THE POWER OF MULTI-AGENT SYSTEMS IN SEARCH AND RESCUE MISSIONS

A STUDY ON SEARCH BEHAVIOUR AND STRATEGY DEVELOPMENT IN MULTI-AGENT SYSTEM IMPLEMENTATIONS SITUATED IN AN ADVERSARIAL ENVIRONMENT

CHRISTOPHE FRIEZAS GONÇALVES

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE

DEPARTMENT OF
COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

STUDENT NUMBER

2059012

COMMITTEE

Dr. Murat Kirtay

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

May 18, 2025

WORD COUNT

8210

ACKNOWLEDGMENTS

I want to thank my supervisor for letting me work with his PhD student, creating the foundation for this thesis while being the second author of said paper de Koning, Friezas, Kirtay, Lindelauf, and Postma (2025). In addition to creating this opportunity by delving head first into the idea of drones and counseling throughout.
An additional acknowledgment and thanks goes to my family for standing by me throughout the process of this writing

THE POWER OF MULTI-AGENT SYSTEMS IN SEARCH AND RESCUE MISSIONS

A STUDY ON SEARCH BEHAVIOUR AND STRATEGY DEVELOPMENT IN MULTI-AGENT SYSTEM IMPLEMENTATIONS SITUATED IN AN ADVERSARIAL ENVIRONMENT

CHRISTOPHE FRIEZAS GONÇALVES

Abstract

Earthquakes, avalanches and fires count towards nature's powerful disasters that humans must face. These events lead to search and rescue operations to find victims under the rubble or stuck in dangerous environments. This study aims to develop effective and cost-efficient reinforcement learning algorithms to support search and rescue missions in adversarial environments by exploiting multi-agent drone implementations in a simulated scenario. To this end, the study compares three strategies in addition to developing a stress test to the final strategy, testing its real-world capabilities. The three strategies consist of a random baseline, a Q-learning simultaneous approach and a substitute approach. The models achieve similar results, with the substitute performing overall the best. All models depict strategic and cooperative behaviours throughout the search runs, with a strong motivation for expanded training. The study proves the viability of using the substitute approach with multiple drone group to tackle search and rescue missions effectively. The study ends by promoting future directions for the field of Search and Rescue to further develop and reinforce the proposed algorithms and emerging behaviours.

1 ETHICS, CODE, AND TECHNOLOGY STATEMENT

The data used in this study was collected by simulating the drone's behaviour in a Gazebo environment running under the Robotic Operating System (ROS2 Jazzy). The collection did not involve any human data collection. The code base and structure are built upon the Crazyflie 2.1 wall following tutorial, freely accessible via this link[1]. The Monte Carlo

[1] https://www.bitcraze.io/2024/09/crazyflies-adventures-with-ros-2-and-gazebo/

localization has been taken from Github[2] in its entirety, been rewritten, and adapted for ROS2 Jazzy and the drone implementation. The Monte-Carlo code is publicly available under the MIT license. All code chunks or parts not created by the author are clearly shown and stated in the code. The developed and used thesis code is available on GitHub under the following link[3], and all Python packages and their versions used are stated in the Methodology. The author of this paper created all the figures presented throughout.

In terms of writing, Grammarly was used to check spelling and coherence for the final submission of this thesis. No other Large Language Model was used to change or enhance the final text.

2 INTRODUCTION

Nature and its creations on planet earth are a beautiful sight to behold, yet this beauty can change from one second to another and thus bring devastating floods, earthquakes and wildfires. These disasters are nearly impossible to prevent, encouraging society to emphasize efficient handling of the aftermath. Through the actions of emergency services, casualties are minimized and lives are saved. In these cases, time is of the essence. The sooner help finds and rescues patients, the higher their chances of survival. Major issues impacting search times are human-resources, terrain and the dimensions of the search area. The bigger the area and the smaller the team, the smaller the chances that people will be found in time. A straightforward approach to mitigate these issues is to increase or decrease one of the three factors. However, the only possible factor that can be tempered with without major restructuring of living areas is human-resources.

In 1995, the scientific community added Search and Rescue (SAR) as a humanitarian research domain, thus prompting researchers to increase their endeavours into said domain (Casper, Micire, & Murphy, 2000). From this point on, scientists delved in a different direction to improve either search strategies or equipment. After 1995, this idea of a lack of human-resources and technological progress brought forward the notion of using robotic agents as a tool for search and rescue operations (Phadke & Medrano, 2023). A robotic agent brings the benefit of not endangering additional humans and being fully customizable to any given task. These agents started as four-wheeled vehicular robots and slowly developed into walking quadrupeds and unmanned aerial vehicles (UAV) for more

[2] https://github.com/yz9/Monte-Carlo-Localization/tree/master

[3] https://github.com/Kansolian/The-power-of-Multi-agent-systems-in-Search-and-Rescue-Missions

efficient traversal of uncertain and complex terrain (Drew, 2021). An addition to the customizability of robotic agents was the fast development of artificial intelligence (AI) and specifically the domain of multi-agent systems, allowing the creation and development of strategies implementing multiple robots at once, behaving in a cooperative and intelligent manner (Drew, 2021).

An important fact missing in early implementations of robotic agents for SAR is the notion of failure brought upon by dangers originating not from the terrain but the environment, i.e. adversarial environments. Not every environment is safe to search. Thus, robots can get stuck under rubble or falling debris if seen in an earthquake scenario. A fact that is still an open challenge as of the writing of this thesis (Solmaz et al., 2024).

The field of Multi-agent systems allows the creation of multi-robot implementations. In the case of simulations and drones, multi-agent reinforcement learning shows promising results for coordination and cooperative behaviour (Canese et al., 2021). Reinforcement learning helps these drone groups learn and adapt to their task and thus mitigate dangerous and hazardous environments (Rahman et al., 2022). The notion of cooperative behaviour and adaptive strategies in adversarial settings has not found much interest yet, but is a key component in tackling SAR efficiently in adversarial environments.

This brings us to the main tackle point of this paper: cooperative drone behaviour in adversarial environments. A study by de Koning et al. (2025) created the foundation for a game theory based search task between one search drone and a hider. The idea of this thesis is to develop and evaluate a hybrid reinforcement learning strategy for a search team consisting of four drones acting in an adversarial search task. The hybrid nature defines itself in the fact that its behaviour is not fully learned, thus having a hardcoded starting behaviour. In other words, fixing the fourth drone as a substitute, having it engage only if one of the three other drones has failed, creating a heterogeneous group in terms of behaviour, not physicality. The strategy aims to use the drone groups' complete power to diminish failure and use their assets most cost-effectively, given that a failed drone might entail costly repairs and wasted time.

2.1 *Research Question*

To efficiently evaluate the proposed search strategy, this paper developed a simulated environment in Gazebo using the Robot Operating System (ROS2) to test four cases. A random baseline, a search paradigm using all four drones simultaneously, the proposed strategy and a final search paradigm incorporating the proposed strategy with Gaussian noise. The

latter poses a stress test, testing a real-world data transfer error. These cases allow the study to pose and answer the following overarching research question:

> *What impact do different learning algorithms deployed on multi-agent systems (i.e., drones) have on performing search and rescue missions in an adversarial environment?*

This overarching research question can be divided into multiple sub-research questions.

Sub RQ1 *How does a homogeneous drone system using a multi-agent reinforcement learning algorithm perform in a search and rescue task situated in an adversarial environment?*

Sub RQ2 *How does a heterogeneous drone system using a multi-agent reinforcement learning algorithm perform in a search and rescue task situated in an adversarial environment?*

Sub RQ3 *What impact does Gaussian noise have on a heterogeneous drone system using a multi-agent reinforcement learning algorithm and its resulting performance?*

3 RELATED WORK

As mentioned in the Introduction 2, this thesis expands the study conducted by de Koning et al. (2025) by incorporating multiple drones in a group-based approach. Consequently, this approach incorporates multiple fields to ensure a fair comparison and a state-of-the-art approach. This section introduces the de Koning et al. (2025) study in addition to the state of the art for all necessary sub-fields used in this study's multi drone approach.

3.1 *Search and Rescue*

Search and rescue operations incorporate any tasks involving known or unknown living entities. These operations are divided into categories based on size and location, maritime or urban, to give some examples (Kumar et al., 2022). The overall goal is to help the distressed entity as fast as possible. Hoang et al. (2023) and Kumar et al. (2022) review a plethora of scientific papers to bring forth the promising gaps UAV implementations can fill in addition to highlighting the challenges these drones have to overcome. The common notion of these reviews being the harsh environments SAR

brings and the necessity for cooperative behaviours for efficient and fast search results (Hoang et al., 2023; Kumar et al., 2022).

These harsh environments, called adversarial environments, have a smaller presence in robotic-based SAR tasks. Rahman et al. (2022) and Kumar et al. (2022) use the term in the notion of drones travelling in harsh weather, damaging the robotic agent, whereas Capitol and Redmill (2023) and Behjat et al. (2021) started to tackle the problem from a proactive direction. In this case, the environment incorporates agents that actively try to harm the drones. These studies prove the concept of drones in SAR. Nevertheless, the notion of passive and proactive danger needs further study, given the different environments and approaches taken throughout both danger scenarios.

3.2 *Adversarial Search Task in Game Theory*

Lidbetter (2020) gives an initial attempt to solve a mathematical search task through game theory while exposing the main searcher to danger. The study by de Koning et al. (2025) takes this gamification further by replacing the graphs with a belief-based approach. The study uses a single drone searcher in a three-box adversarial search game. Participants were tasked to either search or hide and had to play against other participants or an algorithm-controlled drone. The study showed that a simple belief-based algorithm was outperformed by human intuition, given the on-the-fly changes per iteration, which were too fast for a belief-based algorithm (de Koning et al., 2025). The study did prove the concept of implementing search games to tackle the issue of adversarial search spaces and using simulations to train agents on such paradigms. An important note in this study is the need to expand into multi-drone implementations.

3.3 *The Gazebo Simulation Environment*

In the sense of simulations, drones have seen an increase in interest over the past decade, given the customizability and versatility of these robotic agents (Mairaj, Baba, & Javaid, 2019). In addition, simulation environments brought the same features while being highly cost-effective and not endangering hardware through testing (Christiano et al., 2016; Mairaj et al., 2019). A simulation allows for a globalised representation of use cases. A hindrance in this regard is the transition into the real world regarding AI-based simulations, yet this has found promising solutions through Reinforcement Learning advancements (Kaspar, Muñoz Osorio, & Bock, 2020).

In terms of simulation environments, Gazebo provides a strong programming foundation combined with ROS2, a reliable code base and a reliable physics engine to run robotic agent-based simulations, making it the state of the art for scientific research (Nogueira, 2014). A downside to said environment is the missing air currents simulation, which might impact aerial drone performance on real-world transitions.

3.4 *Drone Navigation and Communication*

For an efficient and cost-effective search run, drones must be equipped with state-of-the-art sensors (e.g. Lidar and cameras) and movement algorithms. Drew (2021) shows that an important feature each multi-agent implementation needs is autonomy. This autonomy allows for efficient and real-time adaptation to any environment. In terms of sensors, the state of the art for drones is a mix of motion-based and environment-scanning sensors (Kumar et al., 2022; Phadke & Medrano, 2023; Zhou, Kadhim, & Zheng, 2024). The motion-based sensors allow drones to detect their movement, thus allowing the key function of localization and motion planning. SAR drone implementations use GPS and or Odometry (Dah-Achinanon, Marjani Bajestani, Lajoie, & Beltrame, 2023; Phadke & Medrano, 2023). GPS is a precise sensor, yet unreliable in closed environments. A similar notion of unreliability can be found with odometry, a calculated belief of motor movement. This belief does not account for slippage and thus results in miscalculations of placement (Dah-Achinanon et al., 2023; Phadke & Medrano, 2023).

To circumvent this slippage and unreliability, this study keeps the drone's implementation lightweight by not using a GPS module but implementing simultaneous localization and mapping (SLAM). SLAM uses multiple sensor modalities to create landmarks while mapping its environment. These landmarks are then used as key points for reliable localization (Taheri & Xia, 2021). This brings us to the environment-based sensors. These sensors allow the use of SLAM for localization and creating a visual interface for SAR, mapping the environment and search space. The go-to sensors for SAR are Lidar and RGB, depth and heat cameras (Gui, Yu, Deng, Zhu, & Yao, 2023; Phadke & Medrano, 2023). All stated sensors have the benefit of human readability and major AI compatibility. The use of SLAM and the highly sophisticated drones used in modern studies bring great results. However, the higher the technology in the drone, the bigger the price of the whole drone group and the corresponding repairs (Drew, 2021; Kumar et al., 2022). Thus, motivating this study to focus on a lightweight approach, using only a front camera, Lidar sensors, a small

Wi-Fi antenna for communication and applying a simple SLAM algorithm relying solely on LIDAR data and Monte Carlo Localization.

In terms of communication, Dah-Achinanon et al. (2023) developed a sparse connectivity approach for communication throughout drone swarms. This approach allows for shared communication without the constant need for connection. The downside is the loss of an active searcher, given that their approach uses a stationary relay drone. In the proposed search space for this thesis, the space is small enough not to lose connectivity. This idea of using a drone as a connection relay is used in this study as a means for scalability of the proposed methodology in bigger search areas, in addition to being a key component of the proposed strategy, further discussed in the Methods 4 section.

3.5 *Multi-Agent Reinforcement Learning*

Canese et al. (2021) present a review with challenges and applications of multi-agent reinforcement learning (MARL). The previously stated review, in addition to K. Zhang, Yang, and Başar (2021), show that agents can adapt cooperative behaviour through reinforcement learning. A significant challenge which remains in the field of MARL is the notion of non-stationary environments. Each agent acts independently in the environment, thus changing it actively for every other agent. Consequently, K. Zhang et al. (2021) shows that even if the AI of each agent does not converge, they achieve sufficient and significant performance. Hernandez-Leal, Kaisers, Baarslag, and de Cote (2019) give multiple remedies to the issue of non-stationarity, a promising approach is the use of Q-learning and joint actions. The idea of joint actions is used in this study. However, it is given a specialised and simplified twist for these search groups. Joint actions would put a bigger load on data transfer if all actions were shared and handled by each drone or supervising program.

4 METHODS

With the underlying theory and state-of-the-art set, we continue with the methodology applied in this thesis. To this end, this section starts with a description of the simulated search environment and the drones, followed by a detailed explanation of the movement and applied SLAM algorithms. After which the learning algorithm and training process are described. Subsequently, the four different approaches and thus test cases are stated, and their differences are illustrated. This section is finished by describing the evaluation process and the data collection procedure.

4.1 *Simulation*

As previously stated, the study uses an indoor scenario to simulate a search game. The adversarial aspect is not implemented visually, and the scenario itself is meant to mimic a post-earthquake narrative. This notion will be explained further down in this section. The gazebo simulation environment provides this environment and runs in combination with the Robot Operating System (ROS2). This combination allows for a robust enactment of a warehouse, taken from an open-source gazebo repository [4], to illustrate a room combining open spaces and narrow passages. Figure 1 depicts the simulation environment from an angled view.

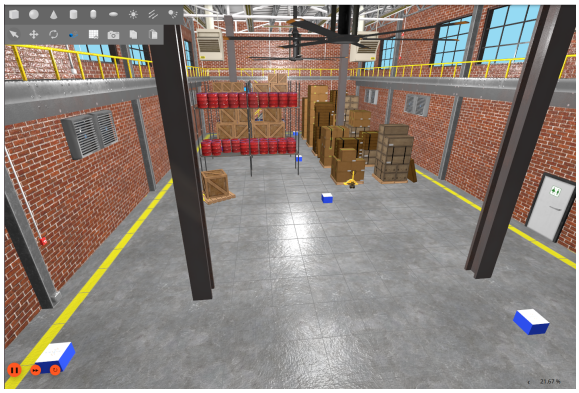


Figure 1: Warehouse environment in the Gazebo environment. Blue boxes represent hiding spots

With the environment introduced, we can now proceed to the search spots. The environment contains eight different search spots. The amount was chosen based on the size of the search space, the number of drones and the search time in a worst-case scenario, resulting in the best combination with eight spots. These are spots, known to the drone by x and y coordinates. They are represented by blue boxes as seen in Figure 1 and 2. The choice of using fixed coordinates compared to open visual-based recognition is due to time constraints and the focus of this thesis lying mainly on strategy and not visual recognition systems. The placement of said boxes has been chosen at a quasi-random rate. In other words, the positions are random but equally distributed regarding danger. These spots have failure probabilities assigned to them, bringing the adversarial nature to our environment. Figure 2 depicts a sensor-based representation of the search space, created by the drones. The closer a box is to the shelves, the higher the probability of failure. This fact is chosen to stay in line with the earthquake narrative. Shelves can collapse, or objects might become

[4] https://app.gazebosim.org/fuel/models

unstable upon them and fall after some time, post-earthquake. The highest probability is 50% and the lowest 10% at the blue ellipses' circumference. An important note is the fact that every box has a danger probability of at least 10% representing a constant danger of the building collapsing.

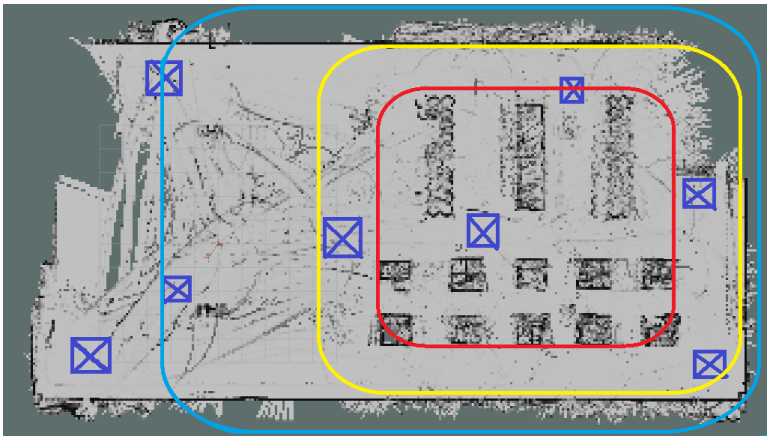


Figure 2: A sensor based representation of the search space after 50 iterations during training. Boxes present in the red ellipses acquire a danger rate of 50%, in the yellow ellipses a danger rate of 50% to 30% the further away they are from the red area. The same applies to the blue area with danger rates from 30% to 10%

Our search agents in this study are the Crazyflie 2.1 drones, Figure 3 showing one of these agents rendered in the gazebo environment. The drones are small and lightweight with an extensive component repertoire and code base for customizability (Bitcraze, 2022). As stated previously, these agents are equipped with Lidar sensors, RGB cameras and Wi-Fi antennas, maximising efficiency in terms of battery life and allowing for scalability for future studies.

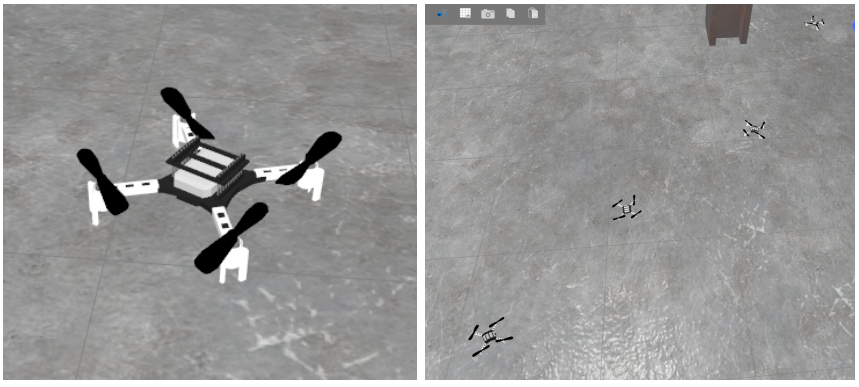


Figure 3: The Crazyflie 2.1 drone and the starting grid of the group at each iteration

4.2 *Movement, Navigation and SLAM*

The search space needs to be traversed effectively, given that the search spot coordinates are known, which gives the drones the ability to take a direct approach. In other words, the movement algorithm calculates the angle of the drone in relation to the desired search spot by using trigonometry and turns the drone towards said box. The distance from the drone to the search spot is traversed by flying forward and using a simple obstacle avoidance function to fly around any object along the path. The avoidance algorithm uses four lidar sensors scanning the four cardinal directions. If the front sensor detects an obstacle, the drones move either left or right, if those sensors deem those paths traversable. As for movement speed and height, the drones are programmed to keep a height of 1.5 meters and a speed of 0.5 meters per second. Higher values for forward speed have been tested, yet any higher value than 0.5 resulted in collisions with objects due to reaction time confounds with the obstacle avoidance algorithm.

As stated in the background section, the drones use simultaneous localization and mapping. This helps the search task by exploring the entire search space and allowing precise localization in said space. To scan and create a map of the surroundings, the drones use the Lidar sensors. The sensors are place on the four cardinal directions and 64 samples cover the front half of the drone. The map is shared among all four drones as a means of communication and better localization capabilities.

The paper uses the motor odometry and the Monte Carlo Localization technique with the Lidar sensors for precise localization. The technique uses a recursive Bayesian filtering scheme (Pfaff, Burgard, & Fox, 2006). In simpler words, the algorithm uses a set of particles, representations of the robot or agent, spread around the entire environment map. A particle then scans the surroundings with a set number of sensors and estimates its position by comparing the scan with the representation map of the environment. (Maggio, Abate, Shi, Mario, & Carlone, 2023; Pfaff et al., 2006). All particles are then averaged to get a belief of the true position of the drone. The higher the particle and sensor amount, the higher the precision, but at the cost of computational load. For this study, the implementation for Python by Amanda (2018) was used and adapted to the drones. The two variables of particles and Lidar sensors were set to 50 and 64, respectively, allowing high precision without impacting the simulation speed too drastically.

An important note of this approach is the beginning of each search cycle, given that the search space is empty at the start. Monte Carlo localization fails to get a precise lock on the position at this stage. This is not a major hindrance, given the high precision of the odometry, at

the start. The longer the drones search, the lower the precision from the odometry, yet the higher the precision for the Monte Carlo, having both methods balance each other's flaws.

4.3 *Q-Learning and Training*

The development of an efficient and lightweight search game strategy demands the creation of a specialized AI algorithm. As the background states, this study applies Reinforcement Learning to create said algorithm and allows it to be scalable and adjustable to any environment, thus giving the drones enough autonomy in case of connectivity issues. The algorithm chosen for this paradigm is Q-learning. It allows the agents to learn probabilities for any-sized search space. In addition, having the AI train on a search space with eight different boxes allows it to be applied to any search space with eight or fewer boxes. This attribute is given by the trained Q-table, having one entry per state-action pair in the environment. A downside of this approach is the resulting size of the Q-table. Each agent necessitates such a personalized table. Higher box amounts result in a larger table. Formula (1) depicts the calculation for an n boxes search game. The C stands for the combination formula and n for the number of search boxes. The formula illustrates the exponential size of higher search spaces. For our use case, each drone ends up with a Q-table of dimensions 255x8. The table itself is initialized with zeroes.

$$\sum_{i=1}^{n} C_i^n \tag{1}$$

The Q-learning algorithm is shown in equation (2). In the given equation, Q stands for the Q-Table, s for the current state and s' for the next state, a for the current action and a' for the next, R represents the reward for the current results and y and lr stand for the discount factor and learning rate, respectively.

$$Q(s,a) = Q(s,a) + lr * (R + y * max_a Q(s',a') - Q(s,a)) \tag{2}$$

Q-learning updates the current state with the Q-value of the next state by applying a discount factor, learning rate and reward, as seen in equation (2). Canese et al. (2021) gives a detailed explanation of the different Q-learning algorithm workings, giving us the liberty of excluding a detailed explanation. The important facts to state are the aforementioned twists applied to the algorithm. In its essence, the algorithm stays the same. The twists are handled by the rewards and states themselves.

As the drones share states, explaining how the algorithm handles this communication is important. The current state is fixed and immutable

during the flight from one box to the next, if a drone successfully searches a box, this box is taken out of the next state. The next state is shared for all drones, thus incorporating a notion of joint actions by sharing the resulting impact of each action, saving computational power without impacting search times. At each box and successful search thereof, the next state becomes the current state, and the cycle repeats.

Action	Reward
Hider found	10
Succesful search	2
Failed search	-1
Redudant search	-5

Table 1: Rewards handling the Reinforcement Learning algorithm

The second twist is handled by the rewards. They help direct the Q-values to develop optimal values for the desired search strategies. Table 1 depicts the different actions and the resulting rewards. A note being the reward for finding the hider and the redundant search, both are key components of the multi-agent-based approach. The reward for finding the hider is shared among all drones. Once a drone finds the target, the signal to retreat is sent to all drones. Each agent then updates their Q-table with a reward of 10 for the action the drone took and then retreats. The redundant search penalty is set up to punish drones searching unnecessary and already searched boxes. This study hypothesises that the combination of both rewards and penalties promotes cooperative behaviour.

For training, all strategies relying on the Q-learning algorithm were trained on 1000 iterations, with a learning rate of 0.8, a discount factor of 0.95 and an exploration probability of 0.2. These variables were based on the literature by Canese et al. (2021); K. Zhang et al. (2021) and reinforced with a 100 iteration grid search for the simultaneous drone search case, from here adopted to all algorithms. As a global rule from here on out, each iteration resets once all drones fail or the hider is found.

4.4 *Strategies*

The study applies a comparison to evaluate the proposed strategy. To this end, comparative cases need to be stated. As the Introduction 2 writes, four cases are tested and programmed. Three of these cases are different in their strategic nature, and the fourth is meant as a stress test of the target strategy. Let us start with the three main categories. The first two adapt homogeneous drone groups, and the third is heterogeneous approach in terms of behaviour.

The first is the baseline. This search paradigm has four drones searching the environment at random. To give this case some fighting chance and evaluate the true cooperative power of the other two categories, this search paradigm can be seen as cooperatively random. Once a box has been searched, the box vanishes from the search space fo everyone, keeping a slight cooperative and communicative nature to the baseline.

The second case applies the AI algorithm to all four drones. As stated in the training section, communication lies solely in the environment map and next state sharing. During these runs, the drones are all active searchers.

The third case and thus the target strategy for this study applies the Q-learning algorithm to three drones with a slight adaptation to the Q-table for the last drone. The three main drones are active searchers, and the fourth is designated as the *substitute*. Once a drone fails, the *substitute* activates and uses the current global next state from the environment as a starting state. In addition, the *substitute* adds the Q-table from the failed drone to its table with a factor of 0.4. The value of 0.4 is applied, given that each active drone develops a drastically different table to which the *substitute* must learn and adapt its own specialized Q-values, putting a slight yet important emphasis on the shared information. To mitigate the search times, the *substitute* places itself in the middle of all currently active searchers. The placement allows for quicker intervention compared to staying on standby at the starting point.

The last case is set as a stress test. As simulations fall under scrutiny regarding real-world translations, we do a preliminary stress test by incorporating a real-world failure point into the third category strategy (Christiano et al., 2016). In this case, we temper with the communication between the homogeneous and the heterogeneous component, simulating a transmission error in dense areas. In other words, the *substitute* adds a Gaussian normally distributed noise to its resulting Q-table once the Q-values from the failed drone have been transmitted. Ending in a test of the coping capability of the proposed model.

4.5 *Simulation Data Collection and Evaluation*

Different stages tackle the evaluation of the previously described cases. The training and test runs will be discussed and evaluated, with a stronger emphasis on the test runs. The training is incorporated to ease the discussion of the strategic development of all cases. Each case runs 200 iterations after the 1000 iteration training runs except for the baseline, given no training necessity. As for the learning algorithms, all variables stay identical to training, except the exploration probability. The variable is set to 0.1 to focus on the learned Q-tables and less on random exploration. The data

undergoes a cleaning process, which entails the exclusion of each iteration lasting over 800 seconds ($\approx$ 13 minutes). The obstacle detection missed some cases of drone-on-drone collisions, which made the reset of the simulation impossible and thus prevented the next one from starting. Human intervention was needed, resulting in the exclusion of these iterations and labelling them as outliers. Table 2 shows the data collected per category. The search times, failed searches, in addition to hiders found and callbacks, are evaluated numerically, comparing means and extreme values between cases.

As clarification, a callback is issued once a drone finds the hider, and all remaining active searchers are then called back.

Data	Unit/Type
Search times	Seconds
Failed searches	Numerical
Hiders found	Numerical
Callbacks	Numerical
Box search Sequence	List
Q-tables	255x8 Array

Table 2: Collected data during the test runs and corresponding types

The box sequences and Q-tables are evaluated through detailed analysis, given that the nature of cooperative behaviour is challenging to translate into numerical values. Given that this thesis focuses on the emergence of cooperative behaviours, the analysis needs to incorporate a deep dive into the development of strategies and search orders during the training runs into the final test runs to thoroughly evaluate the hypothesized emergence of cooperation. Consequently, the evaluation relies strongly on exploration and explanation in this regard.

4.6 *Software*

The drone simulation implementation uses Gazebo harmonic and Ros2 Jazzy (Koenig & Howard, 2004; Quigley et al., 2009). These systems require a Linux-based operating system (Ubuntu LTS 24.04). For the different algorithms interacting and moving the drones, Python (3.12.3) and C++17 were used (Stroustrup, 1986; Van Rossum & Drake, 2009). The following packages were used with Python. NumPy (1.26.4) for coordination, navigation and Monte Carlo calculations and Q-learning, OpenCV(4.6.0.66) for the visual control and camera handling. Pickle (0.7.5) was used for data handling and saving (Bradski, 2000; Van der Walt, Colbert, & Varoquaux, 2011; Van Rossum, 2020). In terms of evaluation,

NumPy and Matplotlib (3.9.0) were used additionally for data handling and visualization (Hunter, 2007; McKinney, 2010).

5 RESULTS

The following section presents the results acquired from the previously mentioned cases. It starts by exploring the training results and proceeds with the test results. The baseline results are only present in the test section, given that no training was necessary for said case.

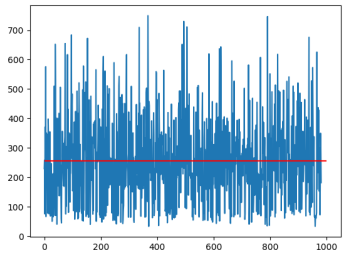
5.1 *Training Results*

The training results are split into a table giving a general overview and figures displaying the detailed results for all cases. Table 3 shows the numerical training results for the different categories. All three cases performed similarly, given the closeness of all results. The only results differing by high margins are the stress test search times, with a difference of 100 seconds to the closest mean time of 275.59. The depicted times reflect that the simultaneous case performed better with a mean of 256.25, a minimum of 33.35 and a maximum value of 749.31 compared to the proposed strategy and stress test. The stress test achieved the worst performance in terms of time, with a mean of 388.30, a minimum of 55.56 and a maximum value of 799.46.

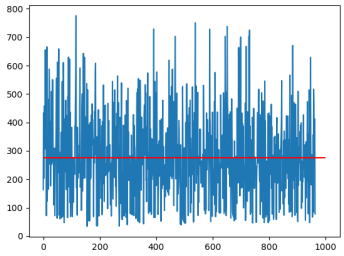
Approach	Search Times (s)			Successful Searches	Drones Failed	Callbacks
	Mean	Min	Max			
Simultaneous	256.25	33.35	749.31	655	2125	1144
Substitute	275.59	34.14	775.80	625	2153	912
Stress Test	388.30	55.56	799.46	598	2068	849

Table 3: Numerical results for the training runs

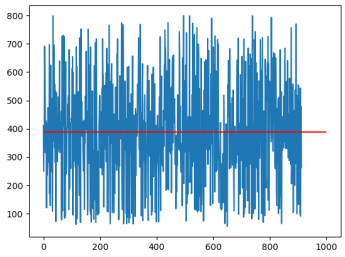
In Figure 4, we find each search time per training iteration. The figure depicts an erratic nature for all categories. The three graphs show highly varying times, highlighting the tendency of the runs to either hit a search time close to the maximum value or minimum value, rarely hitting the recorded mean value.



(a) Simultaneous case search times



(b) Substitute case search times



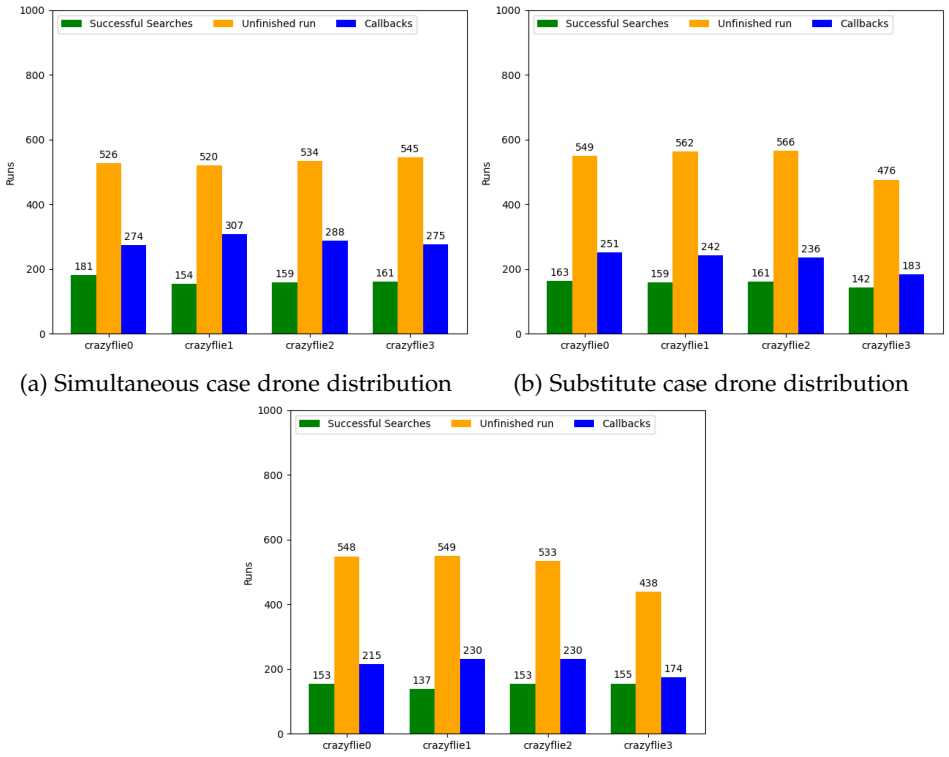
(c) Stress test search times

Figure 4: Search time evolution through the training runs

The statistics concerning the search types in Table 3 show the highest success for the simultaneous case. This strategy resulted in 655 successful searches, 2125 failed searches and 1144 callbacks. Callbacks are counted when an iteration ends by finding the hider and calling back all active remaining searchers. Table 3 additionally shows that the stress test acquired 598 wins, the least compared to the other cases. However, this case achieved the lowest amount of callbacks and failures, achieving 849 and 2068, respectively. Figure 5 gives a detailed breakdown of all drones and their performed searches. The graphs reflect an even distribution among all active search drones for the three search outcomes.

An important note is the callbacks and failed runs for the substitute drone in the substitute and stress test category. The aforementioned figure highlights the decrease in both values. The fourth drone failed 69 runs less for the substitute category and 107 runs less for the stress test than the corresponding drone in the simultaneous approach. The successful searches for the substitute drone remain even with the active searcher drones.

The strategies are explored and evaluated based on a box sequence graph. Figure 1 in Appendix A. (30) depicts four randomly pooled itera-



(a) Simultaneous case drone distribution

(b) Substitute case drone distribution

(c) Stress test drone distribution

Figure 5: Search training result distributions per category

tions from the training runs. For the simultaneous case, the corresponding four graphs show cooperative tendencies from the beginning. Iteration 86 has three drones targeting different boxes from the start. This tendency is not shared among all runs, iterations 361 and 448 have multiple drones searching the same boxes from the start or after a box has already been searched. Iteration 888 depicts a case where all drones actively searched different boxes throughout the entire run. These iterations depict the learning nature of the models during training, resulting in the emergence of cooperative tendencies.

These characteristics are shared among all three cases. The graphs for the substitute case and stress test depict similar behaviour. Iteration 522 for the substitute case shows a good example of these cooperative tendencies and highlights the substitutes' learned cooperative strategy by targeting boxes not yet touched by the active searchers. Iteration 545 illustrates a failed run in the substitute's nature, said drone searches an already cleared box as its starting target. The iteration 953 from the stress test gives an insight into the communication of the drones. Drones 1 and 2 searched the same box, yet drone 0 targeted a separate box. After the drone failed,

drone 3 activated with the current state and instantly targeted the same box. Drone 3 picked up where drone 0 failed. The remaining graphs for the stress test depict the same behaviour and tendencies as the substitute category. A clear strategic behaviour emerges, with flaws of repeated searches or the same starting targets. This behaviour develops due to the training characteristics of RL and the size of the Q-table. A characteristic further explored in the Discussion 6 section.

5.2 *Test Results*

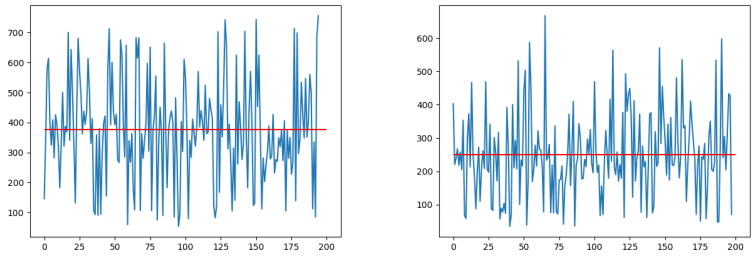
The test runs are evaluated using the same visualization as the training runs and incorporating the random baseline. A note given the results for the stress test is the size of the corresponding test set, 22 of the 200 runs had to be omitted given human intervention. The consequence of this omission will be further explored in the discussion 6.

Table 4 shows the search times and types for all four categories. Regarding search times, the simultaneous and substitute cases performed best, considering the lowest mean values of 249.75 and 250.59, respectively. The baseline took the longest, with a mean of 375.53, over 100 seconds longer than the previously stated fastest categories. The simultaneous case performed the shortest runs, considering the minimum and maximum values of 34.65 and 667.63. The other three cases achieved runs of over 40 seconds as the fastest run and over 700 seconds for their longest run.

Approach	Search Times (s)			Successful Searches	Drones Failed	Callbacks
	Mean	Min	Max			
Random Baseline	375.53	53.37	756.61	149	368	263
Simultaneous	249.75	34.65	667.63	137	431	224
Substitute	250.59	45.34	717.86	142	411	193
Stress Test	370.82	54.99	782.71	129	360	192

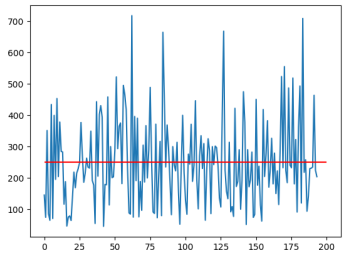
Table 4: Numerical results for the test runs

To get deeper insight into these times, Figure 6 introduces the distributions of the times per individual run. These four graphs highlight a similar behaviour as seen in the training data. The times reflect erratic fluctuations for all four cases. Compared to the remaining three cases, a observation is made in the substitute graph, the amount of major spikes. Four spikes can be seen in this category with over 650 seconds, and the residual times fall under 550 seconds. This effect hints towards greater time stability for the substitute case. A stability that is not as prominent in the other cases.

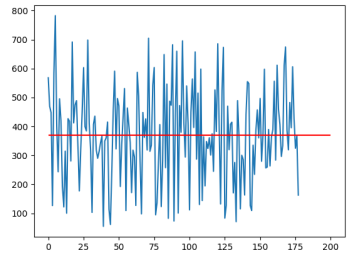


(a) Random baseline search times

(b) Simultaneous case search times



(c) Substitute case search times



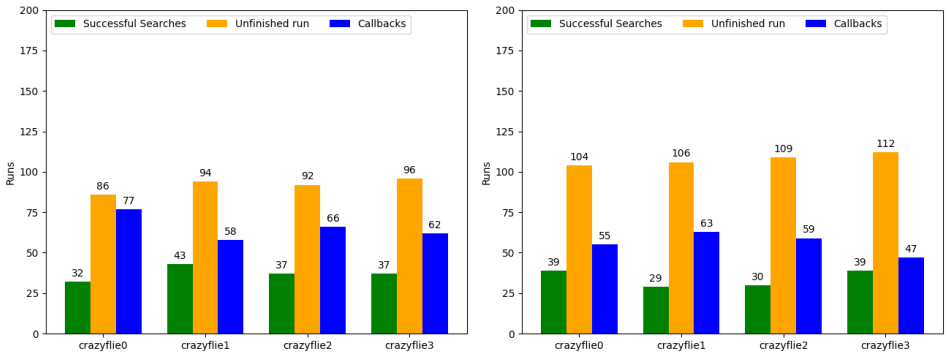
(d) Stress test search times

Figure 6: Search time evolution throughout the test runs

For the search results, the baseline outperformed the other models. It achieved 159 successes, 368 fails and 263 callbacks. The stress test achieved 8 fails less than the baseline, yet it achieved the fewest successful searches and callbacks overall, 129 and 192, respectively. Comparing the models with each other, the substitute and stress test resulted in fewer failures. In terms of success, the substitute category achieved 5 successful runs more than the simultaneous category by having 20 fewer fails.

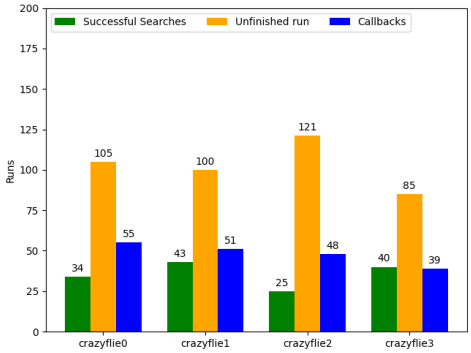
Figure 7 depicts the breakdown of the search results per drone. This figure highlights the same effect as seen in the training results. In the baseline and simultaneous category, all drones fail consistently a similar amount of times. For the baseline, an average of 92 times, for the simultaneous case, an average of 107.75. The substitute drone, crazyflie3, fails an average of 80 runs in the substitute and stress test category, significantly less than the simultaneous category. The remaining three drones fail an average of 108.7 and 95 runs for the substitute and stress test, respectively, nearly identical to the other two categories. This effect shows the cost-effectiveness of the proposed models.

The learned strategies are explored in Figure 2 in Appendix A (page 30). The baseline is omitted from the visualization given its behaviour is random. The selection is identical as in the training results section. Four iterations are picked at random, and the box sequences are graphed. The iterations

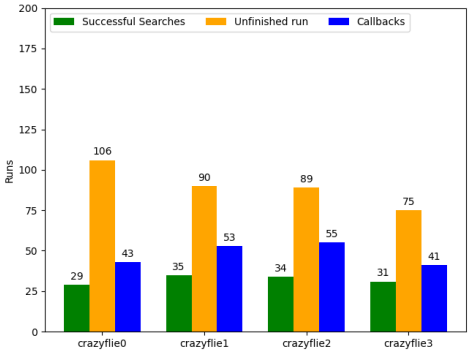


(a) Random baseline drone distribution

(b) Simultaneous case drone distribution



(c) Substitute case drone distribution



(d) Stress test drone distribution

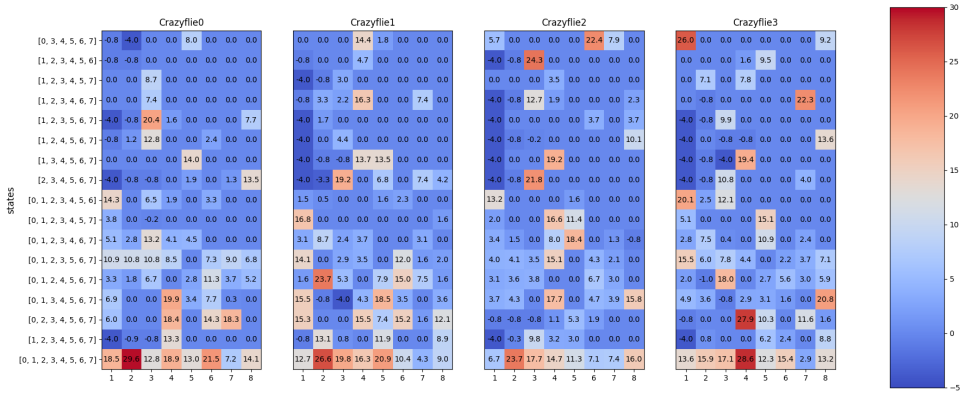
Figure 7: Search result distributions per category

for the simultaneous runs show cooperative behaviour identical to the training results. Different boxes are selected from the start, in addition to searching boxes where other drones failed, as seen in iteration 29. In iteration 116, three drones start by searching the same box, which can be explained by the Q-learning algorithm and exploration probability. An effect further explored in the Discussion 6. For the substitution category graphs, the substitute drone consistently targeted an unsearched box, and iteration 75 and 151 show the drone specifically targeting a box where a different drone failed. A clear cooperative nature emerges from all pooled runs. The same tendencies are observed in the graphs for the stress test. Iterations 60 and 190 depict cases where drones still target already searched boxes, yet show the same overall cooperative nature for all runs.

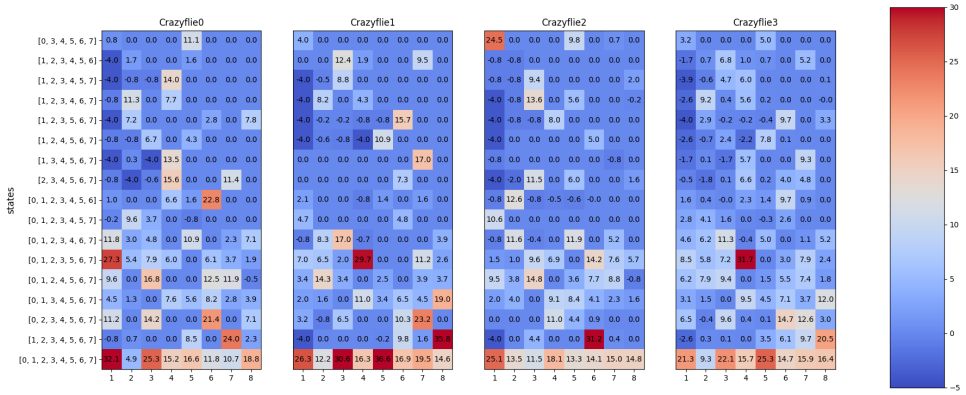
5.3 *Q-Tables*

To further explore and explain the perceived behaviour of the drones, the first 17 rows of the Q-tables are presented in Figure 8. The first seventeen rows are sufficient to explore and underpin the observed behaviour

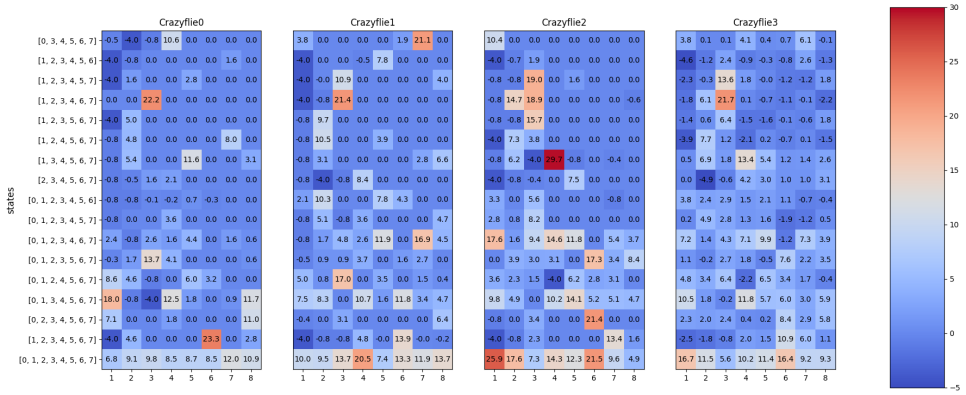
in the box sequence graphs, given the mathematical attribute of the Q-learning algorithm to spread the behaviour throughout the entire Q-table Hernandez-Leal et al. (2019). These rows from the bottom up represent the starting state, all states where one box has been searched, and eight states where two boxes have been searched.



(a) Simultaneous case Q-Table



(b) Substitute case Q-Table



(c) Stress test Q-Table

Figure 8: First seventeen rows of the Q-Table per category

The first row for each category depicts relatively high values for the entire row, symbolizing the starting state. The high values emerge based on an interaction between the exploration probability and the adversarial environment, as each box can give a positive and negative reward. Given said mixed signals, the drones develop very fluctuating and distributed starting behaviours.

The Q-Table for the simultaneous category illustrates the preference of the first three drones to target box 2 and drone 4 targeting box 4 at the starting state, no clear cooperation. A clear distinct cooperative nature emerges on state 8, where three drones deliberately explore different boxes. The first drone in this row did not encounter said state often, shown by the small values in this table entry, playing into the cooperative behaviour. This state is rarely seen by said drone given that it is already resolved by the other drones once the drone searches its current target. From state 8 onwards more and more entries become distinctively cooperative, drones start to target different boxes. A note hinting at not sufficient training is the amount of 0.0 entries in the Q-Table, showing that said path has never been selected and explored by the drone. The higher up we go in the Q-Table, the more we see these empty states. These have two causes as previously mentioned, not enough training or states that are never achieved by the drone given the work of the other drones. As well as the combination of both. An important fact to consider during the discussion.

For the substitute category, the evaluation can be undertaken differently. As from the starting point only three drones engage and the fourth incorporates data from the three drones on failure. In these initial rows, we observe that the distinctively cooperative nature already starts after the second state. The drones prefer other boxes than the ones selected by their peers, thus developing cooperative behaviour. The substitute drone shows a distributed behaviour, building onto the behaviour of others as state 17, 15 and state 7 show. These rows incorporate the information of the other drones while drone 3 learns its own behaviour. State 17 of crazyflie 3 for example is not burdened by the high values of the corresponding state of crazyflie 2, showing the adaptability of the proposed approach. The stress test performs similar to the substitute category. The resulting Q-Tables do not differ significantly in behaviour from said category. The interesting note being the substitute drone. The Gaussian noise, interferes with the transferred information, causing in this case for drone 3 to rely more on the information from other drones. Higher values from other drones, result in more concrete behaviours from drone 3 as state 16, 15 and 3 depict. Smaller values in these states are lost in the Gaussian noise.

This leads to an overall comparison of our test data. The baseline outperforms the developed models. This observation can be explained due to two factors. The first being the clever random implementation. As the Methodology 4 states, no box is explored twice in the baseline, thus drones are less in the danger zones. A behaviour not entirely perfected by the learned algorithms, this plays into the second factor, training time. The models were trained on 1000 iterations, not perfecting and filling the entire Q-Tables. This effect combined with the low deviations in search successes between model categories, proves the viability of the target approach. The proposed model does not decrease in performance compared to the simultaneous approach but diminishes the drone casualties. In addition to the box sequences and Q-Table analyses clearly depicting the emergence of cooperative behaviour.

6 DISCUSSION

This paper aims to compare three primary approaches in a simulation environment to develop efficient and effective multi-agent search drone algorithms. The study posed a random baseline, a simultaneous RL approach and a substitute RL approach against each other. To this end, the paper answers an overarching research question and three sub-questions. The first two sub-research questions, as stated in the Introduction 2, will be answered throughout the following section, by diving into the details of their results, given the explanatory nature of the sub-questions' answers. The collected data shows interesting patterns and behaviours. As a start, the comparison ended in confounding and mixed results. The simultaneous case came out on top for the training runs, achieving the best outcomes for successful searches and search times, yet only by a small margin. The random baseline performed the best in terms of successes for the test runs with a 7 run difference to the next best category. The simultaneous and substitute categories performed best for the search times, and the stress test performed best for the failed drones.

The stress test results lost some statistical power given the high number of iterations that had to be excluded due to human intervention. The performance achieved by the stress test is still relevant, given that the same effects present in the substitute category emerge from the stress test, answering the third sub-research question. The Gaussian noise influences the learned behaviour, yet the model has no issue developing cooperative strategies, relying more actively on higher transmitted values. In terms of raw numbers, the conclusion takes the diminished size into account.

All stated results show that the substitute performed best in the overall comparison, combining the benefits of the other two major competitors, the baseline and the simultaneous approach. A one-on-one distribution of the drones proved the benefit of using a substitute to diminish hardware damage. As previously stated, the results for the substitute stayed consistent with the simultaneous category while achieving significantly fewer fails for the substitute drone. This saves resources and energy while keeping search efficiency high. In addition to proving that more is not always better, as many studies in RAS and Multi-agent systems use all agents for the active task while performing their experiments (Giacomossi et al., 2024; Kumar et al., 2022; Zhou et al., 2024). These studies could be rethought by implementing substitute techniques, without losing efficiency.

An important note to touch upon is one of the comparison metrics, which is search time. The time fluctuations addressed in the Results section 5 can be understood by considering the types of searches. Failing a search and successfully finding the hider can end the run at any point. From the first searched box to the final searched box, which in the worst case must be searched by a single drone, given that all other agents have failed already. A search thus results in many different situations that can arise, making the time component unstable, with a random hider, in addition to the fact that the learned models do not consider the shortest way. The drones do not learn the distance to the boxes and thus might choose a box on the other side of the room, instead of a closer one. Iteration 29 of the test search box sequences for the simultaneous case in Appendix A (page 30) shows this situation. This limitation does not hinder a fair comparison of the models, less emphasis is put on this component, but it is nevertheless taken into account. The instability can be addressed in a further study by implementing a higher reward for closer unsearched boxes, thus diminishing search times.

In terms of strategy, one of the main focuses of this study, the Results section 5 depicts clear cooperative behaviour from the RL models. The box sequences, while not perfect at each iteration, illustrate the emergence of cooperation. Drones develop the habit of targeting different boxes than their peers, diminishing search times significantly compared to the study by de Koning et al. (2025), which used a single drone implementation in the same search environment. The success change compared to de Koning et al. (2025) has been increased by 40%, promoting the use of multi-agent systems in SAR missions.

The emergence of cooperative strategies develops from the reward system of the Q-Learning algorithm. The shared nature of the *hider found* reward allows drones that are not on target to be positively rewarded and thus encourages them to take different paths than the drone that found

the hider, covering more of the search space. To guarantee the search space coverage and not having drones search the same boxes, the *redundant search* penalty works in synergy with the *hider found* reward. This effect is shown by the box sequence graphs, yet said graphs do illustrate gaps in the hypothesized synergy.

The substitute drone develops cooperative tendencies without bigger issues in the corresponding categories. The downside is the imperfect runs and highly distributed values among the same row in the Q-Tables of the drones. The main issues for our study lie in training iterations and the adversarial environment itself. The analysis of the Q-Tables illustrates that many rows have not been touched upon, or drones having the same choices locked in at specific rows. K. Zhang et al. (2021) talk about non-stationary environments and their impact on the RL algorithms. In our case, the non-stationarity comes from two sources—the actions and consequences of other drones being in real-time and the adversarial environment.

The first point impacts the algorithm in a fashion already touched upon in the Results section 5. Some states are rarely or nearly never seen by certain drones, given the simultaneous actions of their peers. This is a positive and wished-for behaviour regarding search speed. However, in combination with the adversarial environment, the drone will end up encountering the state at one point or another. This inevitability has as a consequence, that the drones must train longer than expected to encounter every state in the Q-Table multiple times to learn a robust technique. The notion of training longer leads to overfitting AI algorithms, thus losing generalizability (A. Zhang, Ballas, & Pineau, 2018). For RL, this fact does not change. To undermine the effects of overfitting, the study setup suffices to diminish the effect. No matter the search space, the drones are trained on box coordinates, not specific spaces. Therefore, having a slight bias does not change the effectiveness of the models. The models can be trained on higher iteration counts to fill the entirety of the Q-Table in future studies. An additional remedy would be a higher exploration probability during training to promote faster exploration of all states.

The second point, already introduced in the previous paragraph, is the adversarial environment. The fail probability created by the environment gives additional complexity to the non-stationarity. A drone can explore and successfully search a box, thus being rewarded with a positive reward. In the next run, the drone might fail at said box, being handed a negative reward. This bipolarity impacts the Q-Table entry with varying signals. The penalty for failure being small promotes exploration, yet if the drone fails continuously at the same box, said box can be omitted from the search behaviour. The behaviour seen in the Q-Tables for drones targeting the same box emerges from said penalty. A box with a smaller fail probability

brings higher rewards than boxes with high fail probabilities, creating specific behavioural patterns. The study applied a higher penalty for already searched boxes to guarantee that each box was searched and thus never omitted by the previously stated behaviour. Considering that other papers had already encountered this phenomenon and problem, thus guaranteeing the robustness of our model's approach (Kumar et al., 2022; K. Zhang et al., 2021). The results prove this robustness and the power of our approach. Nevertheless, it is not flawless and motivates the implementation of different reward patterns (i.e., no penalty for failures) and the study of their impact on the resulting behaviour.

7 CONCLUSION

This thesis developed new strategies for effective and money-efficient multi-agent Search and Rescue missions, promoting faster discovery of victims while minimizing hardware damage. To this end, we conclude by answering the overarching research question posed in the Introduction 2.

> *What impact do different learning algorithms deployed on multi-agent systems (i.e., drones) have on performing search and rescue missions in an adversarial environment?*

Multi-agent reinforcement learning algorithms applied in this study developed cooperative strategies that diminish search times compared to single drone implementations while coping proactively with the adversarial component of the environment. The substitute implementation created similar performance to the simultaneous approach while diminishing drone casualties.

The study proves the effectiveness of using substitute drones to reinforce a smaller group of agents rather than all agents simultaneously. It drives the field to use resources to their full capacity in adversarial environments while minimizing casualties on both sides, victims and searchers.

The discussion tackles some points for future research. The basic concept is proven, which drives the notion of expanding and improving. As stated above, the models can undergo further training to verify and reinforce the robustness of the proposed approach. An additional comparison can be made based on different reward paradigms and the corresponding emerging behaviour thereof. A new direction to take this experiment would be based on an active hider. Thus, training a hider to react like a human in danger scenarios, needing the algorithm to prefer specific hiding spots (i.e. under tables). A final expansion would include implementing more complex RL models, such as Deep Reinforcement Learning.

REFERENCES

Amanda. (2018). *Monte-carlo-localization.* https://github.com/yz9/Monte-Carlo-Localization?tab=readme-ov-file#readme. GitHub.

Behjat, A., Manjunatha, H., Kumar, P. K., Jani, A., Collins, L., Ghassemi, P., ... Chowdhury, S. (2021). Learning robot swarm tactics over complex adversarial environments. In *2021 international symposium on multi-robot and multi-agent systems (mrs)* (p. 83-91). doi: 10.1109/MRS50823.2021.9620707

Bitcraze. (2022). *Bitcraze/crazyflie-simulation.* Retrieved from https://github.com/bitcraze/crazyflie-simulation

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Canese, L., Cardarilli, G. C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M., & Spanò, S. (2021). Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, *11*(11). Retrieved from https://www.mdpi.com/2076-3417/11/11/4948 doi: 10.3390/app11114948

Capitol, L., & Redmill, K. A. (2023). On the validation of adversarial threats to cooperative unmanned aerial systems for search and rescue missions. In *2023 ieee international automated vehicle validation conference (iavvc)* (p. 1-7). doi: 10.1109/IAVVC57316.2023.10328069

Casper, J. L., Micire, M., & Murphy, R. R. (2000). Issues in intelligent robots for search and rescue. In G. R. Gerhart, R. W. Gunderson, & C. M. Shoemaker (Eds.), *Unmanned ground vehicle technology ii* (Vol. 4024, pp. 292 – 302). SPIE. Retrieved from https://doi.org/10.1117/12.391640 doi: 10.1117/12.391640

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., ... Zaremba, W. (2016). *Transfer from simulation to real world through learning deep inverse dynamics model.* Retrieved from https://arxiv.org/abs/1610.03518

Dah-Achinanon, U., Marjani Bajestani, S. E., Lajoie, P.-Y., & Beltrame, G. (2023, Jan). Search and rescue with sparsely connected swarms. *Autonomous Robots*, *47*(7), 849–863. doi: 10.1007/s10514-022-10080-7

de Koning, M., Friezas, C., Gonçalves, Kirtay, M., Lindelauf, R., & Postma, M. (2025). Human-agent interaction in combat search and rescue games. In *Submitted to ieee international conference on robot and human interactive com munication (2025).*

Drew, D. S. (2021, Mar). Multi-agent systems for search and rescue applications. *Current Robotics Reports*, *2*(2), 189–200. doi: 10.1007/s43154-021-00048-3

Giacomossi, L., Maximo, M. R. O. A., Sundelius, N., Funk, P., Brancalion,

J. F. B., & Sohlberg, R. (2024). Cooperative search and rescue with drone swarm. In U. Kumar, R. Karim, D. Galar, & R. Kour (Eds.), *International congress and workshop on industrial ai and emaintenance 2023* (pp. 381–393). Cham: Springer Nature Switzerland.

Gui, J., Yu, T., Deng, B., Zhu, X., & Yao, W. (2023). Decentralized multi-uav cooperative exploration using dynamic centroid-based area partition. *Drones*, *7*(6). Retrieved from https://www.mdpi.com/2504-446X/7/6/337 doi: 10.3390/drones7060337

Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2019). *A survey of learning in multiagent environments: Dealing with non-stationarity.* Retrieved from https://arxiv.org/abs/1707.09183

Hoang, M.-T. O., Grøntved, K. A. R., van Berkel, N., Skov, M. B., Christensen, A. L., & Merritt, T. (2023). Drone swarms to support search and rescue operations: Opportunities and challenges. In B. J. Dunstan, J. T. K. V. Koh, D. Turnbull Tillman, & S. A. Brown (Eds.), *Cultural robotics: Social robots and their emergent cultural ecologies* (pp. 163–176). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-031-28138-9_11 doi: 10.1007/978-3-031-28138-9_11

Hunter, J. D. (2007, may). Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, *9*(03), 90-95. doi: 10.1109/MCSE.2007.55

Kaspar, M., Muñoz Osorio, J. D., & Bock, J. (2020). Sim2real transfer for reinforcement learning without dynamics randomization. In *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 4383-4388). doi: 10.1109/IROS45743.2020.9341260

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 ieee/rsj international conference on intelligent robots and systems (iros) (ieee cat. no.04ch37566)* (Vol. 3, p. 2149-2154 vol.3). doi: 10.1109/IROS.2004.1389727

Kumar, G., Anwar, A., Dikshit, A., Poddar, A., Soni, U., & Song, W. K. (2022, Jan). Obstacle avoidance for a swarm of unmanned aerial vehicles operating on particle swarm optimization: A swarm intelligence approach for search and rescue missions. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *44*(2). doi: 10.1007/s40430-022-03362-9

Lidbetter, T. (2020). Search and rescue in the face of uncertain threats. *European Journal of Operational Research*, *285*(3), 1153-1160. Retrieved from https://www.sciencedirect.com/science/article/pii/S0377221720301478 doi: https://doi.org/10.1016/j.ejor.2020.02.029

Maggio, D., Abate, M., Shi, J., Mario, C., & Carlone, L. (2023). Loc-nerf:

Monte carlo localization using neural radiance fields. In *2023 ieee international conference on robotics and automation (icra)* (p. 4018-4025). doi: 10.1109/ICRA48891.2023.10160782

Mairaj, A., Baba, A. I., & Javaid, A. Y. (2019). Application specific drone simulators: Recent advances and challenges. *Simulation Modelling Practice and Theory*, *94*, 100-117. Retrieved from https://www.sciencedirect.com/science/article/pii/S1569190X19300048 doi: https://doi.org/10.1016/j.simpat.2019.01.004

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (p. 56 - 61). doi: 10.25080/Majora-92bf1922-00a

Nogueira, L. (2014). Comparative analysis between gazebo and v-rep robotic simulators. *Seminario Interno de Cognicao Artificial-SICA*, *2014*(5), 2.

Pfaff, P., Burgard, W., & Fox, D. (2006). Robust monte-carlo localization using adaptive likelihood models. In H. I. Christensen (Ed.), *European robotics symposium 2006* (pp. 181–194). Berlin, Heidelberg: Springer Berlin Heidelberg.

Phadke, A., & Medrano, F. A. (2023). Examining application-specific resiliency implementations in uav swarm scenarios. *Intelligence & Robotics*, *3*(3). Retrieved from https://www.oaepublish.com/articles/ir.2023.27 doi: 10.20517/ir.2023.27

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... others (2009). Ros: an open-source robot operating system. In *Icra workshop on open source software* (Vol. 3, p. 5).

Rahman, A., Bhattacharya, A., Ramachandran, T., Mukherjee, S., Sharma, H., Fujimoto, T., & Chatterjee, S. (2022). Adversar: Adversarial search and rescue via multi-agent reinforcement learning. In *2022 ieee international symposium on technologies for homeland security (hst)* (p. 1-7). doi: 10.1109/HST56032.2022.10025434

Solmaz, S., Innerwinkler, P., Wójcik, M., Tong, K., Politi, E., Dimitrakopoulos, G., ... John, R. (2024). Robust robotic search and rescue in harsh environments: An example and open challenges. In *2024 ieee international symposium on robotic and sensors environments (rose)* (p. 1-8). doi: 10.1109/ROSE62198.2024.10591144

Stroustrup, B. (1986). An overview of c++. In *Proceedings of the 1986 sigplan workshop on object-oriented programming* (pp. 7–18).

Taheri, H., & Xia, Z. C. (2021). Slam; definition and evolution. *Engineering Applications of Artificial Intelligence*, *97*, 104032. Retrieved from https://www.sciencedirect.com/science/article/pii/S0952197620303092 doi: https://doi.org/10.1016/j.engappai

.2020.104032

Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*(2), 22-30. doi: 10.1109/MCSE.2011.37

Van Rossum, G. (2020). *The python library reference, release 3.8.2*. Python Software Foundation.

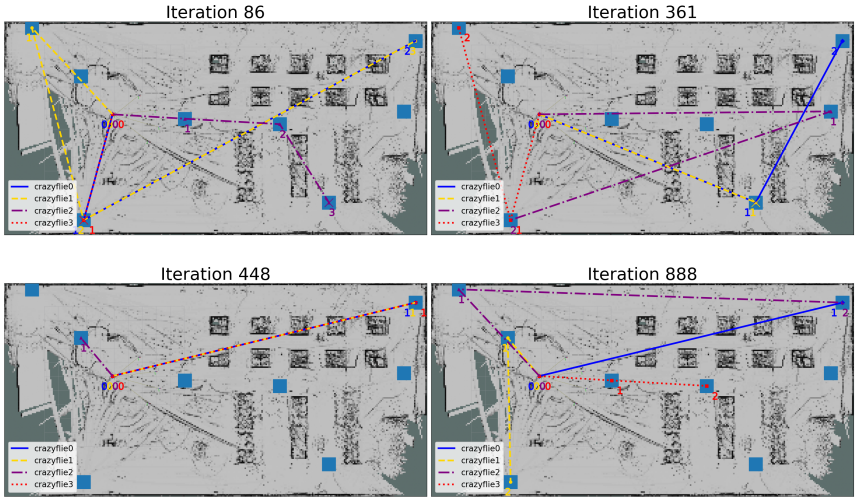
Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Zhang, A., Ballas, N., & Pineau, J. (2018). *A dissection of overfitting and generalization in continuous reinforcement learning.* Retrieved from https://arxiv.org/abs/1806.07937

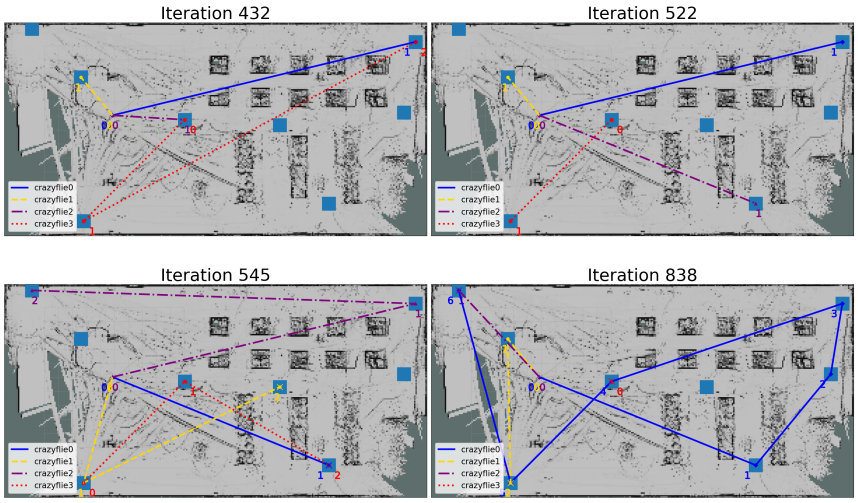
Zhang, K., Yang, Z., & Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. In K. G. Vamvoudakis, Y. Wan, F. L. Lewis, & D. Cansever (Eds.), *Handbook of reinforcement learning and control* (pp. 321–384). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-030-60990-0_12 doi: 10.1007/978-3-030-60990-0_12

Zhou, C., Kadhim, K. M. R., & Zheng, X. (2024). Multi-uavs path planning for data harvesting in adversarial scenarios. *Computer Communications*, *221*, 42-53. Retrieved from https://www.sciencedirect.com/science/article/pii/S0140366424001269 doi: https://doi.org/10.1016/j.comcom.2024.04.004

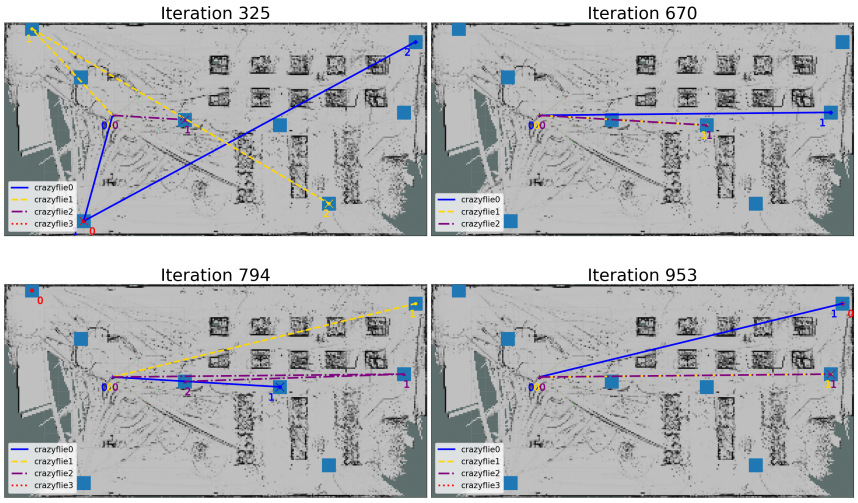
APPENDIX A



(a) Simultaneous case box sequences

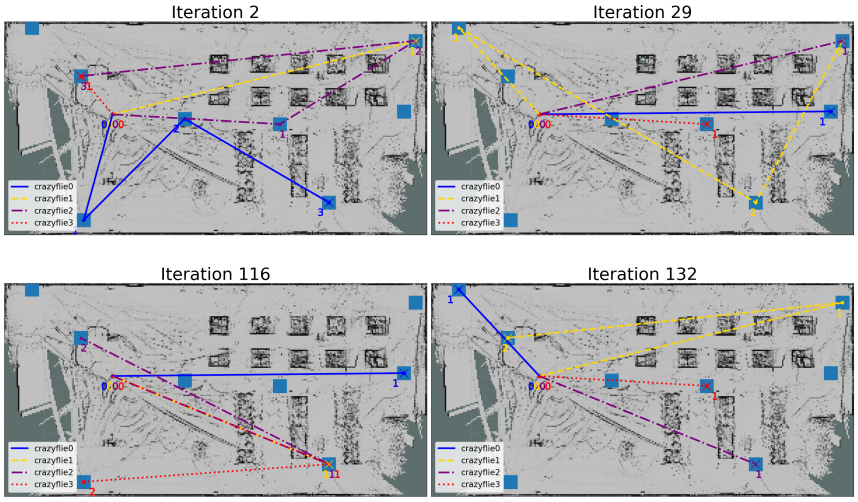


(b) Substitute case box sequences

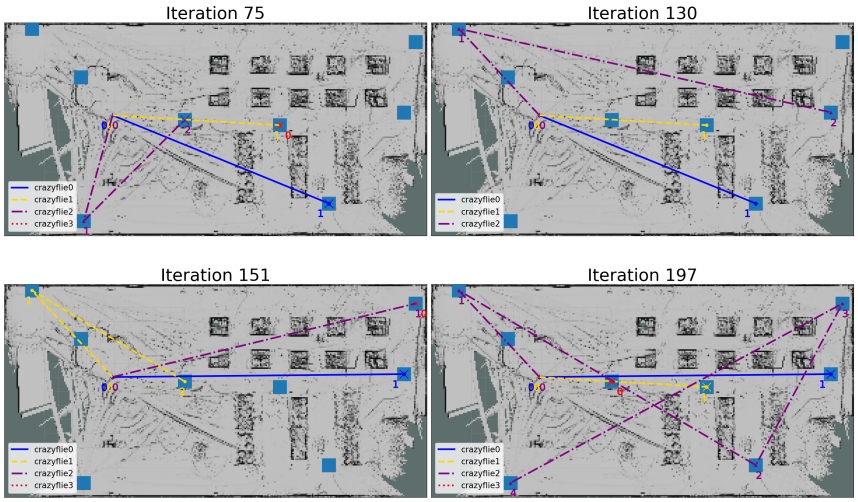


(c) Stress test box sequences

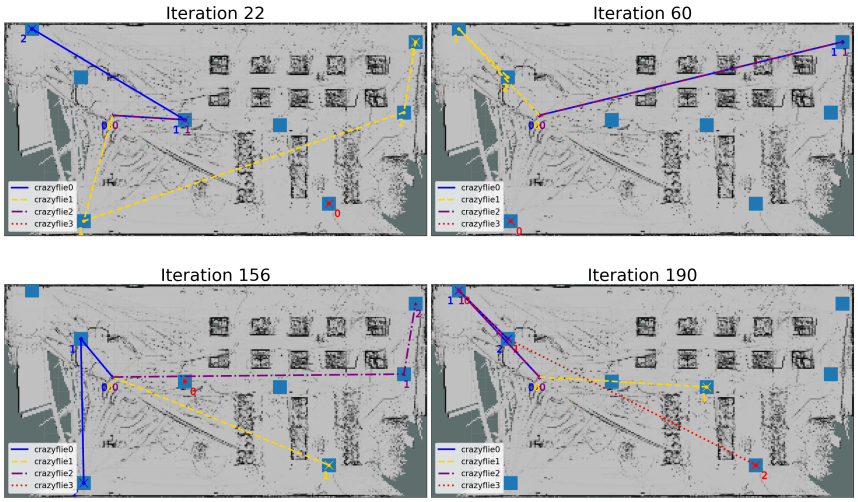
Figure 1: Search box sequences throughout training runs per category



(a) Simultaneous case box sequences



(b) Substitute case box sequences



(c) Stress test box sequences

Figure 2: Search box sequences throughout test runs per category

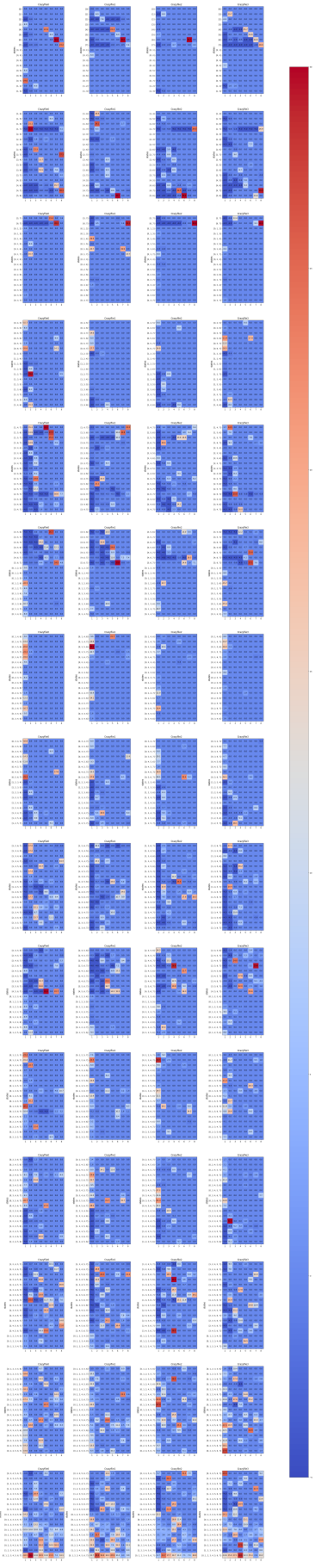


Figure 3: Simultaneous case entire Q-Table

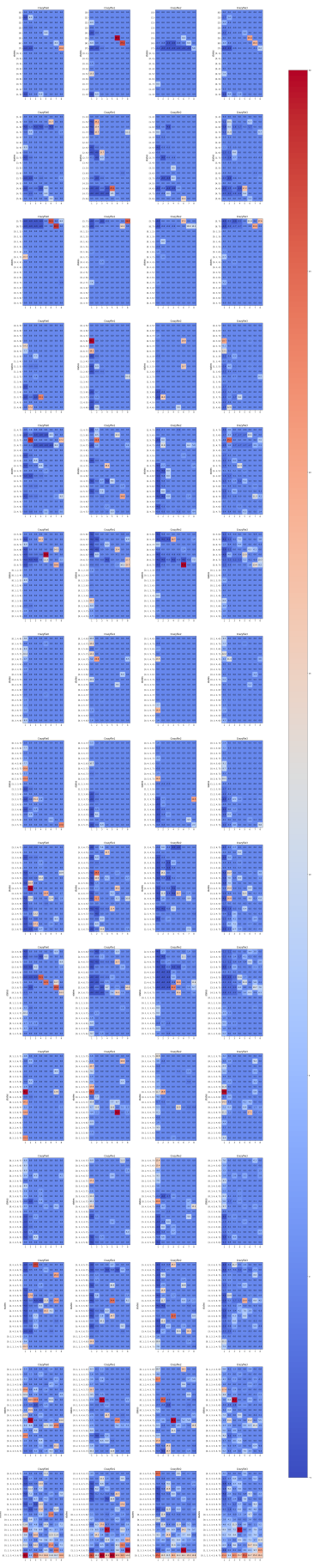


Figure 4: Substitute case entire Q-Table

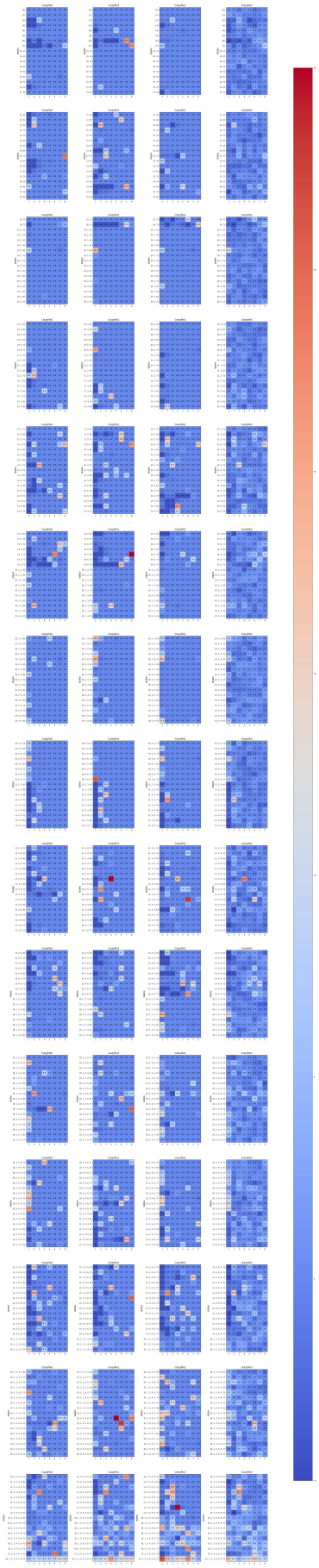


Figure 5: Stress test entire Q-Table