      Reverse velocity; Add position to spiral borders.
37:    else if  $leg = 3$  then
38:      Rotate velocity by  $165^\circ$ ; Add position to spiral borders. If  $s > threshold$ ,  $leg \leftarrow 4$ .
39:    else if  $leg = 4$  then
40:      Set  $velocity \leftarrow 0$ ;  $leg \leftarrow 5$ .
41:    else if  $leg = 5$  then
42:      Move to subsequent spiral border point; Reduce spiral borders on arrival.
43:    end if
44:  end if
45: end if
46: end procedure

```

---

---

**Algorithm 10** Training Q-Learning for Grid-Based Agent Mapping
 

---

```

1: procedure TrainGridMapping_Q_Learning( $Q$ , episodes,  $\alpha$ ,  $\gamma$ ,  $\epsilon$ ,  $t_{terminate}$ )
2:   Initialize Q-table  $Q(o, a)$  for all observations and actions.
3:   for each episode in episodes do
4:     Initialize observation  $o$  based on the agent's local mapping.
5:     Set  $t_{undiscovered} = 0$  ▷ Time counter for undiscovered cells
6:     while  $t_{undiscovered} < t_{terminate}$  do
7:       Choose action  $a$  from  $o$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy).
8:       Take action  $a$ , observe signal strength  $RSS_{dir}$ , and next observation  $o'$ .
9:       if  $RSS_{dir} = 0$  then
10:        Mark border cells in direction  $a$  as -2.
11:       end if
12:       for each neighboring cell  $c$  of current position do
13:        Observe state of  $c$  and update observation  $o$  accordingly.
14:       end for
15:        $Q(o, a) \leftarrow Q(o, a) + \alpha [s + \gamma \max_{a'} Q(o', a') - Q(o, a)]$ .
16:       if no new cells discovered in action  $a$  then
17:         $t_{undiscovered} \leftarrow t_{undiscovered} + 1$ 
18:       else
19:        Reset  $t_{undiscovered} = 0$ 
20:       end if
21:       Update observation  $o \leftarrow o'$ .
22:     end while
23:   end for
24: end procedure

```

---

- $Q(o, a)$ : The Q-value of observation  $o$  and action  $a$ .
- $o$ : Current agent observation based on local mapping.
- $a$ : Action taken, from set  $\{a_{up}, a_{down}, a_{left}, a_{right}\}$ .
- $o'$ : Next observation after taking action  $a$ .
- $RSS_{dir}$ : Received Signal Strength in the direction of action  $a$ .
- $\alpha$ : Learning rate.
- $\gamma$ : Discount factor for future signal strengths.
- $\epsilon$ : Exploration-exploitation balance parameter.
- $t_{terminate}$ : Maximum time period without discovering new cells before termination.
- $t_{undiscovered}$ : Time counter for undiscovered cells.

**Algorithm 11** Utility Determination for Cluster Configuration

---

```

1: procedure configuration_utility(configuration)
2:   utility  $\leftarrow$  0
3:   encapsulation_count  $\leftarrow$  0
4:   Set RSS bounds: max_RSS  $\leftarrow$  -10, min_RSS  $\leftarrow$  -20
5:   for each coalition in configuration do
6:     if len(coalition.agents) > 0 then
7:       Calculate maximum RSS for this coalition:
8:       RSS  $\leftarrow$  max(log(agent.SOI(agent.position, agent.coalition.frequency))) for agent in coalition.agents
9:       Normalize RSS:
10:      normalized_RSS  $\leftarrow$   $\frac{RSS - \min\_RSS}{\max\_RSS - \min\_RSS}$ 
11:      Ensure normalized_RSS is between 0 and 1:
12:      normalized_RSS  $\leftarrow$  max(0, min(1, normalized_RSS))
13:      Add normalized RSS to utility: utility+ = normalized_RSS
14:      Check for non-declination in Max RSS from global metric_data:
15:      prev_rss  $\leftarrow$  self.get_previous_metric(coalition, "Max RSS")
16:      if RSS  $\geq$  prev_rss then
17:        utility+ = 0.5
18:      end if
19:      Calculate agent similarity for the coalition:
20:      similarity_count  $\leftarrow$  0
21:      for i in range(len(coalition.agents) - 1) do
22:        for j in range(i + 1, len(coalition.agents)) do
23:          if np.linalg.norm(coalition.agents[i].position - coalition.agents[j].position)  $\leq$  300
then
24:            similarity_count+ = 1
25:          end if
26:        end for
27:      end for
28:      Penalize utility based on agent similarity:
29:      total_pairs  $\leftarrow$   $\frac{\text{len}(\text{coalition.agents}) \times (\text{len}(\text{coalition.agents}) - 1)}{2}$ 
30:      similarity_penalty  $\leftarrow$   $\frac{\text{similarity\_count}}{\text{total\_pairs}}$  if total_pairs > 0 else 0
31:      utility- = similarity_penalty
32:      Encapsulation check:
33:      if coalition.prev_best_agent and coalition.best_agent and coalition.best_agent  $\neq$  coalition.prev_best_agent then
34:        direction_to_prev_best  $\leftarrow$  coalition.prev_best_agent.position - coalition.best_agent.position
35:        direction_to_prev_best  $\leftarrow$   $\frac{\text{direction\_to\_prev\_best}}{\text{np.linalg.norm}(\text{direction\_to\_prev\_best})}$ 
36:        normalized_velocity  $\leftarrow$   $\frac{\text{coalition.best\_agent.velocity}}{\text{np.linalg.norm}(\text{coalition.best\_agent.velocity})}$ 
37:        if np.dot(normalized_velocity, direction_to_prev_best) > 0.93 then
38:          encapsulation_count+ = 1
39:        end if
40:      end if
41:    end if
42:  end for
43:  Normalize the encapsulation utility based on the number of coalitions:
44:  if len(configuration) > 0 then
45:    encapsulation_utility  $\leftarrow$   $\frac{\text{encapsulation\_count}}{\text{len}(\text{configuration})}$ 
46:  else
47:    encapsulation_utility  $\leftarrow$  0
48:  end if
49:  utility+ = encapsulation_utility
50:  return utility
51: end procedure

```

---

**Algorithm 12** Maximum Utility Configuration (MUC) Procedure

---

```

1: procedure UpdateCoalitionsAndAgents( $C, A$ )
2:   for each  $c$  in  $C$  do
3:     Update the position of the best coalition
4:     Update the position of the coalition leader
5:   end for
6:   Remove coalitions without agents from  $C$ 
7:    $G \leftarrow \text{FormCoalitionClusters}(C)$  ▷ Cluster formation
8:    $AllConfigs \leftarrow \{\}$ 
9:   for each  $g$  in  $G$  do
10:     $ClusterConfigs \leftarrow \text{GeneratePossibleConfigurationsForCluster}(g)$ 
11:     $AllConfigs \leftarrow AllConfigs \cup ClusterConfigs$ 
12:  end for
13:   $BestConfig \leftarrow \text{NULL}$ 
14:   $MaxUtility \leftarrow -\infty$ 
15:  for each  $\omega$  in  $AllConfigs$  do
16:     $CurrentUtility \leftarrow \text{CalculateUtility}(\omega)$ 
17:    if  $CurrentUtility > MaxUtility$  then
18:       $MaxUtility \leftarrow CurrentUtility$ 
19:       $BestConfig \leftarrow \omega$ 
20:    end if
21:  end for
22:  if  $BestConfig$  is not  $\text{NULL}$  then
23:    Update  $C$  to reflect  $BestConfig$ 
24:  end if
25:  Count stationary and alternating states
26:  return  $C$ 
27: end procedure
28: procedure CalculateUtility( $configuration$ ) ▷ Initialize total utility
29:    $U \leftarrow 0$ 
30:   for each  $c$  in  $configuration$  do
31:      $U_C \leftarrow \text{Compute coalition-specific utility for } c$ 
32:      $U_G \leftarrow \text{Compute cluster-specific utility for cluster containing } c$ 
33:      $U \leftarrow U + U_C + U_G$ 
34:   end for
35:   return  $U$ 
36: end procedure
37: procedure FormCoalitionClusters( $C$ ) ▷ Initialize clusters
38:    $G \leftarrow \{\}$ 
39:   for each  $c$  in  $C$  do
40:      $leader \leftarrow \text{ExtractLeader}(c)$ 
41:      $newCluster \leftarrow \text{CreateClusterWithLeader}(leader)$ 
42:     Add  $newCluster$  to  $G$ 
43:   end for
44:   return  $G$ 
45: end procedure
46: procedure GeneratePossibleConfigurationsForCluster( $cluster$ ) ▷ Initialize configurations set
47:    $Configs \leftarrow \{\}$ 
48:   for each  $a$  in  $cluster$  do
49:      $PossibleCoalitions \leftarrow \text{DeterminePossibleCoalitions}(a, cluster)$ 
50:     for each  $c'$  in  $PossibleCoalitions$  do
51:        $newConfig \leftarrow \text{MoveAgentToCoalition}(a, c')$ 
52:       Add  $newConfig$  to  $Configs$ 
53:     end for
54:   end for
55:   return  $Configs$ 
56: end procedure

```

---

**Algorithm 13** MCTS for Coalition Formation in Multi-Target PSO

---

```

1: procedure MCTS_Coalition_Search
2:   initialize_tree()
3:   while not overall_stopping_condition_reached() do
4:     node ← select_node_with_highest_UCB1()
5:     if layer_step_threshold_reached(node) then
6:       expand_node(node)
7:     end if
8:      $R \leftarrow$  simulate_rollout(node)
9:     backpropagate_results(node, R)
10:  end while
11:  return construct_best_decision_path()
12: end procedure
13: procedure initialize_tree
14:   Create root node representing the initial PSO configuration
15:   Set root level to 0 and visit count to 0
16: end procedure
17: procedure select_node_with_highest_UCB1
18:   for each child in the current tree level do
19:     Calculate UCB1 using the equation provided
20:   end for
21:   return child with the highest UCB1 score
22: end procedure
23: procedure expand_node(node)
24:   Generate new 'Options' (children) from the node
25:   For each option, set initial visits to 0 and wins to initial PSO score
26:   Add each new option to the tree
27: end procedure
28: procedure simulate_rollout(node)
29:   Make a deep copy of the current PSO 'swarm'
30:   Simulate PSO process from node's configuration
31:   Continue until a terminal state or the max step count is reached
32:    $R \leftarrow$  Number of transmitters found  $- \frac{\text{Total steps}}{1000}$ 
33:   return  $R$ 
34: end procedure
35: procedure backpropagate_results(node, R)
36:   while node is not null do
37:     Update node's visit count and cumulative wins with  $R$ 
38:     Move to the parent node
39:   end while
40: end procedure
41: procedure construct_best_decision_path
42:   Construct the path from root to the best leaf based on cumulative wins
43:   return this path as the decision path for PSO
44: end procedure
45: procedure overall_stopping_condition_reached
46:   return True if the tree has fully expanded or max steps reached
47: end procedure
48: procedure layer_step_threshold_reached(node)
49:   return True if the node's layer has reached the predefined step count
50: end procedure

```

---

**Algorithm 14** PPO Algorithm (Stable Baselines 3)

---

```

1: procedure PPO(policy, env, epochs, clip_epsilon, target_kl)
2:   for each iteration do
3:     Collect trajectories by running the current policy
4:     Compute rewards-to-go and GAE advantage estimates
5:     for each epoch do
6:       Shuffle the collected trajectories
7:       for each minibatch do
8:         Compute the ratio  $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ 
9:         Compute surrogate objective
           
$$L(\theta) = \hat{\mathbb{E}}_t [\min(\rho_t(\theta)\hat{A}_t, \text{clip}(\rho_t(\theta), 1 - \text{clip\_epsilon}, 1 + \text{clip\_epsilon})\hat{A}_t)]$$

10:        Compute policy gradient and update policy parameters using optimizer
11:        if policy update changes the action distribution too much (measured by KL) then
12:          Break from the loop
13:        end if
14:      end for
15:    end for
16:  end for
17: end procedure

```

---

- $\theta$ : Policy parameters.
- $s_t$ : State at timestep  $t$ .
- $a_t$ : Action at timestep  $t$ .
- $\pi_\theta(a_t|s_t)$ : Probability of action  $a_t$  given state  $s_t$  under policy  $\pi_\theta$ .
- $\pi_{\theta_{\text{old}}}(a_t|s_t)$ : Probability of action  $a_t$  given state  $s_t$  under old policy.
- $\rho_t(\theta)$ : The ratio of the probabilities under the new and old policies.
- $\hat{A}_t$ : Advantage estimate at timestep  $t$ .
- clip\_epsilon: Hyperparameter for clipping in the objective function to prevent too large policy updates.
- target\_kl: Maximum tolerated change in the action distribution (KL divergence) after a policy update.
- $L(\theta)$ : PPO's surrogate objective function that we want to maximize.
- $\hat{\mathbb{E}}_t$ : Empirical average over timesteps.





# Bibliography

- [1] R.D. Arnold, H. Yamaguchi, and T. Tanaka. Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. *International Journal of Humanitarian Action*, 3: 18, 2018. doi: <https://doi.org/10.1186/s41018-018-0045-4>.
- [2] Waqas Haider Bangyal, Abdul Hameed, Wael Alosaimi, and Hashem Alyami. A new initialization approach in particle swarm optimization for global optimization problems. *Computational Intelligence and Neuroscience*, 2021:6628889, 2021. doi: 10.1155/2021/6628889. URL <https://www.hindawi.com/journals/cin/2021/6628889/>.
- [3] Anthony Jack Carlisle, Kai-Hsiung Chang, Gerry V. Dozier, Dean Hendrix, and Stephen L. McFarland. Applying the Particle Swarm Optimizer to Non-Stationary Environments. PhD thesis, Department of Computer Science and Software Engineering, [University Name], 2003. Page 71.
- [4] Kurt Derr and Milos Manic. Multi-robot, multi-target particle swarm optimization search in noisy wireless environments. Idaho Falls, ID, USA, 2009. Idaho National Laboratory Department of Computer Science, University of Idaho at Idaho Falls.
- [5] Enrico Maria Fenoaltea, Izat B. Baybusinov, Jianyang Zhao, Lei Zhou, and Yi-Cheng Zhang. The stable marriage problem: An interdisciplinary review from the physicist's perspective. *Physics Reports*, 917:1–79, 2021.
- [6] G. A. Hollinger and G. S. Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014. doi: 10.1177/0278364914533443.
- [7] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.
- [8] Stanislav Khrapov. Sobol Pseudo-Random Number Generator, 2014. URL <https://sobol.readthedocs.io/en/latest/>. Built with Sphinx using a theme provided by Read the Docs.
- [9] Daisuke Nakanishi, Gurpreet Singh, and Kshitij Chandna. Developing autonomous drone swarms with multi-agent reinforcement learning for scalable post-disaster damage assessment. CUSP-GX-5006: Urban Science Intensive II (Fall 2021), 2021.
- [10] Manoj Gowda S P and Satadal Ghosh. Gradient direction turn switching strategy for source localization. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3754–3759, 2022. doi: 10.1109/CDC51059.2022.9992670.
- [11] D. Rey and M. Neuhäuser. Wilcoxon-signed-rank test. In M. Lovric, editor, *International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-04898-2\_616. URL [https://doi.org/10.1007/978-3-642-04898-2\\_616](https://doi.org/10.1007/978-3-642-04898-2_616).
- [12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
- [13] S. S. SHAPIRO and M. B. WILK. 52(3-4):591–611, dec 1965. doi: 10.1093/biomet/52.3-4.591. URL <https://doi.org/10.1093/biomet/52.3-4.591>.
- [14] Adrian Tam. A gentle introduction to particle swarm optimization. <https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/>, Oct 2021. Accessed: [Insert the date you accessed the website].
- [15] Stanisław Węglarczyk. Kernel density estimation and its application. *ITM Web of Conferences*, 23:00037, 2018. doi: 10.1051/itmconf/20182300037. XLVIII Seminar of Applied Mathematics.