applied sciences

MDPI

Project Report

A Multi-Agent Based Intelligent Training System for Unmanned Surface Vehicles

Wei Han [1,2,*], Bing Zhang [2], Qianyi Wang [2], Jun Luo [1], Weizhi Ran [3] and Yang Xu [3]

1 School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China; luojun@shu.edu.cn

2 System Engineering Research Institute, China State Shipbuilding Corporation Limited, Beijing 100036, China; zhangbing_857@163.com (B.Z.); wangqianyi815@sohu.com (Q.W.)

3 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; 201722060227@std.uestc.edu.cn (W.R.); xuyang@uestc.edu.cn (Y.X.)

* Correspondence: hanwei_seri@163.com

Received: 10 February 2019; Accepted: 6 March 2019; Published: 15 March 2019

Abstract: The modeling and design of multi-agent systems is imperative for applications in the evolving intelligence of unmanned systems. In this paper, we propose a multi-agent system design that is used to build a system for training a team of unmanned surface vehicles (USVs) where no historical data concerning the behavior is available. In this approach, agents are built as the physical controller of each USV and their cooperative decisions used for the USVs' group coordination. To make our multi-agent system intelligently coordinate USVs, we built a multi-agent-based learning system. First, an agent-based data collection platform is deployed to gather competition data from agents' observation for on-line learning tasks. Second, we design a genetic-based fuzzy rule training algorithm that is capable of optimizing agents' coordination decisions in an accumulated manner. The simulation results of this study demonstrate that our proposed training approach is feasible and able to converge to a stable action selection policy towards efficient multi-USVs' cooperative decision making.

Keywords: unmanned surface vehicles; multi-agent system; training system; genetic-based fuzzy rule learning; intelligent autonomous control; modeling and simulation

1. Introduction

The modeling and design of multi-agent systems for applications in the evolving intelligence of unmanned systems is interesting and promising [1–4], especially in situations where traditional methods can be costly, dangerous, or even impossible to realize. Several applications can be found in a very broad range of domains such as energy [5–7], security [8], robotics [9], and resource management [10]. A multi-agent system consists of autonomous agents that interact in an environment to achieve specific goals [11]. An autonomous agent, in this sense, is able to perceive its environment and perform actions using actuators.

Over the years, there have been several studies that have proposed principles for designing multi-agent systems, as well as approaches to coordinate the individual behavior of agents [12–16]. In most multi-agent application domains, a priori specification of the optimal agents' behavior is difficult due to the complexity and/or dynamics of the environment [11]. In such environments, it is natural for agents to adapt or learn optimal actions that maximize performance on-line. However, one of the key challenges is the need to build a simulation platform that can be used for fast training so as to gather enough training data to promote the intelligent development process.

In the context of multi-surface vehicles' modeling and design [17–19], an unmanned surface vehicles (USVs) system is one of the development trends of modern weapon equipment [20].

The unmanned surface vehicle system has become an important means of information confrontation, precision strikes, and special combat tasks in future war [21]. Over the years, unmanned surface vehicles have long been applied to varying kinds of military applications in the real world. Some of the application domains include anti-mine warfare, submarine warfare, reconnaissance, and surveillance [22–24]. Due to the open and complex dynamic combat environment, the USV combat system must develop in the direction of autonomy and synergy, and the future surface unmanned combat should have a strong autonomous capacity, be able to be controlled autonomously, and complete complex and diverse combat tasks independently or collaboratively in a complex uncertain environment [25]. However, human knowledge is difficult to apply directly to the coordination of USVs.

We propose a multi-agent-based intelligent training system for unmanned surface vehicles (USVs). We focus on the problem of training agents of a multi-agent-based unmanned surface vehicles system where no historic data concerning the agent behavior is available. Using the team learning framework, we provide approaches for learning the rule base for multi-agent systems, designing the learning environment for simulating cooperative and competitive agents' behavior, and gathering competition data from agents' observation for off-line learning tasks. To this end, this paper first presents a decentralized coordination platform and simulator design based on a multi-agent architecture. Secondly, a USV-based agent model is presented. The paper also proposes a data collection platform for agent learning and USV training. Lastly, a fast learning and training algorithm design based on the agents' rule base is presented.

To evaluate our approach, we used an island-conquering scenario where two teams of unmanned surface vehicles compete to conquer islands in an environment. We model this case study as a partially-observable stochastic game where one team has to learn the behavior that maximizes their returns against a human-controlled team. Our empirical evaluations show that our approach to learning the knowledge base of a multi-USV system, when applied to the trained team, was able to find a knowledge base (KB) to achieve better performance.

We first present the problem this study seeks to address in Section 2. Section 3 discusses the multi-surface vehicle training system design. Finally, in Section 4, we discuss our experiments and analysis of the simulation results followed by the conclusion of our study in Section 5.

2. Multi-Agent-Based USVs' Training Problem

We first discuss the multi-agent-based intelligent training system learning problem of this study.

In general, a stochastic game is an extension of the Markov decision process (MDP) to the multi-agent context. In such a game, agents may have conflicting goals, and their joint actions determine rewards and transition between states. Stochastic games usually assume agents as having a complete view of their states, whereas the partial observation case is discussed under the more general partially-observable stochastic games (POSGs) domain. Our study is performed under the POSG case.

Theorem 1. *A POSG is a tuple* $< X, U_1 ... U_n, O_1 ... O_n, f, R_1 ... R_n >$ *where:*

- n *is the number of agents*
- X *is the finite set of states*
- $U_i, i = 1, ..., n$ *is the finite set of actions available to the agents, which form the joint action set* $\mathbf{U} = U_1 \times ... \times U_n$
- $O_i, i = 1, ..., n$ *is the finite set of observations of the agents. The joint observation is denoted as* $\boldsymbol{o} = \{o_1, ..., o_n\}$
- $f : X \times \mathbf{U} \times X \rightarrow [0, 1]$ *is the Markovian state transition function where* $f(x'|x, \boldsymbol{u})$ *denotes the probability of reaching state* x' *after the joint action* $\boldsymbol{u}$ *in state* x*. The joint action of all agents* $\boldsymbol{u}$ *at a stage* t *is denoted as* $\boldsymbol{u}_t = [u_{1,t}, ..., u_{n,t}]^\top \; u_t \in \mathbf{U}, u_{i,t} \in U_i$
- $R_i : X \times \mathbf{U} \times X \rightarrow \mathbb{R}$, $i = 1, ..., n$ *are the reward functions of the agents.*

In this study, we consider a team of multiple unmanned surface vehicles that must be trained to perform a given mission m in a mission space $\mathcal{D} \subset \mathbb{R}^n$ consisting of opposite forces. Let $H(m)$ be the objective function dependent on the mission. Given that a team T consists of N USVs, the dynamics of the i^{th} USV is given by:

$$\begin{aligned} \dot{x}_i &= f_i(x_i(t), u_i(t)) \\ i &= 1, 2, 3..., N \end{aligned} \tag{1}$$

where $x_i(t) \in \mathbb{R}^n$ and $u_i(t) \in \mathbb{R}^m$ are the state and control inputs (available actions) of the USV, respectively, at time t. Depending on the USV model and mission type, a set of constraints (such as obstacle avoidance and the physical limitations of the USV) on the state $x_i(t)$ of a USV are imposed. The control inputs of a USV are bounded according to the limits of the USV's actuators $|u_i(t)| < u_{max}$.

The reward of an USV i after taking action $u_{i,t} \in U_i$ in stage t is denoted as $r_{i,t+1}$. The individual policies $h_i : X \times U_i \to [0, 1]$ of the USVs in a team form the joint policy $\mathbf{h}$. In the case of team learning, the joint policy is defined by the single learner. Since the reward of each USV in a team depends on the joint action, the respective USV rewards depend on the joint policy:

$$R_i^{\mathbf{h}}(x) = \mathbf{E}\left\{ \sum_{t=0}^{T} \gamma^t r_{i,t+1} | x_0 = x, \mathbf{h} \right\} \tag{2}$$

where γ is the discounting factor and T is the length of the horizon. The training objective for the intelligent training system is to find a policy that maximizes Equation (2) such that the trained team of USVs can outperform its rival USVs, while adapting to the changing dynamics of it's operating environment.

3. Multi-Agent-Based Training System Design

In this section, we discuss the multi-agent-based training system design of our study, as well as the ideas involved in training the agents. To start with, the section first presents the architecture of the multi-agent-based intelligent training system. Next, we present the USV agent model for cooperative training and competition. Finally, the section presents the agent data collection platform for gathering competition for the agent information sharing and learning algorithm.

3.1. System Architecture

The architecture of the system used in this study is presented in Figure 1. We decouple the agents' control from the main environment in order to allow different implementations of controllers. In this case, we consider the multi-agent simulation platform as a server, and any attached controller becomes a client. Each USV agent selects its action based on the observed information from the environment. Based on the data visualization model of the USV, the data collection platform forms and mounts the corresponding resource tree structure of the USV. Historical data are provided to the upper machine learning algorithm by the data collection platform after data fusion. The obtained algorithm controller is then loaded into the USV.

The multi-agent cooperative reasoner component serves as the policy used to select an action for a controlled agent using the agent's local observation. In this study, we model the reasoner as an FISdecomposed into a genetic fuzzy trees (GFT). Thus, the reasoner maintains a KB, which is used for the mapping fuzzy values of the observation of an agent to an action for execution in the POSG.

The learner component of the intelligent control model facilitates KB learning, tuning, or optimizing. It seeks to find the policy that exhibits the designed behavior of the controlled team as specified by the reward function. Thus, gradient-based approaches such as neural networks (NN) [26], as well as non-gradient-based approaches can be employed for this purpose. As already indicated in our usage of GFT for the reasoner component, we use the GA in this study to learn the fuzzy rules

and tune the MFsof a GFT. Each candidate KB is applied to the control problem for the whole of a simulation episode (we use round and episode interchangeably) as found in the Pittsburgh approach. Hence, the rewards that are received each time step are aggregated and assigned as the fitness value of the candidate KB used at the end of the round/episode.

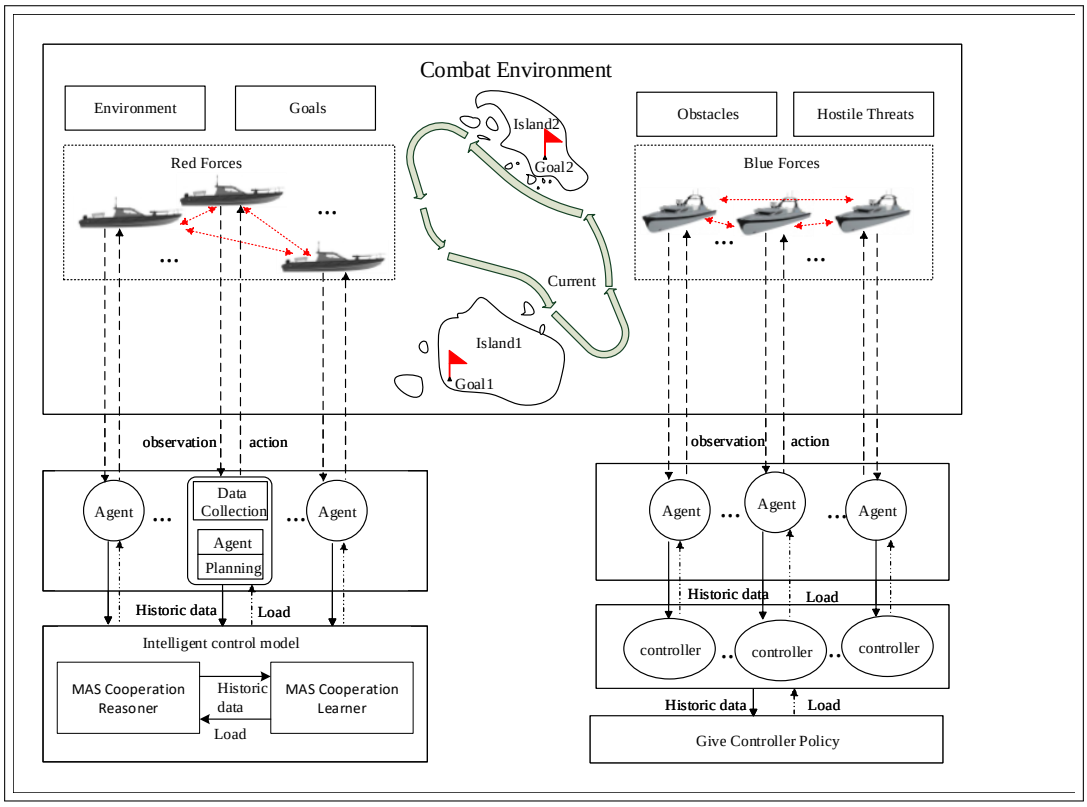


Figure 1. The architecture of the multi-agent based intelligent training system.

3.2. Agent Design

The USV agent model that is used by the intelligent training system for cooperative training and competition is equipped with a radar and weapon system. The radar of a USV can detect a hostile threat within it's detection range. In this work, the weapon system of a USV consists of a calibrated gun with limited forward turning angle and firing range. A USV can turn in both directions (left or right) and with limited speed and turning radius r. In this study, only surge, sway, and yaw are used to describe the USV's movement at sea, as shown Figure 2). Therefore, the kinematic relationship between the USV position in the global inertial frame XYZ and the boat body-fixed frame xyz can be defined as:

$$\begin{aligned} x &= u\cos(\psi) - v\sin(\psi) \\ y &= v\sin(\psi) + v\cos(\psi) \\ \psi &= \lambda \end{aligned} \tag{3}$$

The location and orientation of USVs in the coordinates of the environment are represented by (x, y) and ψ in the Earth-fixed reference frame, while u, v, and λ represent the velocity of surge, sway, and yaw in the body-fixed reference frame, respectively.

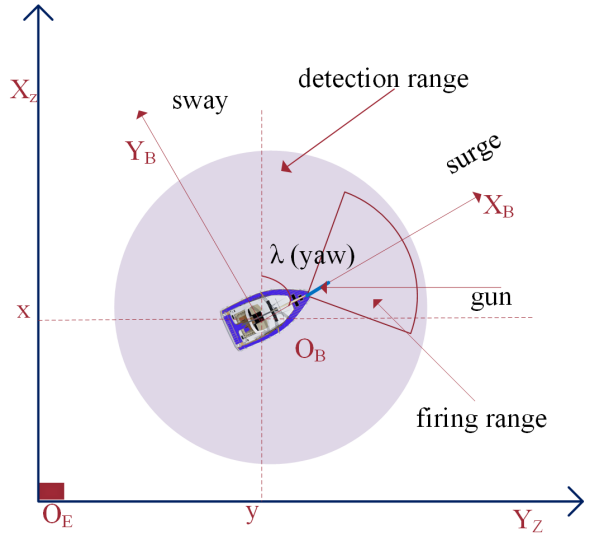


Figure 2. Schematic diagram of the USV model and planar motion where $X_E O_E Y_E$ is the Earth-fixed reference frame and $X_B O_B Y_B$ denotes the body-fixed reference frame.

3.3. Data Collection Model

Agents' observation can persist in the course of training to enable off-line operations. Data collected during training can be useful for agents' cooperative decision making. Furthermore, the gathering of training data means that different agents' learning algorithm can be employed both on-line and off-line for agents' performance improvement. Moreover, the data can be organized to enable communication between agents. The gathered domain data (in resource forms) can be used for purposes such as improving agents' performance using state-of-the-art deep reinforcement learning (DRL) algorithms. Hence, the data collector component is not only necessary, but also can extend the scope of learning algorithms that can be employed for intelligent training.

The data collection model is designed using a training and control platform consisting of loosely-coupled components and a common resource model. One such paradigm for implementing modular services is the Future Airborne Capability Environment's (FACE (http://www.opengroup.org/face)) common operating environment approach. In Figure 3, we show the components of FACE with regards to our study. Their descriptions, in the context of our study, are as follows:

- Historical training data level (HTDL): The HTDL layer consists of the entities involved in learning the KB or agents' policies for a POSG. In situations where the reasoner or policy output corresponds to composite tasks, planning and agent scheduling services may be implemented as modular components in this layer. This layer contains the historical resource data that are leveraged by the underlying learning algorithm to learn the KB.
- Cooperative decision data level: This level consists of entities that handle individual agent-level data for agent decision making. Data at this level are real-time data and observations from the agent.
- Individual agent decision model (IADM): A resource management service for hosting agents' observations composes this layer. Such a service exposes a uniform set of APIs that can be consumed by the training and control platform or the agents for reasoning or cooperation operations. Modeling the observations as resource data enables the data elements to be uniquely addressed through the APIs.
- Data collection services: Services in this layer receive data from the agents in their raw form and then perform adaptation to a common resource model for use in the training and control platform. Thus, they abstract the heterogeneous forms that agents may report from their local observation.

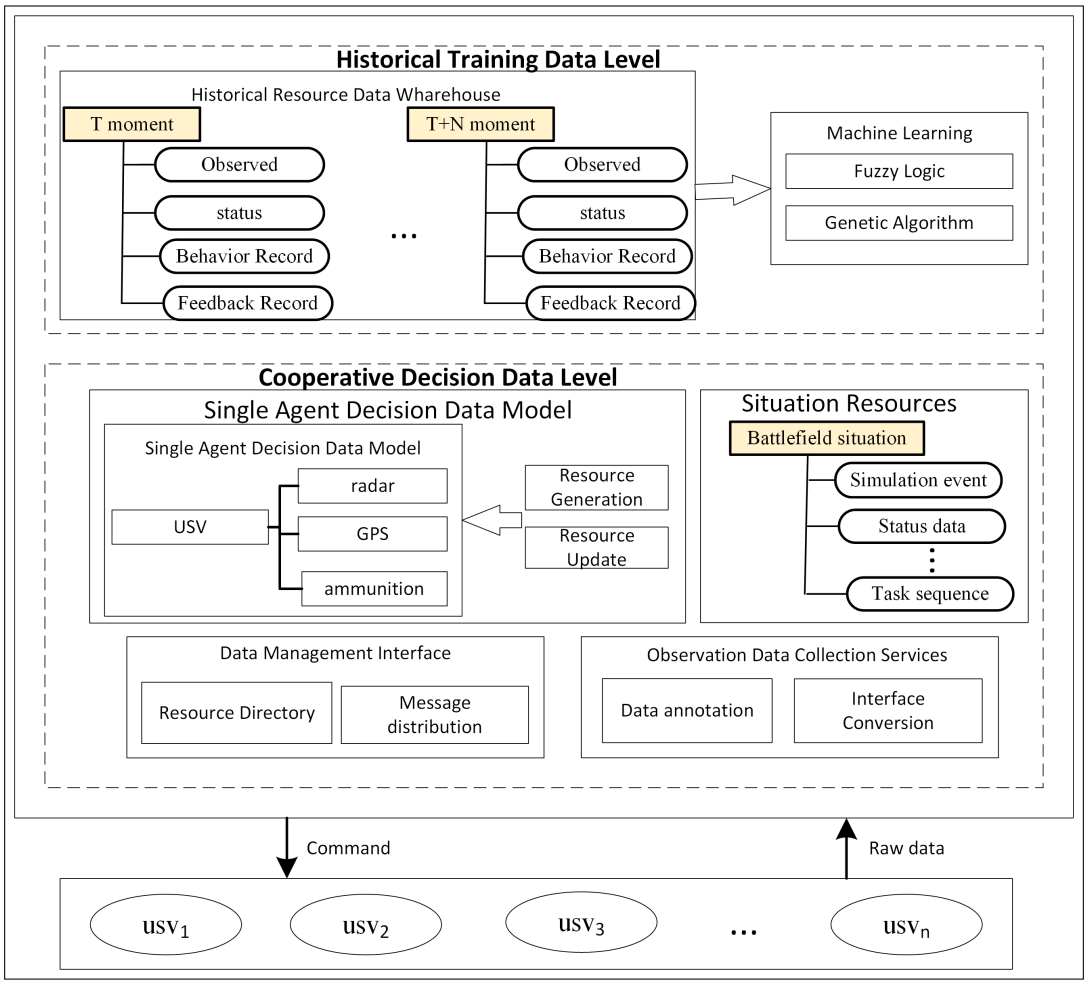


Figure 3. The architecture of the agent data collection platform based on Future Airborne Capability Environment (FACE) components.

3.4. Multi-Agent-Based USV Training Algorithm Design

The intelligent control model, as depicted in Figure 4, is responsible for performing learning and reasoning. It controls a team of agents in the simulation environment by sending agent actions to the environment every time step. Thus, at time step t, it sends the joint actions u_t to the environment for execution where the action for agent $i, i \leq n$, is $u_{i,t} \in u_t$. The intelligent control model consists of a learner and a reasoner that employs fuzzy logic and genetic algorithms (GA) known as genetic fuzzy systems (GFS) [27]. In this technique, GA is used to learn and tune the rule base and membership function of an FIS, respectively. To do this, an initial population of solutions, or strings, is created to encode the rule base and membership functions.

Classical approaches such as Michigan [28], Pittsburgh [29], and iterative rule learning [30] are mostly used to derive GFS fuzzy rules. We give an instance to illustrate how the rule base is encoded in this study using the Pittsburgh approach. Suppose X_1 and X_2 are input variables each with linguistic terms $\{term_1, term_2\}$ and output variable Y with terms $\{a_1, a_2\}$ with an arbitrary rule base of an FIS as follows:

1. IF X1 IS term1 AND X2 IS term1 THEN Y IS a1
2. IF X1 IS term1 AND X2 IS term2 THEN Y IS a2
3. IF X1 IS term2 AND X2 IS term1 THEN Y IS a2
4. IF X1 IS term2 AND X2 IS term2 THEN Y IS a1

Assume we assign codes zero and one to a_1 and a_2 respectively. The chromosome 0110 is obtained. This approach can represent an if-then rule in a single digit. This implies that each chromosome encodes possible outputs of a set of rules.

Similarly to tuning membership functions, each digit in the string corresponds to some endpoint of a membership function. GA can then be used to tune the parameters as part of the evolution process; such as found in [31], where initial parameters of a triangular MF $\mathbb{T}(\alpha, \beta, \gamma)$ are tuned using:

$$\alpha_{i+1} \leftarrow (\alpha_i + \delta_i) - \eta_i \tag{4}$$

$$\beta_{i+1} \leftarrow (\beta_i + \delta_i) \tag{5}$$

$$\gamma_{i+1} \leftarrow (\gamma_i + \delta_i) - \eta_i \tag{6}$$

where δ and η are the tuning coefficients, whereas α, β and γ parameterize $\mathbb{T}$. δ shifts the MF to the right or left, whereas η shrinks or expands the MF. Therefore, every MF has two tuning parameters that can be optimized using GA.

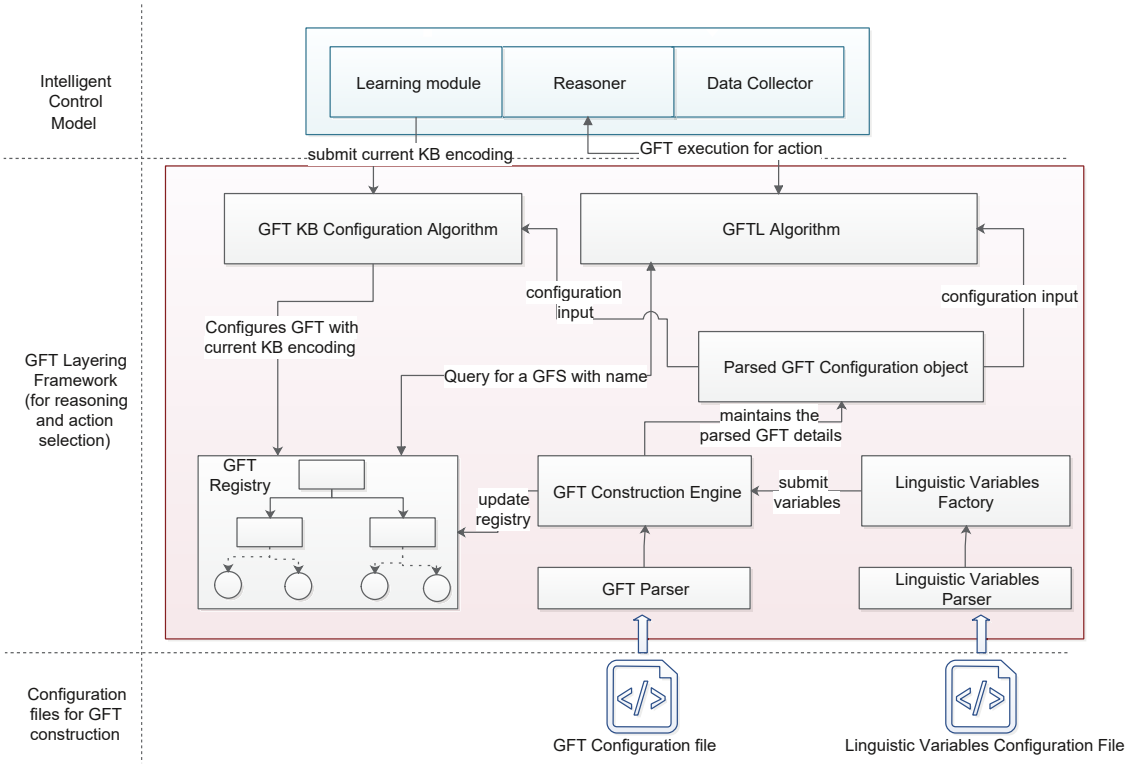


Figure 4. The architecture of the multi-agent-based intelligent training system. KB, knowledge base. GFT, genetic fuzzy trees.

Although a GFS can be used to find KBs that optimize the fitness function, it is inefficient to use a single GFS for a complex control problem. In such a case, there is an increase in GA search space complexity, the tendency to have redundant rules, and KB size. Therefore, using genetic fuzzy trees (GFT) in [2] helps to mitigate this problem. A GFT is, essentially, an ensemble of GFSs arranged in a tree structure according to a logical and conditional sequence of execution. A GFT enables the decomposition of a complex fuzzy control system (FCS) or GFS into logical sections with each node in the tree focusing on an aspect of the control problem. Each GFS in a GFT defines its own KB. Hence, the KB of a GFT consisting of $m \in \mathbb{R}$ GFSs $G = \{gfs_1, gfs_2, ..., gfs_m\}$ may have the genetic structure (represented as a concatenated string) $gfs_1^{rb} gfs_2^{rb} ... gfs_n^{rb} gfs_1^{mf} gfs_2^{mf} ... gfs_n^{mf}$ where gfs_i^{rb} and gfs_i^{mf}, $i \leq m$, denote the RB and MF string/genome, respectively, of a given GFS. This structure can be seen in Figure 5.

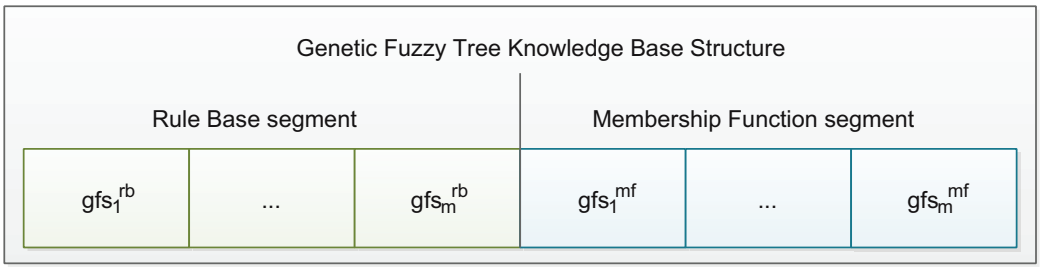


Figure 5. The knowledge base genetic structure of a genetic fuzzy tree. Each chromosome or member of the GA population takes this form.

Algorithm 1 shows how beginning with an initial population, the KB of a GFT can be learned/optimized using the GA process. In Line 2, the initial population of the GFT KB is generated as the current population. This can be based on a predefined KB set or randomly generated. In Lines 3–4, the current population is passed on to the the FIS to be used for the control task. After all these KBs have taken turns performing the control task and have been evaluated, they are then subjected to the GA operators (Lines 6–17). This operation continues till the termination condition is reached. The resulting candidate KBs are returned as the best performing or most suitable of the KBs.

Algorithm 1 Procedure: GFS procedure.

Input: GA hyperparameters
Output: Best set of GFT KBs
1: Initialize $t = 0$
2: Generate initial population as $\mathcal{P}_0$
3: Set current population $\mathcal{C}_t := \mathcal{P}_0$
4: Run the fuzzy control system using $\mathcal{C}_t$
5: Evaluate the members of $\mathcal{C}_t$
6: **while** *Not termination condition* **do**
7: $\mathbb{P}_t := selection(\mathcal{C}_t)$
8: $\mathcal{O}_t := crossover(\mathbb{P}_t)$
9: $\mathcal{C}_{temp} := mutate(\mathcal{O}_t)$
10: **if** elitism **then**
11: $\mathcal{C}_{temp} := applyElitism(\mathcal{C}_{temp})$ //keeps a percentage of previous chromosomes
12: **end if**
13: $t := t + 1$
14: $\mathcal{C}_t := \mathcal{C}_{temp}$
15: Run the fuzzy control system using $\mathcal{C}_t$
16: Evaluate the members of $\mathcal{C}_t$
17: **end while**
18: Return best-performing candidate solutions

Non-stationarity is an inherent problem in the multiple learning agents' environment. In this section, we present an approach for incorporating agent-induced non-stationarity awareness into a GFT based on the framework of [32].

Since the root of the GFT has the strongest impact on the learning process or action selection, we consider it to be the main component for addressing this problem. Unlike the other GFSs in the GFT, which can be modeled to address specific aspects of a control problem (domain dependent), we consider the root GFS to be a special case.

We regard the sub-trees that are formed under the root GFS as the elements of the BR function co-domain. Thus, a BR function, in the GFT case, selects a sub-policy to be used for selecting an action for the agent. The influence function is also represented as an FIS (or GFS in our case, since GA is applied to learn the KB), which takes PGFsas input variables and a BF as output the variable. This is motivated by the idea that an influence function maps beliefs to possible best responses and can be

designed to use deductive reasoning. These are properties that an FIS exhibits due to fuzzy logic. Thus, the root of the GFT takes incomplete/partial observations and produces its belief of another agent's policy. This connotes that the terms of the output variables then serve as possible opponent strategies as perceived by the reasoning agent.

In the case of the PGFs that serve as input variables to the GFT root FIS, the history of observations concerning the opponent or another agent is maintained and used by each PGF variable to produce an input value. This PGF input value can then be fuzzified and used in the inference process. The problem that arises is then how to design a PGF to capture the adaptation dynamics of another agent. In the work of [33], an agent i estimated the model of agent j as:

$$\hat{\sigma}_j^i(u_j) = \frac{C_j^i(u_j)}{\sum_{\bar{u} \in U_j} C_j^i(\bar{u}_j)} \tag{7}$$

where $C_j^i(u_j)$ is the frequency that agent i observed agent j taking action u_j. Therefore, given the history of observations, Equation (7) can be extended to include the observation of agent i as:

$$\hat{\sigma}_j^i(o_i, u_j) = \frac{C_j^i(o_i, u_j)}{\sum_{\bar{u} \in U_j} C_j^i(o_i, \bar{u}_j)} \tag{8}$$

where $C_j^i(o_i, u_j)$ is the frequency that agent i observed agent j taking action u_j when its local observation was o_i. $\hat{\sigma}_j^i(o_i, u_j)$ becomes the input value of agent i's PGF for monitoring a specific action u_j of agent j. With the input variables (PGFs) and output variable (BF) determined for the root FIS, GA can be used to find the best response for agent-induced non-stationarity and thereby stabilize learning in the POSG.

4. Multi-Agent-Based Simulator Design

The simulator used for the experiment was developed using the Java programming language. The fuzzy control library [34] was extended for the design and encoding of fuzzy rules for learning purposes as implemented by the intelligent control model of our architecture. In this section, we present the experiment conducted for this study and follow it up with the analysis of the simulation results.

4.1. Environment Setup

To evaluate the effectiveness of the proposed learning system, a scenario of multiple boats competing for conquering more islands while engaging in combat was adopted. In this scenario, the environment set in a maritime setting consisted of N islands and two teams of unmanned surface vehicles (boats) referred to as blue and red forces. The boats were equipped with radar and guns for detecting and shooting enemies, respectively. The guns were set to have a fixed left-to-right traversal angle and shooting range, as shown in Figure 6. Since both teams had conflicting goals, a team achieved its goal by contending with opponent boats. Each team had information of the location and number of islands and their states, whether conquered or unconquered by the team. An island is said to be conquered by a team if a member of the team moves to the coordinate of the island and stays there for that time step or no opponent boats move to that particular island conquered by the team. If two opponent boats occupy an island at the same time, the island is not awarded to any team for the elapsed time steps. For our experiment, we implemented two controllers for both teams. The controller for the blue force was made to use fixed rules provided by humans, whereas the red force controller, which we sought to train, had to learn the best rules that maximized their performance. At the beginning of each time step, the simulation environment received a batch of commands from the (ally and enemy) controllers of both teams and updated the environment.

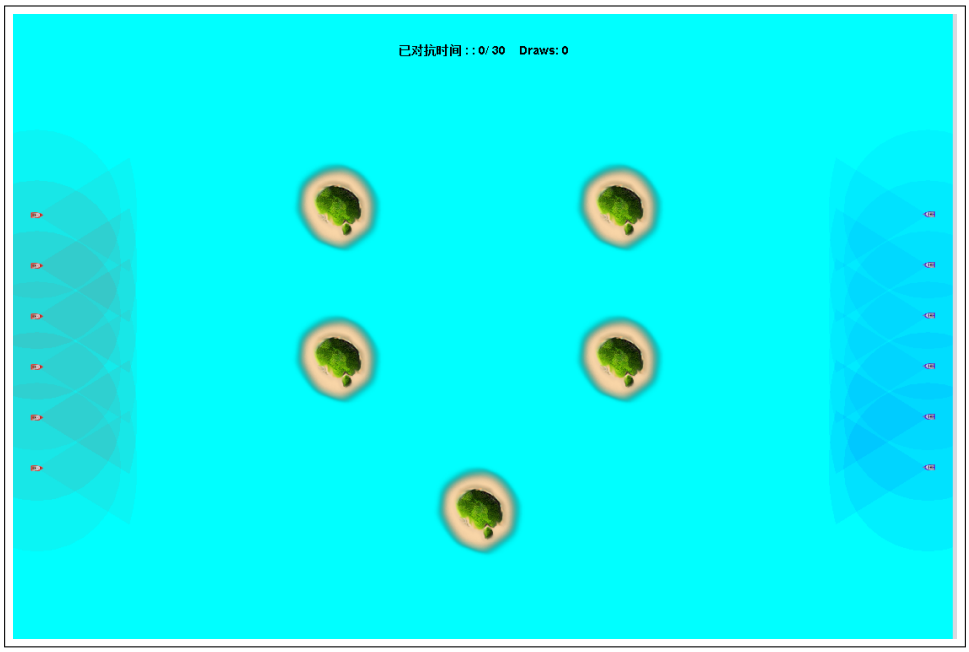


Figure 6. A snapshot of the island-conquering scenario. In the middle are the 5 islands to be conquered by both teams. The circles around the boats are their detection ranges, and the forward looking arc is the the firing regions of the boats.

The possible actions of each agent are also explained below:

- retreat: the agent moves towards its base.
- left: turn left and move.
- right: turn right and move.
- straight: move forward at the current heading.
- stop: stop moving.
- assist: move to help a teammate that is under enemy fire; a boat can assist a random or the closest teammate.
- units: causes a boat to team up with other ally boats to conquer an island.
- closest: the boat conquers an unconquered island that is closest to it.

4.2. Competition Objective

The goal of each team was to conquer all islands and destroy opponents, while staying alive. The performance of each team was evaluated after each time step using the function:

$$R_t^f = A_t^f \times p + I_t^f - D_t^{f'} \times p \tag{9}$$

where A_t^f is the destruction suffered by the opponent team f' as a result of attacks from team f in time step t, I_t^f the conquered island points received by team f, D_t^f the damaged caused by the opponent f' to the team being assessed f, and p the points awarded for boat attacks/destruction. The winner of an encounter is decided after the end of the episode. The team with the highest score is declared a winner of that encounter.

4.3. Data Collector for Training and Learning

As mentioned earlier, gathering of training data is useful in a number ways. The cooperation of agents can be enhanced when agents share their observation. Furthermore, data gathered during training can be used later by other learning algorithms such as deep learning methods for analysis and

performance improvement. The resource model shown in Figure 7 is used for on-line data gathering during training. The data that are sent to a controller as feedback take this form. The main components are as follows:

- Simulation information: This node provides general information about the simulation for a given time step.
- Observation: This provides the observation made by all agents in the team. Each agent also reports opponents that have been detected, as well as the observable properties of each detected opponent.
- System data: All miscellaneous data concerning the simulation platform (e.g., time, CPU consumption, memory usage, etc.) may be provided under this resource element.

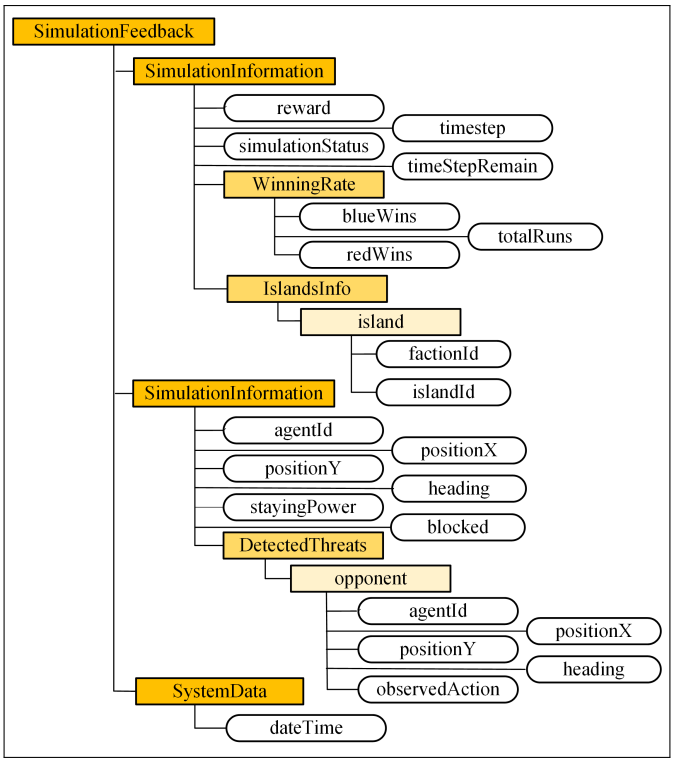


Figure 7. The simulation feedback resource structure of the island-conquering case study.

4.4. Training and Learning Algorithm

In order to provide a realistic virtual environment and opponent for the red force to train against, the blue force control algorithm uses human-defined rules. Table 1 shows an example encoding rules used for the blue force task selection during simulation. The GFT of the red force control algorithm is illustrated in Figure 8. The details of each of the GFSs of the GFT can also be seen in Table 2.

Table 1. Example encoded task selection rules used by the enemy team.

Unconquered	DetectedEnemies	TaskOutput
None	None	Retreat
None	Moderate	Intercept
None	Many	Intercept
Moderate	None	Conquer
Moderate	Moderate	Conquer
Moderate	None	Conquer
Moderate	Many	Intercept
Many	Many	Intercept
Many	Moderate	Conquer

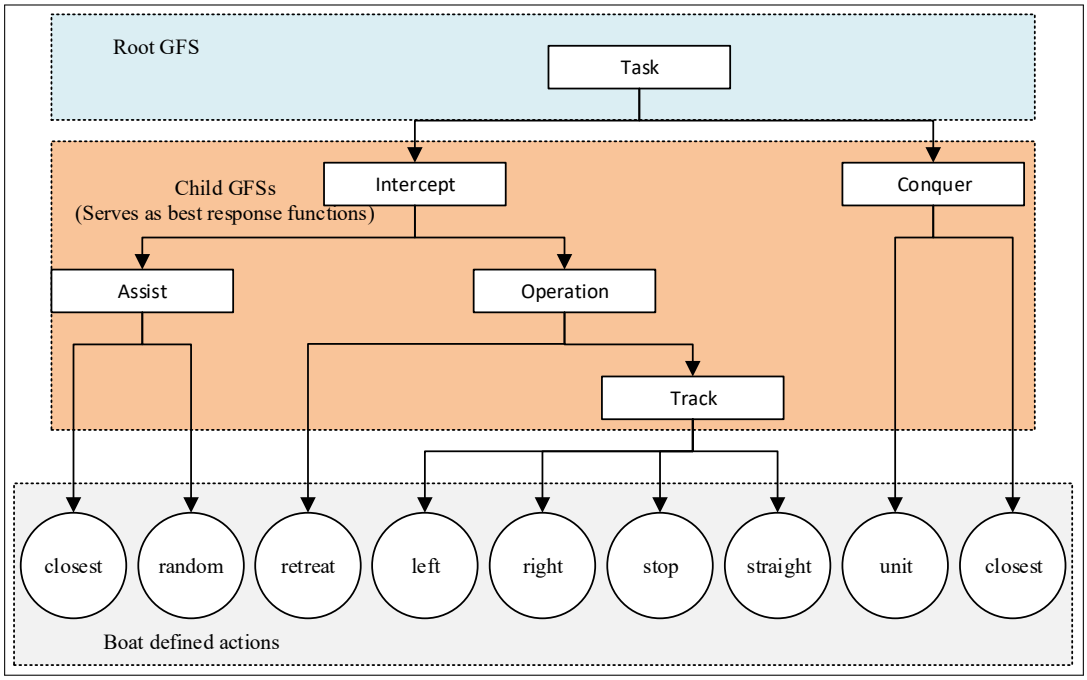


Figure 8. The genetic fuzzy tree structure used for training the ally team. The rectangles are FISs, whiles the circles represent predefined actions in the simulation.

Table 2. Details of the GFSsthat constitute the GFT used for training the ally team.

GFS	Input Variables	Output Variable	Output Variable Terms	RBSize	No. of MF Tuning Parameters
Task	UnconqueredIslands PGF_attacked PGF_moved PGF_conquered PGF_retreated	BeliefFunction	conquering Intercepting	243	30
Intercept	DetectedEnemies InEnemyFiringRange	InterceptOutput	Assist Operation	6	6
Assist	DetectedEnemies FactionDetectedEnemies	InterceptOutput	assistClosest assistRandom	9	12
Operation	StayingPower DetectedEnemies	OperationOutput	track retreat	9	12
Track	DistanceToEnemy HeadingDifference	TrackOutput	left, right stop, straight	21	20
Conquer	UnconqueredIslands	UnconqueredIslands	conquer in units conquer closest	3	6

The Assignment GFS was designed to consider opponent-induced non-stationarity with the following PGFs:

- PGF_attacked: reports the confidence level that the detected enemy boats are attacking
- PGF_moved: reports the confidence level that the detected enemy boats are moving towards an island or the agent whose local observations are being used for reasoning
- PGF_conquered: computes the confidence level that the detected enemies are conquering an island
- PGF_retreated: evaluates the confidence level that the detected enemies are withdrawing to their initial positions or base.

The linguistic terms of each PGF are low, moderate, and high. Each PGF first computes the action probabilities for each detected agent. A softmax is then computed over the reported action probabilities, and the maximum is selected as the PGF input value. The belief function output variable selects the perceived opponent strategy in each IF-THEN rule. The other input variables are:

- UnconqueredIslands: The number of islands not conquered by the team to which a boat belongs.
- DetectedEnemies: The number of detected enemies by a boat.
- FactionDetectedEnemies: The number of detected enemies by a team.
- InEnemyFiringRange: Returns 1 if this boat is in the enemies' firing range, and 0 otherwise.
- HeadingDifference: The difference in heading between this boat and the target boat.
- DistanceToEnemy: The distance between this boat and the target boat.
- StayingPower: The current strength of a boat to sustain enemy attacks.
- TeammatesUnderFire: Reports the number of teammates a boat has observed to be under attack by opponent boats.

Triangular MFs are used to define the semantics of the fuzzy rules. The initial MF tuning parameters were sampled uniformly from $[-1.5, 1.5]$ for each input variable MF. The GA parameters we set for training the ally team controller are presented in Table 3.

Table 3. The GA parameters used for training the ally team.

Parameter	Value
Population size	20
Number of generations	101 (initial population included)
Crossover probability	0.7
Mutation probability	0.1
Mutation decay Schedule	Exponential decay (decay rate = 0.15)
Crossover operator	BLXcrossover (alpha = 0.15)
Mutation operator	Adaptive non-uniform mutation (b = 5)
Selection operator	Tournament selection

As indicated in Table 3, we combined a mutation probability schedule with an adaptive non-uniform mutation to control exploration and exploitation. The ally team learner was set to maintain the last 5 observations of each detected enemy boat.

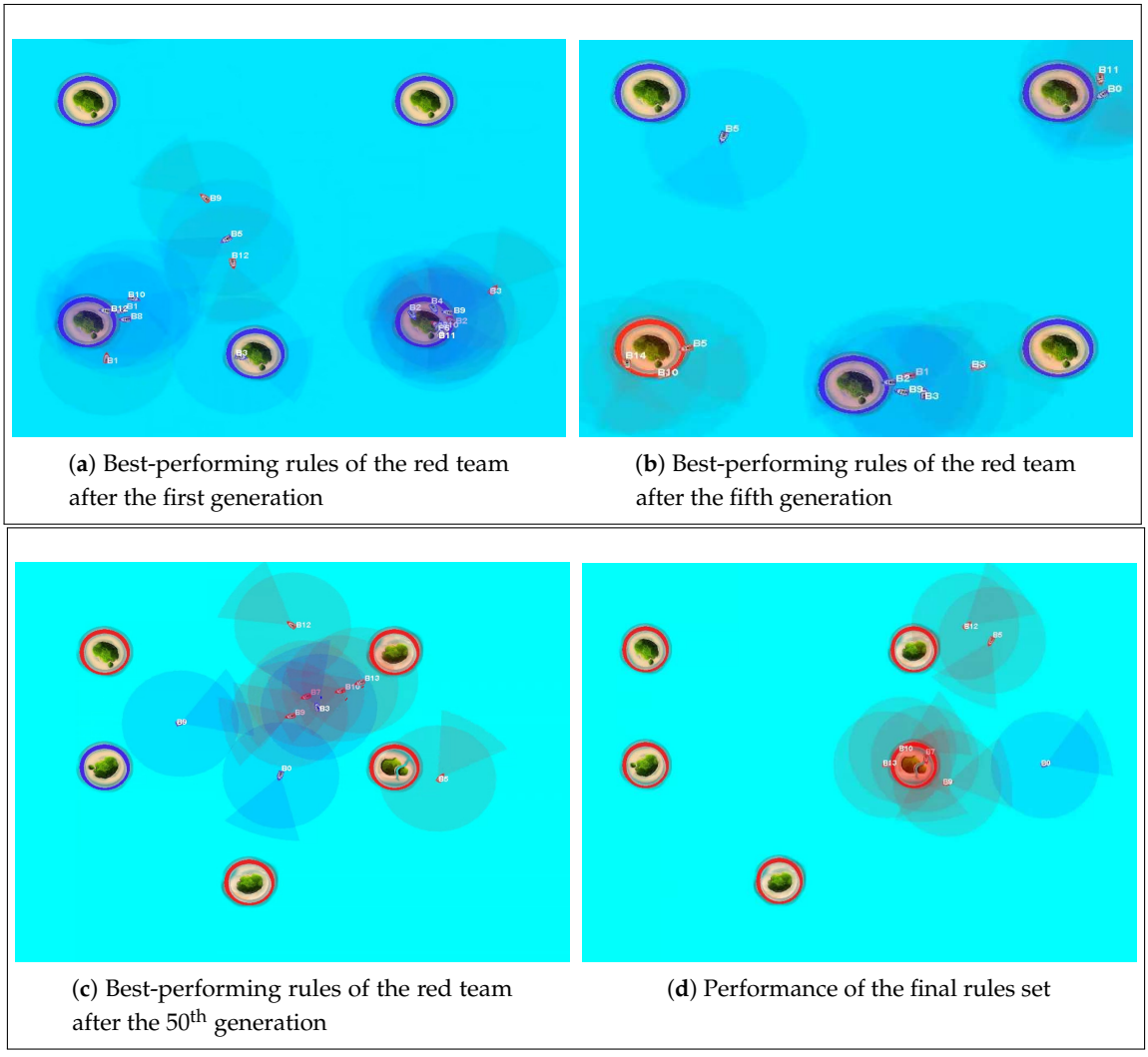
4.5. Experiment Results and Validation Analysis

We have evaluated our intelligent training system using four distinct scenarios of the island conquering case study. The parameters used in Scenario 1 for both team are shown in Table 4. Furthermore, in Scenario 1, sinking an opponent and conquering islands were both worth two points each. In Scenario 2, the number of USVs in a team was reduced to three, and sinking an opponent point was reduced to one. The simulation for Scenarios 1 and 2 was run concurrently on two different machines for 2020 from 20 randomly-generated chromosomes as the initial rules for the blue force against the enemy team, which used a human-sampled rules sets. Figure 9 shows the snapshots of the competition as simulation progresses.

If the blue force began an episode knowing exactly what to do in the environment and the red force was yet to learn, then the blue force should perform better than the red force in the initial stages of the simulation. However, as the simulation progresses, the performance of the red force should be improving since the red force learns, while the blue force uses the static rule. Furthermore, if the blue blue force is using a different set of rules even though a static set of rules, then there is a tendency for a dynamic rise and fall in the performance measure by the strength of a simulating rule set.

Table 4. The fixed simulation parameters.

Parameter	Value
Number of boats per team	6
Number of islands	5
Simulation steps per episode	30
Time per step	3 s
Radar radius	140
Forward firing angle	20
Boat's velocity	4/system iteration
Firing range	100



(**a**) Best-performing rules of the red team after the first generation

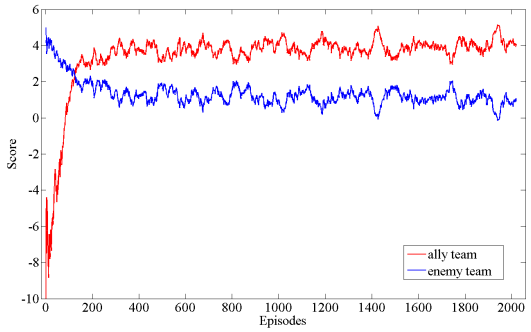
(**b**) Best-performing rules of the red team after the fifth generation

(**c**) Best-performing rules of the red team after the 50^{th} generation

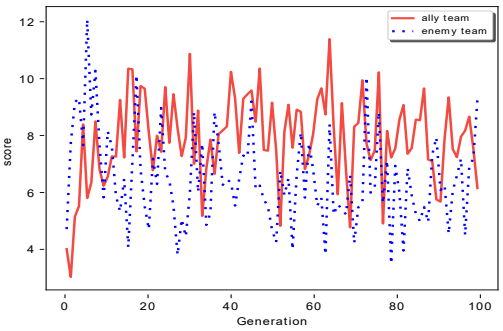
(**d**) Performance of the final rules set

Figure 9. Illustration of the best-performing rules as training progresses. At the initial stages, the blue team had total control over the red team. However, during the final stages of the training, the performance of the red surpasses that of the blue.

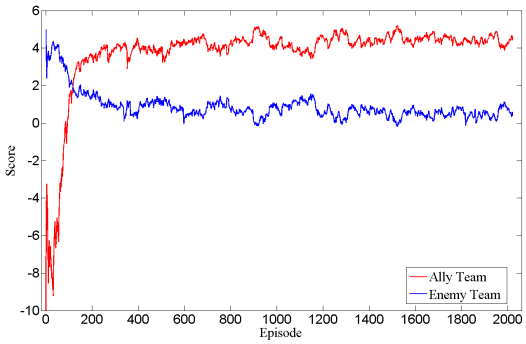
The observed simulation results of both teams during training are presented in Figure 10. As can be seen in Figure 10a,c, the blue force was performing better than the red force during the first generation of rules. However, after 10 and 13 generations, respectively, the red force began to outperform the blue force in the environment, and this is what caused the red force performance to rise in Figure 10a,c. In Figure 10b,d, we compare the highest score attained by both teams in each generation. Furthermore, the same analogy can be drawn from this graph, as in the the initial generations, the blue force seemed to be scoring high as compared to the counterpart red force.



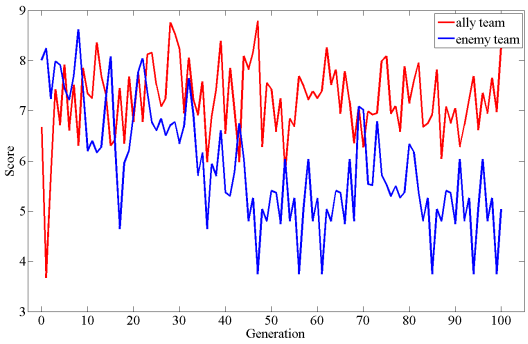
(**a**) Performance of teams during training of six boats.



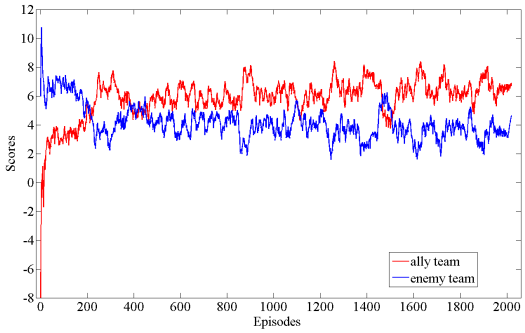
(**b**) Best performance of teams per generations of rules of six boats



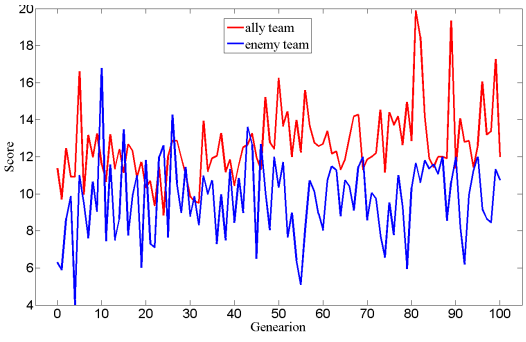
(**c**) team size = 3 boats; sinking an enemy = 1 pts; island = 2 pt.



(**d**) Generational performance of three boats

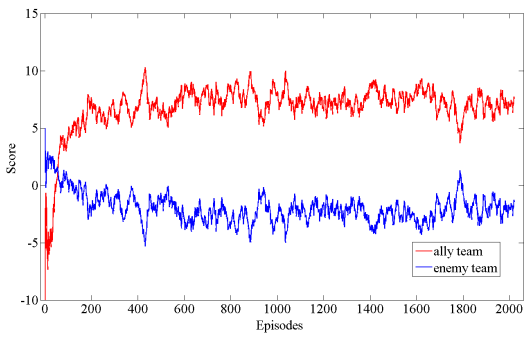


(**e**) Team size = 10, firing range, detection range, and firing angle = 80% of that of the enemy, respectively

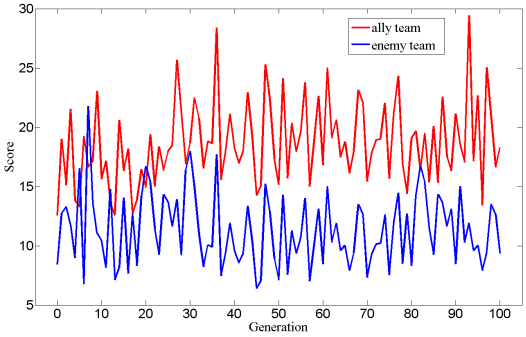


(**f**) High score per generation of 10 boats

Figure 10. *Cont.*



(**g**) boats = 15; sinking an enemy = 3 pts; island = 1 pt, ally firing range = 80% of enemy firing range



(**h**) High scores per generation of 15 boats

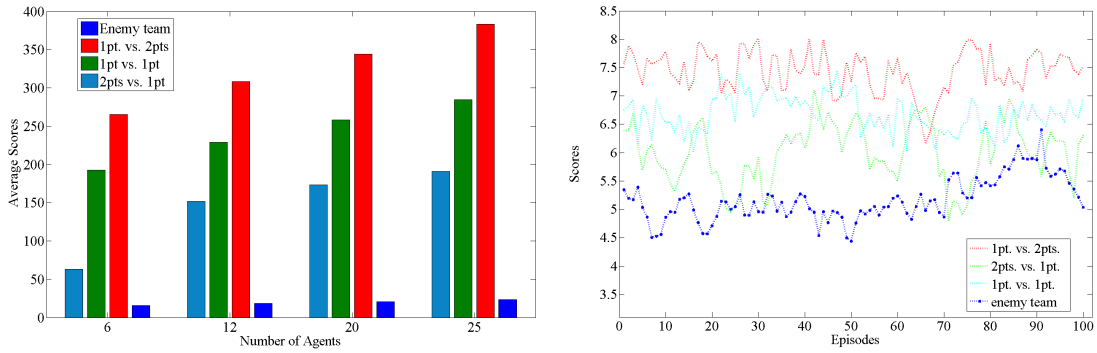
Figure 10. Illustration of the performance of the red and blue teams over 2020 (100 generations) episodes during training. Both series of the episodic performances are averages over 20 different sequences of episodes.

In the third simulation, the capabilities of the red force was constrained below that of its counterpart blue force. With each team consisting of 10 boats, the firing range, detection range, and firing radius of the ally team was set to 80% of the opponent team, respectively. Furthermore, we set the initial population of rules to that of the rule base of Simulation 1. Figure 10e,f shows the performance graphs of both teams.

Finally, a simulation of 15 boats with the objective of destroying more opponent boats while still countering islands was conducted. To this end, we assigned three points for destroying opponent boats and a point for conquering an island. All other parameters were reset to the initial fixed parameters with the exception of the firing range, which was maintained as 80% of the opponents' firing range. The initial population of rules in this scenario was the final population rules' set from the scenario. The performance graph is shown in Figure 10g,h.

The observed performance of these KBs run over one hundred episodes against the enemy team is presented in Figure 11. The blue line in Figure 11a represents the best performance of the enemy team in the three cases against the ally team's performances in all cases. In Figure 11b, the average scores over 100 episodes with increasing number of boats in a team is illustrated. The blue bar is the average scores of the the enemy teams, while the rest of the bars represent the performance of the ally team with varied team size and point allocation.

Whilst the blue force's performance degraded, the red force's reasoner was able to maintain a set of KBs that achieved relatively high performance for most parts of the entire simulation of all training scenarios. Comparing the corresponding episode scores of the final KB of the red force against the blue force rule base used during training revealed that the trained red force was better in at least 82% of all cases, as can be seen in Figure 11. The simulation results demonstrate the feasibility of our approach for multi-agent KB learning and the ability to converge to a stable action selection.



(**a**) Performance of the learned sets of KBs with increasing team size.

(**b**) Performance of the learned set of KBs with varying rewards for islands and enemy sinking.

Figure 11. Performance validation of the best-performing rules of the red force after training. Blue represents the highest performance of the blue force. The red is the performance when two points are assigned to destroying enemy boats and a point for conquering an island. The green is the scores obtained when the value of an island is two points and the destroying opponent boat fetches a point; while the light blue is the case when both conquering an island and destroying the opponent is worth a point.

5. Conclusions and Future Work

In this study, we have presented an unmanned multi-surface vehicle training approach for complex control. We used team learning where a central learner controls the agents in an environment modeled as a partially-observable stochastic game. We modeled the control problem as a decomposed FIS and have provided practical ways for constructing and learning the FIS KB. Our proposed framework enables the usage of different FIS decompositions for a complex control problem with minimal or no modification to the reasoner implementation. It also enables the incorporation of agent-induced non-stationarity awareness in the learning process and a resource model for gathering agents' local observations for off-line learning tasks. Our contributions enable multi-agent control to be performed in domains where no historic data are unavailable for training, but the desired system behavior can be specified as a function of the agents' performance. The multi-surface vehicle island-conquering case study demonstrates the feasibility and convergence properties of this study. Our future works will focus on improving the performance of our approach using multi-agent deep reinforcement learning techniques with the gathered agents' observations.

Author Contributions: Conceptualization, B.Z.; Investigation, W.H.; Methodology, J.L.; Software, W.R.; Writing—original draft, Q.W. and Y.X.; Writing—review & editing, J.L.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Weyns, D.; Van Dyke Parunak, H.; Michel, F.; Holvoet, T.; Ferber, J. Environments for Multiagent Systems State-of-the-Art and Research Challenges. In *Environments for Multi-Agent Systems*; Weyns, D., Van Dyke Parunak, H., Michel, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 1–47.
2. Ernest, N.D. Genetic Fuzzy Trees for Intelligent Control of Unmanned Combat Aerial Vehicles. Ph.D. Thesis, University of Cincinnati, Cincinnati, OH, USA, 2015.
3. Magessi, N.T.; Antunes, L. Modelling Agents' Perception: Issues and Challenges in Multi-agents Based Systems. In *Progress in Artificial Intelligence*; Pereira, F., Machado, P., Costa, E., Cardoso, A., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 687–695.

4. Weyns, D.; Michel, F.; Parunak, V.; Boissier, O.; Schumacher, M.; Ricci, A.; Brandao, A.; Carrascosa, C.; Dikenelli, O.; Galland, S.; et al. Agent Environments for Multi-agent Systems—A Research Roadmap. In *Introduction and Challenges of Environment Architectures for Collective Intelligence Systems*; Springer-Verlag: New York, NY, USA, 2015, doi:10.1007/978-3-319-23850-0_1.
5. Pipattanasomporn, M.; Feroze, H.; Rahman, S. Multi-agent systems in a distributed smart grid: Design and implementation. In Proceedings of the 2009 IEEE/PES Power Systems Conference and Exposition, Seattle, WA, USA, 15–18 March 2009; pp. 1–8, doi:10.1109/PSCE.2009.4840087. [CrossRef]
6. Valogianni, K.; Ketter, W.; Collins, J. A Multiagent Approach to Variable-Rate Electric Vehicle Charging Coordination. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15), Istanbul, Turkey, 4–8 May 2015; pp. 1131–1139.
7. Riedmiller, M.; Moore, A.; Schneider, J. Reinforcement Learning for Cooperating and Communicating Reactive Agents in Electrical Power Grids. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 137–149.
8. Pita, J.; Jain, M.; Ordóñez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; Kraus, S. Using Game Theory for Los Angeles Airport Security. *AI Mag.* **2009**, *30*, 43–57. [CrossRef]
9. Stone, P.; Veloso, M. Multiagent Systems: A Survey from a Machine Learning Perspective. *Auton. Robots* **2000**, *8*, 345–383, doi:10.1023/A:1008942012299. [CrossRef]
10. Crites, R.H.; Barto, A.G. Elevator Group Control Using Multiple Reinforcement Learning Agents. *Mach. Learn.* **1998**, *33*, 235–262, doi:10.1023/A:1007518724497. [CrossRef]
11. Busoniu, L.; Babuska, R.; Schutter, B.D. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Part C* **2008**, *38*, 156–172, doi:10.1109/TSMCC.2007.913919. [CrossRef]
12. Bibuli, M.; Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D.; Khan, A. A Two Layered Optimal Approach towards Cooperative Motion Planning of Unmanned Surface Vehicles in a Constrained Maritime Environment. *IFAC-PapersOnLine* **2018**, *51*, 378–383, doi:10.1016/j.ifacol.2018.09.458. [CrossRef]
13. Polvara, R.; Patacchiola, M.; Sharma, S.; Wan, J.; Manning, A.; Sutton, R.; Cangelosi, A. Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 115–123, doi:10.1109/ICUAS.2018.8453449. [CrossRef]
14. Obst, O. Using a Planner for Coordination of Multiagent Team Behavior. In *Programming Multi-Agent Systems*; Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 90–100.
15. Nalepka, P.; Kallen, R.W.; Chemero, A.; Saltzman, E.; Richardson, M.J. Herd Those Sheep: Emergent Multiagent Coordination and Behavioral-Mode Switching. *Psychol. Sci.* **2017**, *28*, 630–650, doi:10.1177/0956797617692107. [CrossRef] [PubMed]
16. Foerster, J.; Assael, I.A.; de Freitas, N.; Whiteson, S. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 29*; Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; pp. 2137–2145.
17. Heins, P.H.; Jones, B.L.; Taunton, D.J. Design and validation of an unmanned surface vehicle simulation model. *Appl. Math. Model.* **2017**, *48*, 749–774, doi:10.1016/j.apm.2017.02.028. [CrossRef]
18. Sonnenburg, C.R.; Woolsey, C. Modeling, Identification, and Control of an Unmanned Surface Vehicle. *J. Field Robot.* **2013**, *30*, 371–398, doi:10.1002/rob.21452. [CrossRef]
19. Yue, J.; Ren, G.; Liang, X.; Qi, X.W.; Li, G.T. Motion Modeling and Simulation of High-Speed Unmanned Surface Vehicle. Frontiers of Manufacturing and Design Science. *Appl. Mech. Mater.* **2011**, *44–47*, 1588–1592, doi:10.4028/www.scientific.net/AMM.44-47.1588. [CrossRef]
20. Liu, Z.; Zhang, Y.; Yu, X.; Yuan, C. Unmanned surface vehicles: An overview of developments and challenges. *Ann. Rev. Control* **2016**, *41*, 71–93, doi:10.1016/j.arcontrol.2016.04.018. [CrossRef]
21. Yan, R.j.; Pang, S.; Sun, H.b.; Pang, Y.j. Development and missions of unmanned surface vehicle. *J. Mar. Sci. Appl.* **2010**, *9*, 451–457, doi:10.1007/s11804-010-1033-2. [CrossRef]
22. IsraelDefense. Rafael's Protector USV Conducts Successful Missile Firing Demo for NATO. 2018. Available online: https://www.israeldefense.co.il/en/node/34530 (accessed on 14 March 2019).
23. Opall-Rome, B. Israel's Elbit Unveils USV for Anti-Sub, Anti-Mine Missions. 2016. Available online: https://www.defensenews.com/naval/2016/02/08/israels-elbit-unveils-usv-for-anti-sub-anti-mine-missions/ (accessed on 14 March 2019).

24. Schnoor, R.T. Modularized unmanned vehicle packages for the littoral combat ship mine countermeasures missions. In Proceedings of the Oceans 2003: Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No. 03CH37492), San Diego, CA, USA, 22–26 September 2003; Volume 3, pp. 1437–1439, doi:10.1109/OCEANS.2003.178073. [CrossRef]
25. Manley, J. Unmanned surface vehicles, 15 years of development. In Proceedings of the OCEANS 2008, Quebec City, QC, Canada, 15–18 September 2008; pp. 1–4, doi:10.1109/OCEANS.2008.5152052. [CrossRef]
26. Wen, G.X. Neural-network-based adaptive leader-following consensus control for second-order non-linear multi-agent systems. *IET Control Theory Appl.* **2015**, *9*, 1927–1934. [CrossRef]
27. Cordón, Ó.; Herrera, F.; Hoffmann, F.; Magdalena, L. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning Of Fuzzy Knowledge Bases*; World Scientific: London, UK, 2004; Volume 141, pp. 161–162, doi:10.1016/S0165-0114(03)00262-8.
28. Holland, J.H.; Reitman, J.S. Cognitive Systems Based on Adaptive Algorithms. *SIGART Bull.* **1977**, 49, doi:10.1145/1045343.1045373. [CrossRef]
29. Smith, S.F. A Learning System Based on Genetic Adaptive Algorithms. Ph.D. Thesis, University of Pittsburgh, Pittsburgh, PA, USA, 1980.
30. Herrera, F.; Lozano, M.; Verdegay, J. A learning process for fuzzy control rules using genetic algorithms. *Fuzzy Sets Syst.* **1998**, *100*, 143–158, doi:10.1016/S0165-0114(97)00043-2. [CrossRef]
31. Herrera, F.; Lozano, M.; Verdegay, J.L. Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis. *Artif. Intell. Rev.* **1998**, *12*, 265–319, doi:10.1023/A:1006504901164. [CrossRef]
32. Hernandez-Leal, P.; Kaisers, M.; Baarslag, T.; de Cote, E.M. A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *arXiv* **2017**, arXiv:1707.09183.
33. Claus, C.; Boutilier, C. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI '98/IAAI '98), Madison, WI, USA, 26–30 July 1998; American Association for Artificial Intelligence: Menlo Park, CA, USA, 1998; pp. 746–752.
34. Rada-Vilela, J. Fuzzylite: A Fuzzy Logic Control Library in C++. 2017. Available online: https://pdfs.semanticscholar.org/ec93/4e26ea2950d0f3ab30d31eb8ac239373b4e8.pdf (accessed on 14 March 2019).