UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Novel Fuzzy Systems for Human-Autonomous Agent Teaming

by

Yu-Cheng Chang

Thesis submitted in fulfilment of
the requirements for the degree of

Doctor of Philosophy

under the supervision of
Professor Chin-Teng Lin

Computational Intelligence and Brain Computer Interfaces Lab
Australian Artificial Intelligence Institute
Faculty of Engineering and Information Technology
University of Technology Sydney

February 2021

Declaration of Authorship and Originality

I, Yu-Cheng Chang, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctoral of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

05/02/2021

Acknowledgements

I wish to thank my esteemed supervisor – Professor Chin-Teng (CT) Lin for his invaluable supervision, support and tutelage during the whole PhD candidature period. I also thank Professor CT for offering the opportunities involving in other academic activities, which are necessary to train independent researchers.

Helps and supports from Professor Nikhil R. Pal, Dr. Yu-Kai, Wang, Dr. Ye Shi, and Dr. Zehong Cao are highly appreciated and it is also manifest from the published/revised/submitted/preparing papers enlisted as co-authors.

Furthermore, I would like to thank Dr. Jijoong Kim and Dr. Anna Dostovalova from the Defence Science and Technology Group (DSTG) for their frequent discussions about different aspects of the research. I also pay attributes to other members in the CIBCI lab for a variety of help hard to be categorized but contributing to an enjoyable and beneficial experience.

Lastly, my appreciation also goes out to my family, especially my wife PohWee Song, for their encouragement and support all through my studies.

Abstract

The Multi-agent Teaming (MAT) systems that have been widely applied in many fields provide a novel method for establishing models, conducting the analysis, implementing complex tasks and so on. The agents in an MAT system can be defined as intelligent agents, machine agents and human agents based on a particular task to exhibit flexible behaviours. This research investigates various fuzzy models to resolve the problems of designing MAT systems, such as the coordination of agents, the interpretability of actions and states, high-dimensional data and data privacy.

First, a hierarchical fuzzy logic system that is proposed to deal with the coordination of agents is applied to the simultaneous arrival of multiple mobile agents. The proposed hierarchical fuzzy logic system consists of two levels: a lower-level individual navigation control for obstacle avoidance and a higher-level coordination control to ensure the same time of arrival for all robots at their target; it enables the synchronisation of the agents' arrival times while avoiding collisions with obstacles. Apart from the hierarchical structure, a grouping and merging mechanism is developed to optimise transparent fuzzy sets and integrated into the training process to improve the fuzzy models' interpretability. Additionally, a Multi-objective hybrid GA and PSO (MGAPSO) algorithm is developed to design the hierarchical fuzzy controller efficiently. The MGAPSO leverages the exploring capability of Genetic Algorithm (GA) and the convergence capability of Particle Swarm Optimization (PSO). The simulation results demonstrate that the proposed hierarchical fuzzy controller successfully controls various numbers of robots to navigate and reach the target simultaneously safely. The optimised fuzzy sets can be interpreted by explaining the mining of fuzzy sets and the consequent components.

Moreover, this research also proposes a fuzzy Covert State Transition Diagram (FCOSTD), which provides a mechanism to automatically identify humans' external and covert states that machine agents can understand. The fuzzy-inference mechanism is used to represent the activities of the human states associated with varying behaviours. The proposed system consists of a supervised-learning-based fuzzy network featuring real-world data representing the salient features of human biosignals.

Unsupervised clustering is then conducted on the extracted feature space to determine the human's external and covert states. A state transition diagram is introduced to investigate state change and enable the visualisation of connectivity patterns between every pair of states. We compute the transition probability between every pair of states to represent the relationships between the states. We then apply FCOSTD to real-world Electroencephalography (EEG) data collected from distracted driving experiments. FCOSTD successfully discovers the external and covert states and faithfully reveals the transition of the brain between states and the route of the state change when humans are distracted during a driving task. The experimental results demonstrate that different subjects have similar states and inter-state transition behaviour (establishing the consistency of the system) but different methods for allocating brain resources as different actions are being taken. The discovery of covert brain states offers machine agents the possibility to understand human cognitive states in a human-autonomous system.

Finally, a Distributed Fuzzy Neural Network (D-FNN) model is developed to address data privacy for multiagent decision-makers. The proposed D-FNN model considers consensus for both the antecedent and consequent layers. A novel consensus learning, which involves distributed structure learning and distributed parameter learning, is proposed to handle the D-FNN model. The proposed consensus learning algorithm is built on the well-known alternating direction method of multipliers, which does not exchange local data among agents. The simulation results on popular datasets demonstrate the superiority and effectiveness of the proposed D-FNN model and consensus learning algorithm.

The main contributions of this research are as follows. 1) For multiple-agent coordination, a hierarchical fuzzy system is proposed. This hierarchical fuzzy system consists of two levels and is applied to navigation and simultaneous arrival of mobile agents. 2) Two types of explainable fuzzy systems are proposed. One is the fuzzy system with fuzzy set transparency improvement, which optimises the number of fuzzy sets and reduces the overlap between fuzzy sets. Hence, human agents can understand the rules learned by the fuzzy controller. The second is fuzzy rule information visualisation, which considers the firing strength of fuzzy rules as useful information to extract and visualise the hidden state in the human brain. 3) Finally, the distributed fuzzy system is proposed to resolve the data privacy and high-dimensional data in designing MAT systems. A novel consensus learning is developed for the distributed fuzzy system to learn antecedent and consequent components.

Publications

Related Papers

Y. C. Chang, A. Dostovalova, C. T. Lin and J. Kim, ”Intelligent Multirobot Navigation and Arrival-Time Control Using a Scalable PSO-Optimized Hierarchical Controller”, *Frontiers in Artificial Intelligence*, vol. 3, 2020.

Y. C. Chang, A. Dostovalova, Z. Cao, J. Kim, D. Gibbons and C. T. Lin, “Interpretable Fuzzy Logic Control for Multi-Robot Coordination in a Cluttered Environment”, submitted to *IEEE Trans. on Fuzzy Systems.*

Y. C. Chang, Y. K. Wang, N. R. Pal, C. T. Lin, “Hybrid-Learning-Based Fuzzy-Inference Covert States for Exploring the Dynamics of Brain States”, submitted to *IEEE Trans. on Neural Systems and Rehabilitation Engineering.*

Y. Shi, C. Lin, **Y. C. Chang**, W. Ding, Y. Shi and X. Yao, ”Consensus Learning for Distributed Fuzzy Neural Network in Big Data Environment,” in *IEEE Transactions on Emerging Topics in Computational Intelligence*, June, 2020.

Other Papers

L. Zhang, Y. Shi, **Y. C. Chang** and C. T. Lin, ”Hierarchical fuzzy neural networks with privacy preservation on heterogeneous big data”, *IEEE Transactions on Fuzzy Systems*, pp. 1-1, 2020.

L. W. Ko, Y. C. Lu, H. Bustince, **Y. C. Chang**, et al., ”Multimodal Fuzzy Fusion for Enhancing the Motor-Imagery-Based Brain Computer Interface”, *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 96-106, 2019.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Multi-Agent Teaming

Most Artificial Intelligence (AI) systems are built based on the concept of an agent that can be defined as anything being able to perceive external information and then generate corresponding actions or decisions. However, some systems are designed for many situations that coexist with other agents and interaction protocols for various tasks. Such a system consisting of multiple agents that can form a team and interact is called a Multi-agent Teaming (MAT) system.

MAT systems have been widely used in many applications and provide a novel method for establishing models, conducting analyses, implementing complex tasks and so on. The agents in a MAT system can be defined as intelligent agents, machine agents and human agents based on a particular task to exhibit flexible behaviours [1]. Although MAT systems have beneficial features in large systems as follows [2], 1) increasing reliability and robustness, 2) scalability and flexibility, 3) distributed computation and asynchronous operation, and 4) modular structure of agents.

However, various issues might arise when designing a MAT system for a particular achievement. One of the essential issues is coordination among multiple agents. In a MAT system, the coordination mechanism is considered to prevent conflicts, chaos and anarchy, meet global constraints, and improve overall performance [3]. In other words, the coordination mechanism is used to ensure stability and efficiently achieve the final goal. Centralised coordination approaches have been considered a reliable solution to establish models for all agents. Most of the centralised coordination models rely on the information or data collected from neighbourhood agents, resulting in high-dimensional input and output variables of models. This problem can be resolved by decomposing a task into several subtasks in a hierarchical manner.

Such a hierarchical structure enables all agents to learn a modular solution with subtasks by local information, and then the centralised coordination mechanism receives higher-level information from agents.

The second issue of designing a MAT system is the interpretation of actions and states, that is, to understand agent states and interpret agent behaviour. A lack of interpretation of actions and states might lead a MAT system to understandability and predictability and further impact the coordination among agents [4], [5]. For example, in a system with human and machine agents, humans need to know the rationale of machine agents for making particular decisions to help humans track and interact with machine agents. Additionally, machine agents need to know the states of humans to make effective decisions to achieve the task. This research, therefore, considers an explainable fuzzy system a solution to resolve the above problem.

Finally, data privacy is another issue that should be considered for specific applications, such as hospital service systems and bank networks [6]. Data transition between agents might lead to severe security and privacy issues that expose data security in danger. Such an issue has attracted public attention in recent years. In this case, centralised models are neither practical nor safe. Thus, this research proposes a distributed model that relies on local data and limited communication among agents for decision making. Another advantage of using the distributed model is that it can facilitate designing the MAT system for big data environments; each agent focuses on local information and computes independently.

1.2 Literature Survey

MAT systems, which generally refer to a group of intelligent agents that collaborate to solve particular tasks, are often designed by integrating various computational intelligent techniques, such as fuzzy logic, neural networks, evolution computations and machine learning.

Hierarchical Frameworks. Paper [7] proposed a multiagent fuzzy system that used concrete crack detection and type classification based on image analysis. It defined a Fuzzy Inference System (FIS) as the central intelligent agents to communicate and exchange information with other agents. Lyshevski [8] adapted the hierarchical distributed multiagent system to coordinate and control Unmanned Aerial Vehicles (UAVs). The UAVs are controlled by a hierarchical-level controller that can navigate the UAVs in unstructured environments. For multiagent coordination and navigation, several conventional approaches have also been proposed. Yao and Qi [9] proposed a novel dynamical model to adjust the path length and voyage speed of several Autonomous Underwater Vehicles (AUVs) to achieve their simultaneous arrival at a destination. Misra et al. [10] combined a cooperative localisation technique with a proportional navigation (PN) guidance law to manipulate multiple unmanned vehicles to reach a moving target in a GPS-denied environment simultaneously. Babel [11] developed a multiagent path planning algorithm considering the shortest paths between all pairs of air vehicles and targets, target allocation, and concatenating feasible and suitable short paths, which guarantees that all UAVs arrive at their targets on time and without the risk of mutual collision.

Explainable Models. Additionally, one advantage of fuzzy systems is their inherent interpretability because they can learn knowledge from data and then represent it in fuzzy rules. Such a mechanism can enable a human to understand the inherent knowledge in the model. However, many fuzzy systems do not consider the transparency of fuzzy sets, which may result in highly overlapping and a large number of fuzzy sets. These issues of fuzzy sets will degrade the interpretability of the fuzzy system. Ishibuchi et al. [12] and Cococcioni et al. [13] improved the transparency of fuzzy sets by minimising the number of fuzzy rules and fixing the partition granularity of the input space during the optimisation process. The performance of these approaches is limited since the parameters of fuzzy sets are not optimised. To enhance both the accuracy and interpretability of fuzzy systems simultaneously, several multiobjective optimisation algorithms [14]–[19] have been developed. Studies [14]–[16] set a constraint condition to assign a proper distribution of fuzzy sets in each input variable to preserve the transparency of fuzzy sets after tuning the free parameters. Juang et al. [17] used an online clustering and fuzzy set merging mechanism to build flexible partitions of the input space with distinguishable fuzzy sets and a transparency-oriented objective function to maximise the transparency of fuzzy sets. Furthermore, the measurement of fuzzy rule relevance was proposed to identify fuzzy-based redundancy in feature space, which helps to reduce the complexity of fuzzy rules and trade-off the performance between accuracy and interpretability in multiobjective optimisation algorithms [18], [19].

Distributed Models. To consider data privacy, this research considers the distributed model in MAT systems. Distributed machine learning algorithms, such as distributed extreme learning machines[20], [21], distributed support vector machines[22], [23], and distributed deep neural networks[24], [25], have been widely investigated. In [20], a distributed extreme learning machine with kernels based on MapReduce was proposed to realise its parallel computation. Very recently, decentralized multitask learning based on extreme learning machines was proposed in [21], where Alternating Direction Method of Multipliers (ADMM) [26] was employed in an alternating optimization procedure. It is clear that the computational complexity of this method increases dramatically due to the alternating procedure. The ADMM procedure was also employed in [22], [23] to achieve distributed solutions for support vector machines. In [24], [25], ADMM-based algorithms were proposed to train deep neural networks to avoid gradient-based methods. Furthermore, a decentralised learning algorithm has recently been adapted to train FNNs. Several decentralised algorithms for random weights FNN [27], [28] were developed to deal with distributed streaming data, where the parameters in the antecedent components are randomly selected instead of being estimated. However, distributed FNN models suffer from the curse of dimensionality as the number of fuzzy rules increases exponentially with increasing input space. The proposed distributed algorithms can only assure consensus on the consequent layer instead of both antecedent and consequent layers of the FNN. Specifically, such distributed algorithms are not distributed since multiple agents cannot agree with a single model.

1.3 Research Overview

This research investigated various fuzzy models to resolve the problems of designing MAT systems, shown as the research map in Figure 1.1. The map consists of the four issues of MAT systems mentioned above; each is connected to a solution representing the respective fuzzy model. For multiagent coordination, a hierarchical fuzzy system is proposed and presented in Chapter 2. This hierarchical fuzzy system consists of two levels and is applied to multiple robot navigation control. The lower-level fuzzy controller is for obstacle avoidance, while the higher-level controller coordinates robot agents to ensure the same arrival time for all robots at their destination points.

Additionally, this research proposes two types of explainable fuzzy systems. Chapter 3 presents a fuzzy system with fuzzy set transparency improvement, which optimises the number of fuzzy sets and reduces the overlapping between fuzzy sets. Hence, human agents can understand the rules learned by the fuzzy controller. Chapter 4 presents a framework for fuzzy rule information visualisation, which considers the firing strength of fuzzy rules as useful information to identify humans' external and covert states. Afterwards, a state transition diagram is introduced to investigate state change and enable the visualisation of connectivity patterns between every pair of states. The transition probability between every pair of states is computed to represent the relationships between the states. This mechanism allows human agents to understand machine agents' actions and states and then establish machine-to-human interaction in a human-machine MAT system.

Finally, a distributed fuzzy system is proposed in Chapter 5 to resolve the data privacy and high-dimensional data in designing MAT systems. The proposed consensus learning algorithm is built on multipliers' well-known alternating direction method, restricting data exchange among agents to protect data privacy in privacy-focused MAT systems.

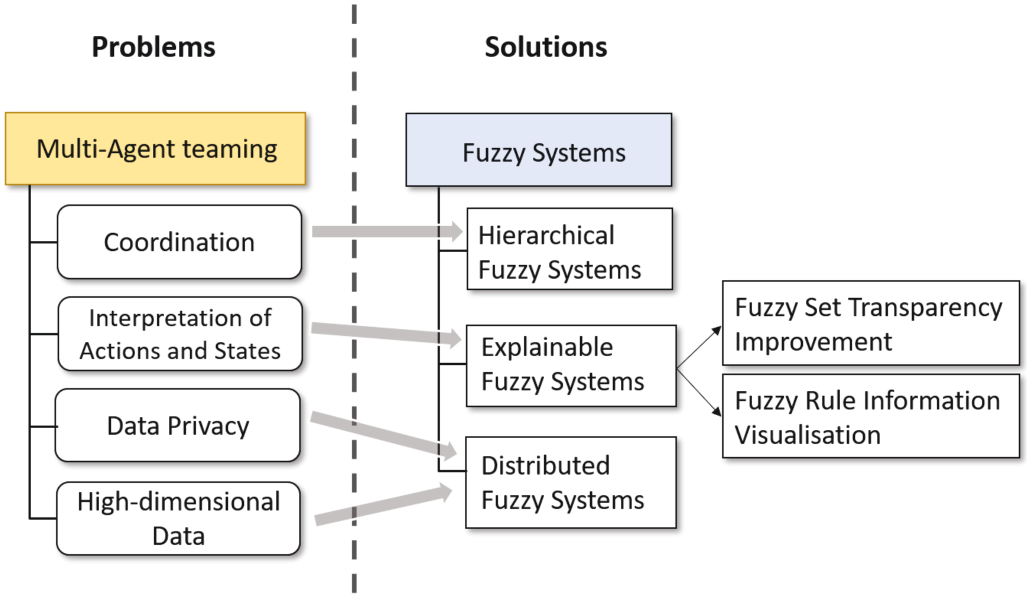

Problems
Solutions
Multi-Agent teaming
Coordination
Interpretation of Actions and States
Data Privacy
High-dimensional Data
Fuzzy Systems
Hierarchical Fuzzy Systems
Explainable Fuzzy Systems
Distributed Fuzzy Systems
Fuzzy Set Transparency Improvement
Fuzzy Rule Information Visualisation

Figure 1.1: Research map of novel fuzzy systems for human/autonomous agent teaming.

Chapter 2

Hierarchical Neural Fuzzy System

2.1 Background

Coordination among agents is a critical problem when designing a MAT system. A suitable coordination mechanism can provide the system with stability and reliability, and improve the overall performance. Recent years, centralised coordination approaches [9]–[11], [29], [30] has been used to coordinate multiple agents across multiple disciplines. However, centralised coordination approaches might result in high-dimensional input and output variables of the system due to the considerable information collected from neighbourhood agents. Therefore, to overcome the above issue, This research considers a hierarchical structure to decompose a task in several sub-tasks in a hierarchical manner. Such a hierarchical structure enables all agents to learn a modular solution with local information for sub-tasks, and then the centralised coordination mechanism can learn from higher-level information collected from modularised agents.

For this research, the designed a MAT system is applied to a multi-agent navigation task to verify and demonstrate coordination performance. One aim of multi-agent navigation task is to guide mobile agents moving between obstacles to reach their targets from a starting point with collision-free performance. Apart from collision-free movement, this research also considers arrival-time control as a sub-task of the multi-agent navigation. More specifically, the mobile agent need to move towards their targets with collision-free motion in a timely manner. In the experiment, the mobile agents are defined as mobile robots that equipped sensors and actuators with noisy and uncertain signals. Therefore, Fuzzy Logic Systems (FLSs) [29], [31]–[37] have been used in automated navigation tasks to enhance the robot control quality.

Fuzzy logic provides a robust solution with anti-noise ability to remove the uncertainty. However, the performance of FLSs depends on the design of the membership function and efficient rules, which often require a considerable amount of time to analyse the experimental input and output data. Machine learning technology, therefore, has been used for fuzzy system design. Zhu and Yang [31] and Pothal and Parhi [29], respectively, exploit supervised learning to train neuro-fuzzy models for one or multiple robots to perform navigation tasks. The precise input-output training data should be collected in advance for supervised learning. To reduce the training effort, evolutionary algorithms have been used to design FLSs. Two popular optimisation algorithms are Genetic Algorithm (GA) [37]–[40] and Particle Swarm Optimization (PSO) [32], [41], [42]. These two optimisation approaches can be easily applied to the design of FLSs because they can be formulated as optimisation problems by defining a metric for the solution performance evaluation.

This research develops a fuzzy-based control system with two levels: lower-level individual navigation control for obstacle avoidance and higher-level coordination to ensure the same time of arrival for all robots at their destination points. FLSs [32], [43], [44], combining the Takagi–Sugeno–Kang (TSK) FIS with a derivative-free global optimisation technique, is used to design the fuzzy "IF-THEN" rules and tune the parameters of the membership functions. The controllers are trained in a cascading manner. In the first phase of training, we employ the PSO algorithm [45] to optimize the fuzzy rules that comprise individual navigation control. The role of this controller is to generate the motion direction command that steers robots away from obstacles and towards the target location based on their sensory inputs (each robot is equipped with a laser ranger) and information about the target location. In the second phase of training, the same technique is used for multiple robots to learn how they need to coordinate with each other to reach their targets at the same time. The coordination controller controls both the moving speeds and moving direction of each robot to achieve the simultaneous target-reaching task. We develop a fuzzy-logic-based coordinator and recurrent-based coordinator. The recurrent-based coordinator includes a Long-Short-Term Memory (LSTM) block [46] that take the input variables and produce the output variables identical to the fuzzy-logic-based. Both fuzzy-logic-based and recurrent-based coordinator are designed by PSO. The Webots software [47] is used as a physics-based robot simulation environment for the training and testing of the proposed solutions.

2.2 The Proposed Models

This section describes the proposed models for multi-robot navigation and arrival-time control. Figure 2.1 shows the configuration. There are two hierarchical levels of the control module. The lower-level controllers are robot navigation controllers for each single mobile robot. They enable each robot to perform collision-free navigation. The higher-level controller is a Multi-agent Coordinator (MAC), which coordinates the robots' speed and direction so that they reach their targets at the same time.

In the proposed model, each navigation controller controls a robot, and all navigation controllers share an identical structure and a set of parameters. The navigation controller receives the adjustment angle, i.e., $\theta_{adj}(t)$, from the output of the MAC, the distances between the robot and nearby obstacles, i.e., $\vec{L}(t)$, from a 2D lidar and the direction angle to the target from the robot, i.e., $\theta_{goal}(t)$. Specifically, a 2D lidar rangefinder on the front of the robot scans from 0° to 180° and outputs $\vec{L}(t) = (L_1, \ldots, L_8)$, which are the minimum distances to any obstacle in each of the eight sectors (Figure 2.2). The output of the navigation controller is the motion direction of the robot. To achieve collision-free navigation, a Fuzzy Logic Controller (FLC) is added to the navigation controller. The robot avoids obstacles using a boundary-following (BF) behaviour. A navigation controller decides what the robot should do at each control time step.

The MAC is a centralized controller used to determine the speed and direction of each robot for the next control time step with five inputs: $D_{goal}^{rank_1}(t-1)$, $D_{goal}^{rank_n}(t-1)$, $v_r^{rank_1}(t-1)$, $v_r^{rank_n}(t-1)$, and $\Delta D(t-1)$. For each control loop, the robots are ranked in ascending distance from the target. Accordingly, $robot_{rank_1}$ is the robot closest to the target, and $robot_{rank_n}$ is the robot farthest away. $D_{goal}^{rank_1}$ and $D_{goal}^{rank_n}$ are, respectively, the distances from $robot_{rank_1}$ and $robot_{rank_n}$ to the target. $v_r^{rank_1}$ and $v_r^{rank_n}$ are, respectively, the speeds of $robot_{rank_1}$ and $robot_{rank_n}$. Finally, $\Delta D(t-1) = D_{goal}^{rank_n}(t-1) - D_{goal}^{rank_1}(t-1)$. The outputs of the MAC to the navigation controllers are the speed $v_r(t)$ and heading angle $\theta_{adj}(t)$ for each robot, $rank_1, \ldots, rank_n$. We developed two types of MAC, one with a fuzzy-logic-based model and the other with a recurrent-based model. The fuzzy-logic-based MAC is implemented by an FLC, while the recurrent-based model uses LSTM. The details of the proposed models are introduced in the following sections.

2.2.1 Fuzzy-Logic-Based Multiple Robot Coordinator

The proposed fuzzy-logic-based MAC is responsible for speed regulation and heading angle adjustment. The robot speed is changed at each control time step. If a

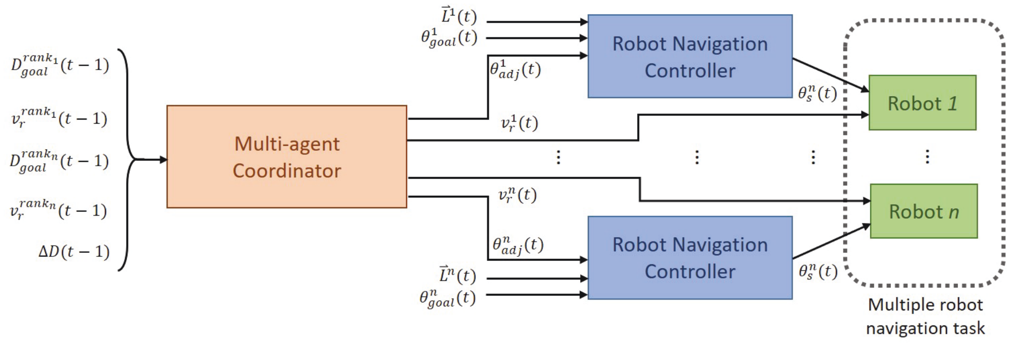


Figure 2.1: The block diagram of the control configuration for multi-robot navigation and arrival-time control.

robot is much closer to the target than the other robots, a heading angle adjustment is made so that it moves away from the target; otherwise, it will need to stop and wait for the others.

Speed Regulation For each control loop, n robots are ranked based on their distance to the target in ascending order. An FLC called FLC_{SR} is used for robot speed regulation, which directly controls $robot_{rank_1}$ and $robot_{rank_n}$. The outputs of FLC_{SR} are speed factors α_1 and α_n for these two robots, which are used to increase or decrease their speeds. For the remaining robots ($robot_{rank_2}, \ldots, robot_{rank_{(n-1)}}$), the speed scale factors $\alpha_2, \ldots, \alpha_{(n-1)}$ are generated by the following interpolation process:

$$\alpha_i = \frac{i-1}{n-1}(\alpha_n - \alpha_1) + \alpha_1, \tag{2.1}$$

where $i = 2, \ldots, n-1$ and $\alpha_i \in [0.5, 1.5]$. The speeds have upper and lower bounds, which define a safe operating region. The robots are not allowed to stop. The speeds for the robots at control time step t are given by:

$$v_r^i(t) = \alpha_i v_r^i(t-1), \quad i = 1, \ldots, n. \tag{2.2}$$

FLC_{SR} uses zero-order TSK fuzzy IF-THEN rules with the form:

$$R_j^{SR}: \text{If } x_1 \text{ is } A_{1j} \text{ And} \ldots \text{ And } x_5 \text{ is } A_{5j} \text{ Then } y \text{ is } a_j, \tag{2.3}$$

where $x_1, \ldots, x_5$ correspond to the input variables: $D_{goal}^{rank_1}(t-1)$, $D_{goal}^{rank_n}(t-1)$, $v_r^{rank_1}(t-1)$, $v_r^{rank_n}(t-1)$, and $\Delta D(t-1)$; $A_{i1}, \ldots, A_{i5}$ are fuzzy sets; and $\vec{a}_i = (a_{i1}, a_{i2})$

is a real vector. Here, we use a Gaussian membership function. Thus, A_{ij} is given by

$$\mu_{ij}\left(x_i\right) = exp\left\{-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right\} \tag{2.4}$$

where m_{ij} and σ_{ij} represent the centre and width of the fuzzy set A_{ij}, respectively. The firing strength of rule R_j^{SR} is obtained by implementing the following algebraic product:

$$\Phi_j = \prod_{i=1}^{M} \mu_{ij}\left(x_i\right), \tag{2.5}$$

where M is the dimension of the input variable, e.g., $M = 5$. Suppose that FLC_{SR} has r rules. An output $\overrightarrow{y} = (y_1, y_2) = (\alpha_1, \alpha_n)$ can be obtained using the weighted average defuzzification method:

$$\overrightarrow{y} = \frac{\sum_{j=1}^{r} \Phi_j a_j}{\sum_{j=1}^{r} \Phi_j}. \tag{2.6}$$

Heading Angle Adjustment To make the lengths of the robot paths roughly equal, the MAC adjusts each robot's heading angle as part of its arrival-time control. An adjusted heading angle for robot i at control time step t is calculated by:

$$\theta_{adj}^i(t) = \theta_{goal}^i(t) + \beta_i \theta_{max\_adj}, \tag{2.7}$$

where β_i is a scale factor that determines the strength of the heading angle adjustment, θ_{goal}^i is the search angle for robot i, and θ_{max_adj} is the maximum angle in changing the direction of robot i, e.g., $\theta_{max\_adj} = 90^\circ$. $\theta_{goal}^i = \theta_t - \theta_{front}^i$ is the deviation between the target angle θ_t and the robot orientation angle θ_{front}^i, as shown in Figure 2.3. β_i is calculated from $D_{goal}^{rank_n}$, D_{goal}^i and α_i as follows:

$$\beta_i = \sqrt{\frac{1}{\alpha_i}\left(1 - \frac{D_{goal}^i(t-1)}{D_{goal}^{rank_n}(t-1)}\right)} \tag{2.8}$$

As β_i increases, robot i is guided away from the target; on the other hand, as β_i decreases, robot i is guided towards the target. This adjustment keeps changing the search behaviour of each robot, except for the furthest $robot_{rank_n}$, until either $(D_{goal}^{rank_n} - D_{goal}^{rank_1}) < 0.1\ m$ or each robot is within 10 m of the target. Algorithm 1 is an overview of the fuzzy-logic-based MAC for the multi-robot navigation and arrival-time control.

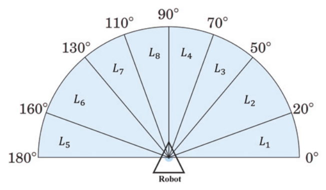


Figure 2.2: Scanning area of the 2D Lidar in the simulation setting.

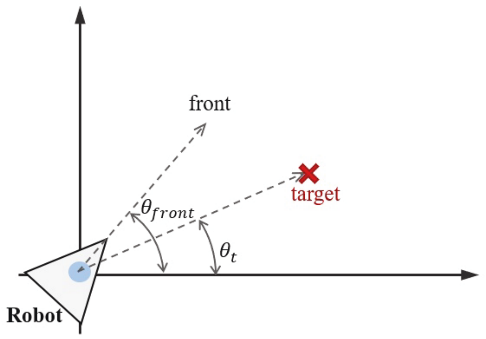


Figure 2.3: Target angle and robot orientation angle.

2.2.2 Recurrent-Based Multiple Robot Coordinator

In addition to the fuzzy-logic-based MAC, we also consider an LSTM-based model because it performs well on problems involving sequential data with long time dependencies. Its memory mechanism allows the use of historical data, which could be useful for optimizing trajectory-related problems. The vanilla version of LSTM is used because it is simple to implement and its performance is close to that of other variants. Figure 2.4 shows the architecture of the LSTM block. In the recurrent-based MAC configuration, we use two LSTM blocks, with input (W_z, W_i, W_f, W_o), recurrent (R_z, R_i, R_f, R_o), peephole (p_i, p_f, p_o), and bias (b_z, b_i, b_i, b_o) weights. The input/output interface of the LSTM controller matches that of the fuzzy-logic-based MAC. Given input $\mathrm{x}^k = (D_{goal}^{rank_1}(t-1),\ D_{goal}^{rank_n}(t-1)\ v_r^{rank_1}(t-1),\ v_r^{rank_n}(t-1),\ \Delta D(t-1)\ )$, the LSTM block forward pass is

$$\begin{aligned}
&\textbf{Block input: } \boldsymbol{z}^k = \boldsymbol{h}\left(\boldsymbol{W}_z \mathbf{x}^k + \boldsymbol{R}_z \mathbf{y}^{k-1} + \boldsymbol{b}_z\right),\\
&\textbf{Input gate: } \boldsymbol{i}^k = \boldsymbol{\sigma}\left(\boldsymbol{W}_i \mathbf{x}^k + \boldsymbol{R}_i \mathbf{y}^{k-1} + \boldsymbol{p}_i \odot \boldsymbol{c}^{k-1} + \boldsymbol{b}_i\right),\\
&\textbf{Forget gate: } \boldsymbol{f}^k = \boldsymbol{\sigma}\left(\boldsymbol{W}_f \mathbf{x}^k + \boldsymbol{R}_f \mathbf{y}^{k-1} + \boldsymbol{p}_f \odot \boldsymbol{c}^{k-1} + \boldsymbol{b}_i\right),\\
&\textbf{Cell: } \boldsymbol{c}^k = \boldsymbol{z}^k \odot \boldsymbol{i}^k + \boldsymbol{c}^{k-1} \odot \boldsymbol{f}^k,\\
&\textbf{Output gate: } \boldsymbol{o}^k = \boldsymbol{\sigma}\left(\boldsymbol{W}_o \mathbf{x}^k + \boldsymbol{R}_o \mathbf{y}^{k-1} + \boldsymbol{p}_o \odot \boldsymbol{c}^k + \boldsymbol{b}_o\right),\\
&\textbf{Block output: } \boldsymbol{y}^k = \boldsymbol{h}\left(\boldsymbol{c}^k\right) \odot \boldsymbol{o}^k,
\end{aligned} \tag{2.9}$$

where σ is the logistic sigmoid function used for gate activation and h is the hyperbolic tangent function for the block input/output activation. During the training process, all bias weights are set to 0.5.

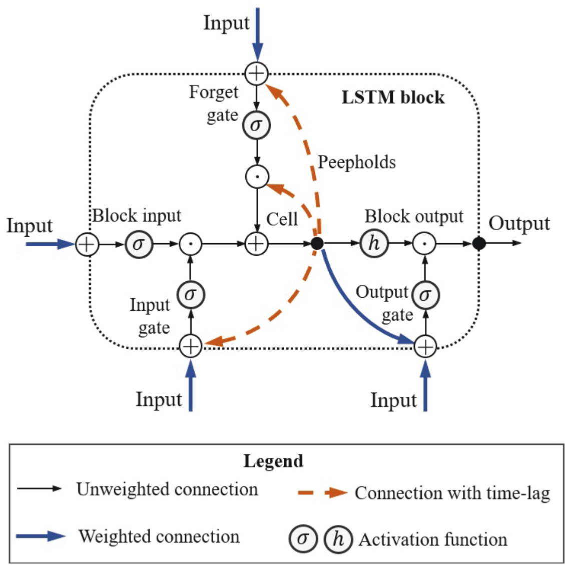


Figure 2.4: Long-short-term memory block.

2.2.3 Single Robot Navigation

In this study, robot behaviour cam be simply categorised in target-searching (TS) and obstacle avoidance. The obstacle avoidance behaviour is implemented by performing the left BF behaviour or right BF behaviour. The behaviour a robot executing is determined by a navigation controller according to the robot's current position, target position and real-time outputs from the 2D lidar sensor.

Navigation Controller In the simulation for robot navigation, the mobile robot is equipped with a 2D lidar sensor that scans the area in front of the robot from right (0±) to left (180±). The coverage area is divided into eight sectors $L_1, \ldots, L_8$, as shown in Figure 2.2. The navigation controller uses a simple logic proposed in [32] to switch between the TS behaviour and the left and right BF behaviours. If no obstacles are detected within the sensing range of the robot's lidar, then the robot starts moving directly towards the target. Figure 2.5 shows the logic of the behaviour selection based on the robot-target distance and timestep counter. When the robot switches its behaviour from TS to BF, the distance d_1 between the robot and the target is recorded, and the step counter c_{step} is set to zero. At the location where the robot decides to switch its behaviour from BF to TS, the distance d_2 between the robot and the target is calculated. If $d_1 > d_2$, or if the step counter $c_{step} > 100$, the robot keeps the original BF behaviour; otherwise, the robot switches from BF to TS.

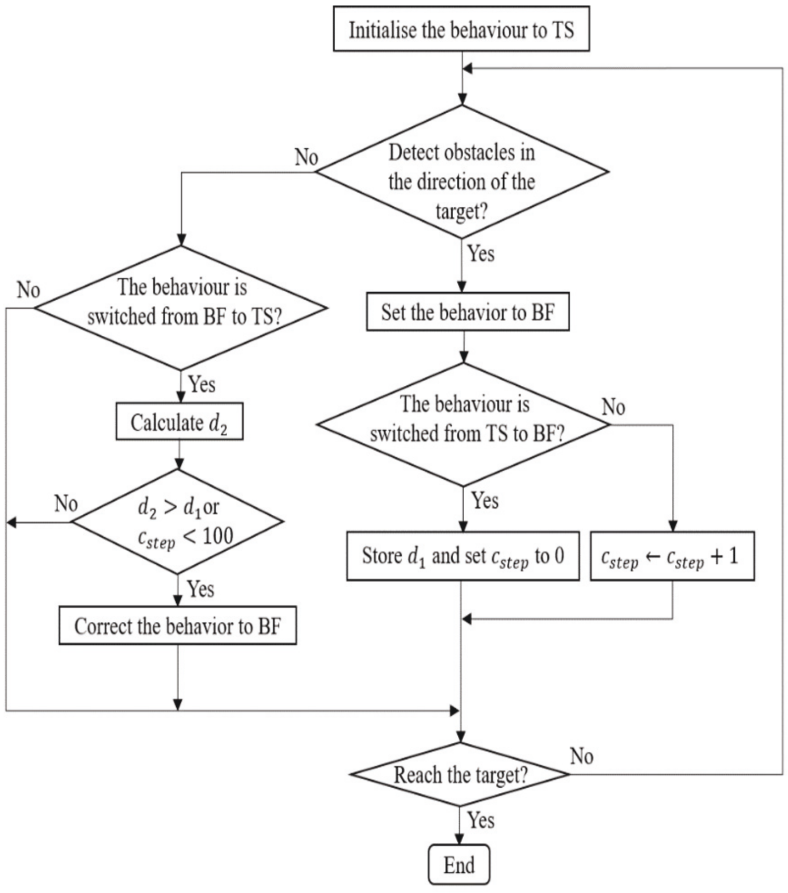

Initialise the behaviour to TS
Detect obstacles in the direction of the target?
No
Yes
Set the behavior to BF
The behaviour is switched from TS to BF?
No
Yes
Store d_1 and set c_{step} to 0
$c_{step} \leftarrow c_{step} + 1$
No
The behaviour is switched from BF to TS?
Yes
Calculate d_2
No
$d_2 > d_1$ or $c_{step} < 100$
Yes
Correct the behavior to BF
Reach the target?
No
Yes
End

Figure 2.5: The block diagram of the navigation control.

This timestep constraint prevents the robot from immediately switching between the TS and BF behaviours.

The control of the robot performing BF in the navigation task is implemented by two fuzzy controllers: the left BF controller and the right BF controller. The left BF controller is used when the robot is close to an obstacle in the left-hand-side region, whereas the right BF controller is used for the right-hand-side region. The number of rules for the right BF controller is identical to that for the left BF controller. The rules for the right BF behaviour share the same antecedent part as those for the left BF behaviour except that the left sensor inputs $L_5, \ldots, L_8$ are changed to right sensor inputs $L_1, \ldots, L_4$. For the rule consequent part, the steering angle in each rule for the right BF behaviour is simply the reverse of that for the left BF behaviour. For example, suppose that the i-th rule in the left BF controller is represented as follows:

$$R_i^{left} : \text{If } L_5 \text{ is } B_{i1} \text{ And } L_6 \text{ is } B_{i2} \ldots \text{ And } L_8 \text{ is } B_{i4} \text{ Then } \theta_B F \text{ is } \theta_i, \tag{2.10}$$

Then, the corresponding rule for the right BF controller is:

$$R_i^{right} : \text{If } L_1 \text{ is } B_{i1} \text{ And } L_2 \text{ is } B_{i2} \ldots \text{ And } L_4 \text{ is } B_{i4} \text{ Then } \theta_B F \text{ is } -\theta_i, \tag{2.11}$$

where $B_{i1}, \ldots, B_{i4}$ are fuzzy sets defined by a Gaussian membership function and given as equation (2.4). The output of the left BF controller is computed according to equation (2.6) with a singleton consequent value $a_i = \theta_i$; similarly, the output of the right BF controller has a consequent value $a_i = -\theta_i$. During the navigation task, the robot should decide to carry out either the left or right BF behaviour at each control time step.

2.3 Training Strategy and Simulation Configuration

In this study, both the fuzzy-logic-based MAC and recurrent-based MAC are trained in a cascading manner. First, we train the BF controller to perform collision-free navigation toward the target. In the second phase of training, the fuzzy-logic-based MAC and recurrent-based MAC learn to coordinate a group of BF-controller-equipped robots to arrive at a target at the same time. The PSO algorithm [45] is used to optimize the tuneable parameters of all controllers.

2.3.1 Particle Swarm optimisation

PSO is a swarm intelligence optimisation approach in which each solution is represented as a particle [45]. Each particle has a position, represented by the vector s_i. The swarm in PSO is initialised with a population of random solutions. A swarm of particles moves through the solution space, and the velocity of each particle is represented by the vector v_i. The performance of a particle is measured by a fitness function f, which is evaluated using s_i. Each particle keeps track of its own best position p_i, which is associated with the best fitness that the particle has achieved. Additionally, it is guided towards the best position found by any member of the swarm (the global best position g). For particle i at iteration t, each element k of the new velocity can be calculated as

$$v_i^{(t)}(k) = wv_i^{(t-1)}(k) + c_1 r_1 \left(p_i(k) - s_i^{(t-1)}(k)\right) + c_2 r_2 \left(g(k) - s_i^{(t-1)}(k)\right), \tag{2.12}$$

Algorithm 1 Pseudocode for the multi-robot navigation and arrival-time control.

```
for all $robot_i$ do
    Initialise $robot_i$
    Set the initial position and moving speed.
    Set the position of the target.
end for
Main control loop
while stop conditions have not been met do
    for all $robot_i$ do
        $L_1^i, \ldots, L_8^i \leftarrow$Lidar outputs
        Compute distance to the target $D_{goal}^i$
    end for
    Sort all robots according to $D_{goal}^i$ in ascending order
    $v_r^1 \leftarrow$ moving speed of $robots_{rank_1}$
    $v_r^n \leftarrow$ moving speed of $robots_{rank_n}$
    $D_{goal}^{rank_1} \leftarrow$ distance between $robots_{rank_1}$ and the target
    $D_{goal}^{rank_n} \leftarrow$ distance between $robots_{rank_n}$ and the target
    $\Delta D \leftarrow D_{goal}^{rank_n} - D_{goal}^{rank_1}$
    $\alpha_1, \alpha_n \leftarrow \mathbf{FLC}_{SR}(v_r^1, v_r^n, D_{goal}^{rank_1}, D_{goal}^{rank_n}, \Delta D)$
    for $robots_{rank_2}$ to $robots_{rank_{n-1}}$ do
        $\alpha_i \leftarrow$ equation 2.1
    end for
    for all $robot_{rank_i}$ do
        $v_r^{rank_i} \leftarrow$equation 2.2
        if $robot_{rank_i}$ is performing TS behaviour and $\Delta D > 0.1m$ or $D_{goal}^{rank_n} > 10m)$
        then
            $\beta_i \leftarrow$ equation 2.8
            $\theta_{adj}^{rank_i} \leftarrow$ equation 2.7
        else
            $\theta_{adj}^{rank_i} \leftarrow \theta_{goal}^{rank_i}$
        end if
        Update steering angle of $robot_{rank_i}$
        $\theta_r^{rank_i} \leftarrow \mathbf{RobotNavigationController}(L_1^i, \ldots, L_8^i, D_{goal}^{rank_i}, \theta_{adj}^{rank_i})$
    end for
end while
```

where w is the inertia weight, c_1 and c_2 are positive acceleration coefficients, and r_1 and r_2 are uniformly distributed random numbers in the interval $[0, 1]$. All components of vi have lower and upper bounds defined by the geometry of the search space. The new position of each particle is calculated with

$$s_i^{(t)}(k) = s_i^{(t-1)}(k) + v_i^{(t)}(k). \tag{2.13}$$

With a careful choice of the parameters w, c_1, and c_2, equations 2.12 and 2.13 ensure that the particle population clusters around the best solution.

2.3.2 Training Phase 1: Boundary-following Behaviour Learning

Figure 2.6 illustrates the environment for training phase 1. The main goal of this phase is to control the robot with the BF behaviour at a constant speed using the PSO-based fuzzy controller. Without loss of generality, this value is set as 0.4 m/s in this paper. The BF behaviour enables collision-free movement of the robot during navigation. Since only the left BF controller is trained, the distances detected by sectors L_5, L_6, L_7, and L_8 are used and fed as the inputs to the left BF controller. The right BF behaviour is directly available by a slight modification to the learned consequents for the left BF behaviour. The left BF controller output is the steering angle of the robot BF behaviour θ_{BF}, where $\theta_{BF} \in [-3.14,\ 3.14]$ in radians. A positive value of θ_{BF} means a clockwise rotation. The constraints for successful left BF behaviour at each time step during the learning process are

$$\min(L_5, L_6, L_7, L_8) > D_{min}, \text{and } L_5 \leq D_{max}. \tag{2.14}$$

In this simulation, D_{min} and D_{max} are set to 0.5 and 1.5, respectively. The first constraint prevents a collision with the object, and the second constraint prevents the robot from moving too far from the object. In PSO-optimized training phase 1, a particle represents a whole fuzzy controller for the left BF behaviour. The performance of the left BF behaviour is evaluated as follows. The robot moves along the side of an object and stops when one of the constraints in (2.14) is violated, which indicates that the controller has failed. If the robot stops, the total number of control time steps is recorded as $T_{control}$. The fitness function f_{phase_1} for training phase 1 is

$$f_{phase_1} = \frac{1}{T_{control}}. \tag{2.15}$$

A low f_{phase_1} indicates a good left BF behaviour. The control process from when movement starts to when it stops is called a trial. If the left BF behaviour fails, the robot moves back to its initial position for the next trial, and a new fuzzy controller is constructed and evaluated. The learning process is repeated until a successful fuzzy controller is found or the maximum number of iterations is met. A left BF behaviour is deemed successful if it successfully controls the robot for a total of T_{suc} time steps. In training phase 1, T_{suc} is set to 4000 so that the robot moves along the object boundary for over two cycles. The maximum number of iterations is set to 200 for a trial.

2.3.3 Training Phase 2: Multi-robot Navigation and Arrival-time Coordination Learning

In training phase 2, there are three robots moving in a complex environment, as shown in Figure 2.7. Each robot is controlled by the navigation controller, the BF controller of which is optimized in training phase 1. During training, both the fuzzy-logic-based MAC and recurrent-based MAC are applied in the navigation of the three robots so that they reach the target simultaneously. The robots start from different positions and head towards the same target. The performance of the MAC is evaluated using a fitness function f_{phase_2}:

$$f_{phase_2} = w_1 f_1 + w_2 f_2 \tag{2.16}$$

The first term of (2.16), f_1, is used to optimize the difference in the arrival times of $robot_{rank_1}$ and $robot_{rank_n}$:

$$f_1 = \left| T_{rank_1} - T_{rank_n} \right|, \tag{2.17}$$

where T_{rank_1} is the time that $robot_{rank_1}$ takes to reach the target and T_{rank_n} is that for $robot_{rank_n}$. The second term of (2.16), f_2, is used to make the robot move as fast as possible:

$$f_2 = \frac{\left| T_{rank_1} - T_{rank_n} \right|}{2}. \tag{2.18}$$

Here, w_1 and w_2 are set to 10 and 0.1, respectively.

2.4 Simulation Results

2.4.1 Simulation 1 (Boundary-following Behaviour Learning)

This example shows the simulation results of training phase 1 (left BF learning result) obtained using the PSO-optimized FLC . The number of fuzzy rules is set to

10. The simulation environment is shown in Figure 6. The environment is built using Webots 8.5.3 on a platform equipped with an Intel i5-4200H 3.40 GHz CPU, NVIDIA GT 745M 2 GB graphics card, and 8 G 1600 MHz RAM. The learning objective is to find a successful FLC for the left BF behaviour satisfying the constraints in (2.14) for a total of 4000 time steps. The control loop stops when the robot violates the constraint of the left BF FLC . For this optimisation problem, the objective is to design a successful FLC using as few iterations as possible. Figure 2.8 shows the left BF behaviour learning results for all 50 runs. PSO fails to find a successful left BF FLC for one of the 50 runs. The average number of iterations of the PSO needed to find a successful FLC is 13.987.

2.4.2 Simulation 2 (Multi-robot Navigation and Arrival-time Coordination Learning)

The objective of this simulation is to optimize the MAC to enable three robots to navigate and simultaneously reach a target in a cluttered environment, as shown in Figure 2.7. We set the centre point of the map as the origin of coordinate (x, z) = $(0, 0)$, and the target is located at (x, z) = $(-11, -23)$. The initial distance between the target and $robot_1$ is 30.4 m, $robot_2$ is 39.5 m, and $robot_3$ is 43.8 m. The performances of both the fuzzy-logic-based MAC and recurrent-based MAC are evaluated by equation (2.16). A smaller solution to equation (2.16) means that the MAC can move the three robots towards the target as fast as possible and coordinate their arrival-time as precisely as possible. For each evaluation process, the MAC controls all robots until they all reach the target. Once all robots have reached the target, the positions of the robots are set to their initial positions, which are

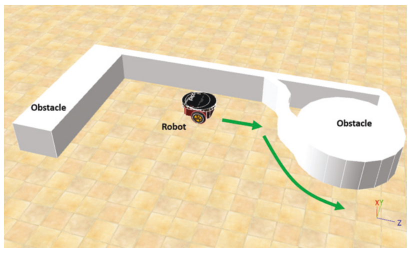


Figure 2.6: The environment for training phase 1.

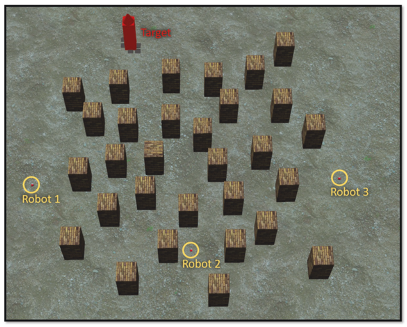


Figure 2.7: Environment for training phase 2.

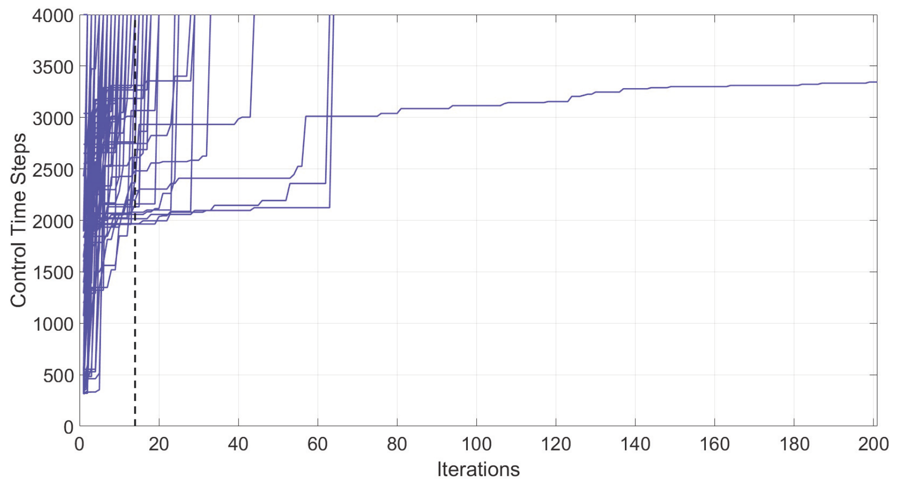


Figure 2.8: Robot left-BF behaviour-learning results for all 50 runs.

Table 2.1: Performance of fuzzy-logic-based and recurrent-based MAC in the training phase 2.

		f_{phase_2}	f_1	f_2
Fuzzy-logic-based MAC	**Average**	57.73	532.0	0.41
	STD	2.97	29.85	0.068
Recurrent-based MAC	**Average**	54.46	524.62	0.2
	STD	3.31	32.42	0.053

fixed during the whole training process. The number of learning iterations is set to 50. The PSO process for training phase 2 involves 50 runs for the statistical evaluation. Figure 2.9 demonstrates the average best-so-far fitness of the MACs. Table 2.1 presents the performances of the fuzzy-logic-based MAC and recurrent-based MAC in training phase 2. The average best-so-far fitness of the fuzzy-logic-based MAC converges at 57.2 (the average value of f_1 is 532.0 time steps, and the average value of f_2 is 0.41 time steps), while the recurrent-based fitness is 54.46 (the average value of f_1 is 524.62 time steps, and the average value of f_2 is 0.20 time steps).

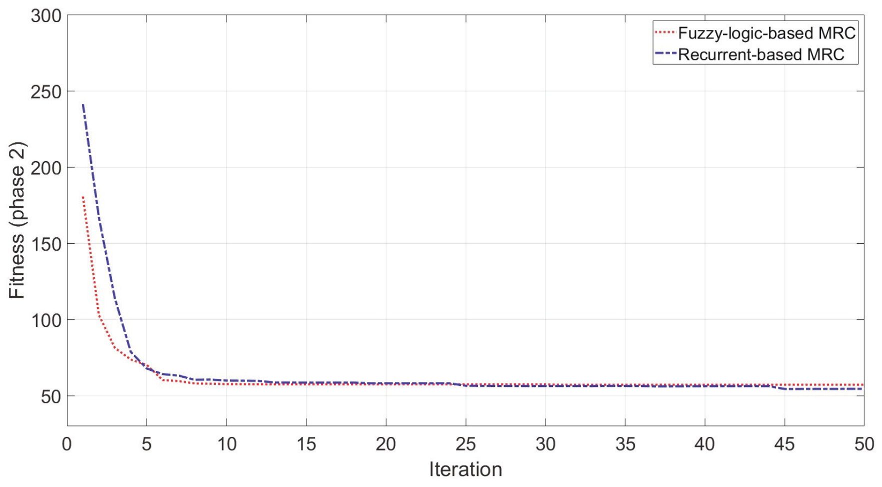


Figure 2.9: Average best-so-far fitness value at each iteration for the MACs during the training phase 2.

2.4.3 Simulation 3 (Multi-robot Navigation and Arrival-time Coordination)

In this simulation, the optimized MACs and BF controller are applied to perform the navigation and arrival-time tasks. We deploy three robots and six robots in a cluttered environment with various starting positions to test the scalability of the optimized MACs, as shown in Figure 2.10 and Figure 2.11, respectively. Both the fuzzy-logic-based and recurrent-based MACs are used in this simulation. To demonstrate the ability of arrival-time coordination, examples of robots controlled without an MAC are included for comparison.

Three-robot Navigation and Arrival-time Control To validate the performance of the proposed control systems, we deploy three robots in a complex environment (see Figure 2.10), with the target building set at (x, z) = $(-19.54, 8.78)$. The initial distance between the target and $robot_1$ is 19.34 m, $robot_2$ is 32.94 m, and $robot_3$ is 39.05 m. Figure 2.12 illustrates the trajectories of the three robots controlled by the fuzzy-logic-based MAC, controlled by the recurrent-based MAC and in the absence of an MAC during the navigation task. Figure 2.13 shows the remaining distance between the target and the three robots. The performance of the arrival-time coordination in the three-robot setting is shown in Table 2.2. The time difference is evaluated by measuring the difference in arrival time of the fastest robot

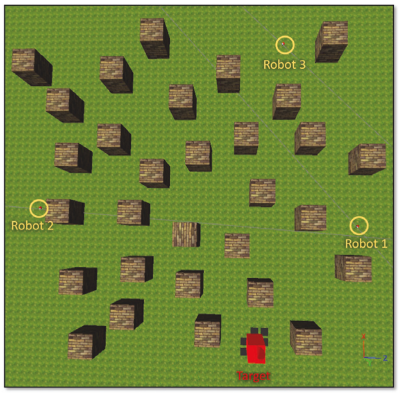


Figure 2.10: An environment setting for three robot's navigation.

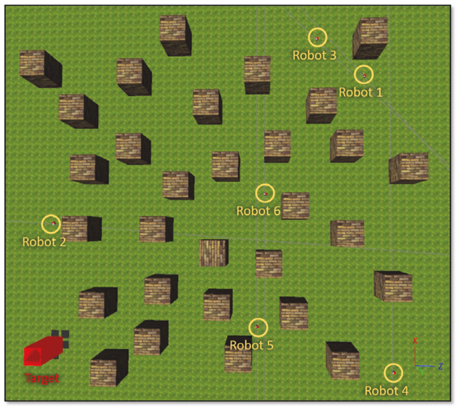


Figure 2.11: An environment setting for six robot's navigation.

and the slowest robot in the simulation. The best achieved time difference under the fuzzy-logic-based MAC is 5 time steps, while it takes 785 time steps for the slowest robot to complete the navigation task. The recurrent-based configuration resulted in an 8-time-step difference between the first and last arriving robots, while it takes 778 time steps for the slowest robot to complete the navigation task. By comparing Figure 2.13(a) and 2.13(b), the robots controlled by the recurrent-based MAC have a faster convergence speed, but the fuzzy-logic-based MAC has a better coordinating ability in this simulation. We further compare with the case without MAC control; see Figures 2.12(c) and 2.13(c). The three robots move directly towards the target without changing their search directions and moving speeds such that $robot_1$ arrives at the target much earlier than the other two robots.

Six-robot Navigation and Arrival-Time Control The reason for us to use the simple interpolation method to decide the velocity scaling factors and search directions of the robots is to increase the scalability of the proposed methods such that they can deal with a different and changing number of robots without retraining the neural networks. To validate the scalability of the proposed methods, we deploy six robots in this simulation, as shown in Figure 2.11, and the target building is set at $(\text{x}, \text{z}) = (-17.71, -20.34)$. The initial distance between the target and $robot_1$ is 50.11 m, $robot_2$ is 15.26 m, $robot_3$ is 49.44 m, $robot_4$ is 41.69 m, $robot_5$ is 24.08 m, and $robot_6$ is 34.27 m. The trajectories of the six robots controlled by each proposed model are illustrated in Figure 2.14. Figure 2.15 shows the remaining distances between the target and the six robots. The performance of the arrival-time coordination in

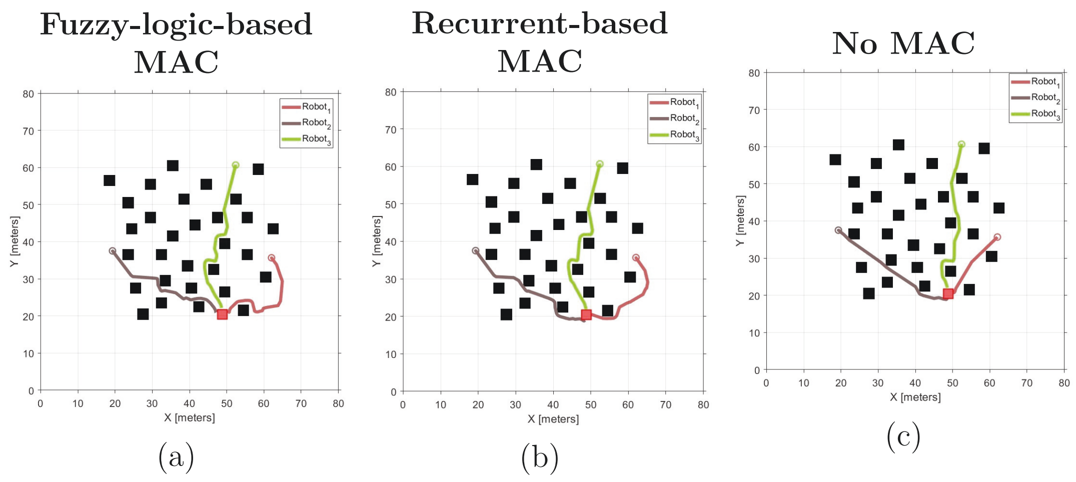
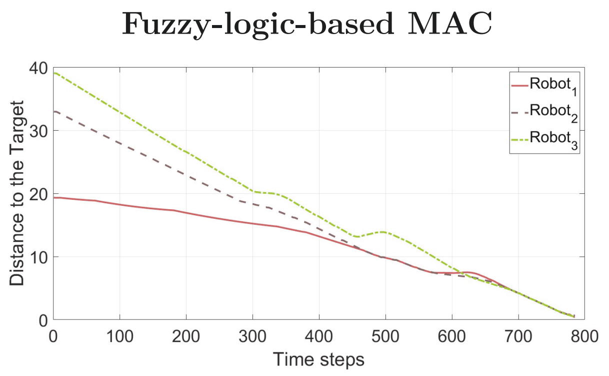


Figure 2.12: The trajectories of three robot's navigation in complex environment setting. (a) Robots are controlled by the fuzzy-logic-based MAC. (b) Robots are controlled by the recurrent-based MAC. (c) Robots are controlled by only their own navigation controllers.

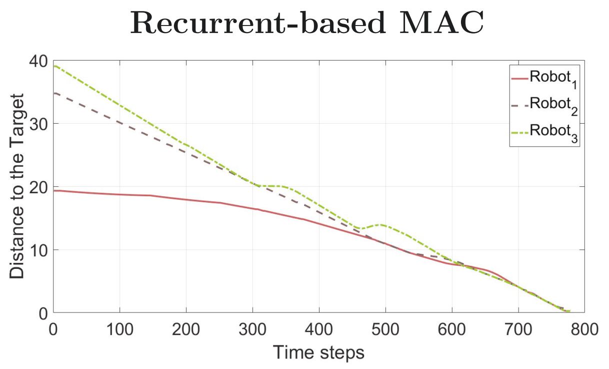
the six-robot setting is shown in Table 2.3. The fuzzy-logic-based MAC achieves a 68-time-step time difference between the fastest robot and the slowest robot, while it takes 965 time steps for the slowest robot to complete the navigation task. The recurrent-based MAC has a better result, with a 34-time-step difference between the first and last arriving robots, while it takes 907 time steps for the slowest robot to complete the navigation task. The robots controlled by the recurrent-based MAC have faster convergence speed as well as a smaller time difference between the fastest robot and the slowest robot. In this simulation, the recurrent-based MAC has a better arrival-time control performance than does the fuzzy-logic-based MAC. The control results also demonstrate that the proposed models have scalability and are able to control the navigation of different numbers of robots without retraining the model.

2.5 Summary

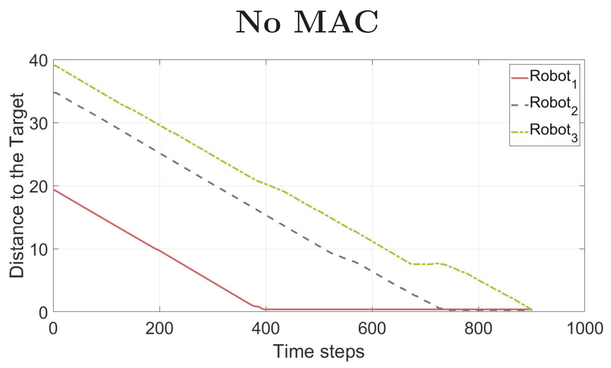
We developed a fuzzy-logic-based coordinator and recurrent-based coordinator for safely navigating multiple robots in cluttered environments, where the controller regulates their speeds and adjusts their search directions to enable simultaneous arrival of the targets. The environment for the test was an imbalanced setting,



(a)

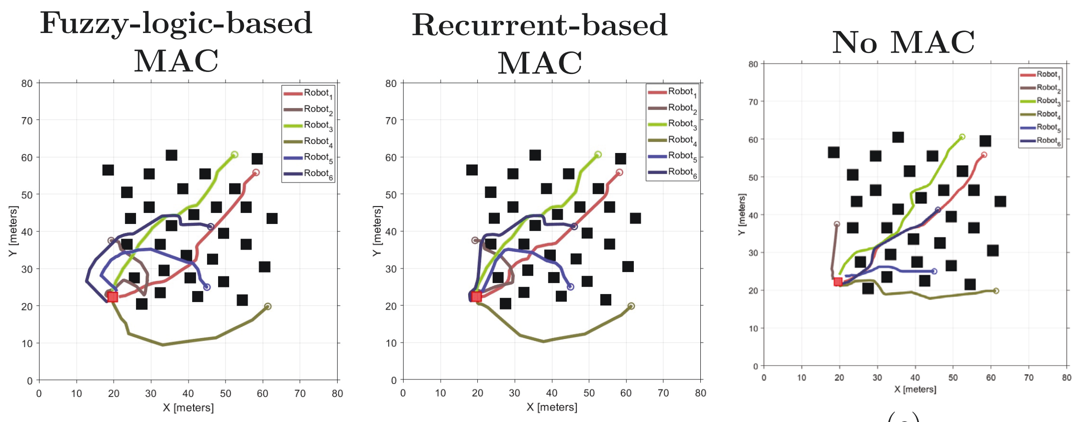


(b)



(c)

Figure 2.13: The remaining distance between the target and the three robots during the three-robot navigation and arrival time control simulation. (a) Robots are controlled by the fuzzy-logic-based MAC. (b) Robots are controlled by the recurrent-based MAC. (c) Robots are controlled by only their own navigation controllers.



(a) (b) (c)

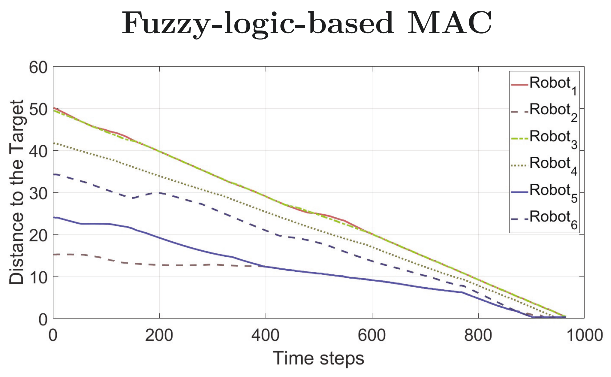
Figure 2.14: The trajectories of six robot's navigation in complex environment setting. (a) Robots are controlled by the fuzzy-logic-based MAC. (b) Robots are controlled by the recurrent-based MAC. (c) Robots are controlled by only their own navigation controllers.

Table 2.2: Performance of time arrival coordination with the three-robot setting.

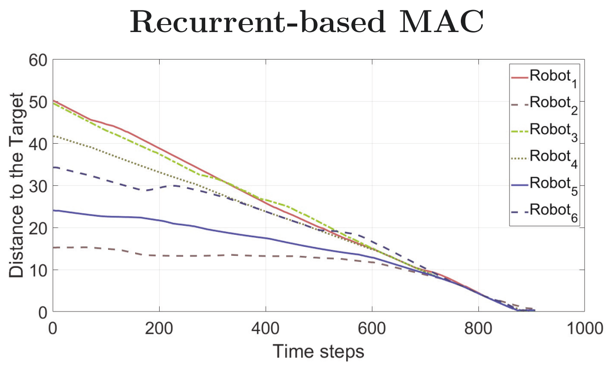
Unit: Time step	**Time difference**	**Time to complete task**
Fuzzy-logic-based MAC	5	785
Recurrent-based MAC	8	778
No MAC	504	901

Table 2.3: Performance of time arrival coordination with the six-robot setting.

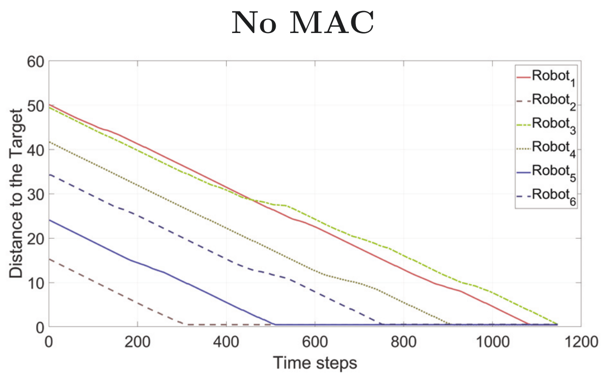
Unit: Time step	**Time difference**	**Time to complete task**
Fuzzy-logic-based MAC	58	965
Recurrent-based MAC	34	907
No MAC	839	901



(a)



(b)



(c)

Figure 2.15: The remaining distances between the target and the six robots during the six-robot navigation and arrival-time control simulation. (A) Robots are controlled by the fuzzy-logic-based MAC. (B) Robots are controlled by the recurrent-based MAC. (C) Robots are controlled by only their own navigation controllers.

in which each robot starts at positions with totally different distances to the target. The simulation results demonstrate that the two proposed models successfully enable different numbers of robots to safely navigate the environment and reach the target on time. According to the simulation results shown in Table 2.2 and Table 2.3, the recurrent-based controller outperforms the fuzzy-logic-based controller in both three-robot and six-robot settings. The LSTM structure includes cells to store the information over time, which benefits dealing with coordination problems, such as arrival time coordination in this study. However, the LSTMs has a lower noise tolerance and lower interpretability, comparing with fuzzy-logic controllers. Fuzzy logic controllers inherently have a better noise tolerance because of their Gaussian-based fuzzification operation. Furthermore, the fuzzy if-then rules can represent an inference logic for different input observations associated with output actions, which enables human users to interpret the knowledge learned by the fuzzy controller.

Chapter 3

Fuzzy-Set Interpretable Neural Fuzzy System

3.1 Backbround

This chapter presents a solution to deal with the interpretation of actions and states when designing a MAT system. The interpretability of a MAT system enables understandability and predictability that makes agents' behaviours trackable. Fuzzy systems inherently have interpretability since they can convert the knowledge into fuzzy rules that offers linguistic terms which can be understood by the human users. Many studies algorithms [12]–[14] proposed fuzzy systems considering the transparency of fuzzy sets that aims to optimise the number of fuzzy rules and fixing the partition granularity of input space during the training. Others [15]–[19] use multi-objective optimisation algorithms to assign a proper distribution of fuzzy sets in each input variable by setting constrain conditions when tuning the free parameters.

For this work, we design an interpretable FLC by using a new Multi-objective hybrid GA and PSO (MGAPSO) that incorporates a grouping and merging mechanism to improve the transparency of fuzzy sets during the optimisation process. The grouping and merging mechanism is based on a distance metric to measure the similarity between the center and variance of each fuzzy set. Any two fuzzy sets with the similarity under a given threshold will be merge into a single one. The designed FLC will be used to coordinate mobile agents in a collision-free multi-agent navigation task to explain the knowledge learned from the simulations.

Additionally, Autonomous robot navigation can be decomposed into two main behaviours: target seeking and obstacle avoidance. In [32], [48], a simple switching strategy was employed in which the target seeking mode is changed to obstacle avoidance when the distance between the robot and an obstacle is less than a given tolerance. This hard switching strategy relies on manually setting proper tolerances for various environmental conditions. In our previous work [49], we similarly used hand-coded logic for behaviour selection. In this work, we attempt to replace this mechanism with an FLC that can learn to switch between the two navigation modes. In [50], a switching mechanism based on a stochastic deep neural network was proposed for selecting among different behaviours for robots. That method performs well on a simple map, but it is not robust in the presence of large concave obstacles.

In this work, we consider scenarios in which agents cannot rely on a map because it may be outdated or change mid-mission. To coordinate a team of robots in such an environment, we construct a multi-layered system of FLCs to achieve multi-robot navigation and simultaneous arrival-time control. The control system consists of two levels of controllers: a high-level controller to ensure the same time of arrival for all robots at their destination point and a low-level controller to enable each robot to perform collision-free navigation.

The main contributions of this research are four-fold:

- This research proposes a scalable FLC-based method to enable multi-robot navigation in an unknown and complex environment. This method is based on a feature-splitting strategy, which can significantly reduce the number of parameters to be learned in the fuzzy rules without compromising navigation performance.

- In contrast to traditional switching mechanisms[32], [48], [50] between target seeking and obstacle avoidance, this research proposes a novel and robust rule-embedded FLC to improve navigation performance.

- This research develops a grouping and merging mechanism to obtain more transparent fuzzy sets and integrates this mechanism into the training process for all developed FLCs, thus increasing the interpretability of the FLCs.

- A new Multi-objective hybrid GA and PSO (MGAPSO) is presented to train all of the proposed FLCs. The MGAPSO method outperforms its single-objective counterpart in terms of both coordination accuracy and navigation performance. Specifically, the proposed method is based on performance and similarity grouping such that each swarm has its own adaptive neighbourhood.

3.2 Fuzzy Controllers for Multi-robot navigation and coordination

We consider the following scenario. A team of agents is to navigate through a cluttered (possibly dynamic) area to arrive at a designated destination point, which might also change during the mission. Each agent is equipped with a 2D lidar sensor that has an 80 m range and a 180° scanning angle. The agents cannot stop while in transit, and there is a limit on their angular velocity. Such a scenario requires a set of reactive controllers that take readings from the on-board sensors of the agents and are stable with respect to input noise. FLC is a natural choice in these circumstances. To alleviate the dimensional issues of the problem, we create a two-layered system (Fig. 3.1) for multi-robot navigation and coordination. The lower-level controller (FLC1) is responsible for the safe navigation of an individual robot, and the higher-level controller (FLC2) provides coordination for simultaneous arrival by adjusting the speeds and headings of all robots. As shown in Fig. 3.2, FLC1 consists of an obstacle avoidance controller and a behaviour selector.

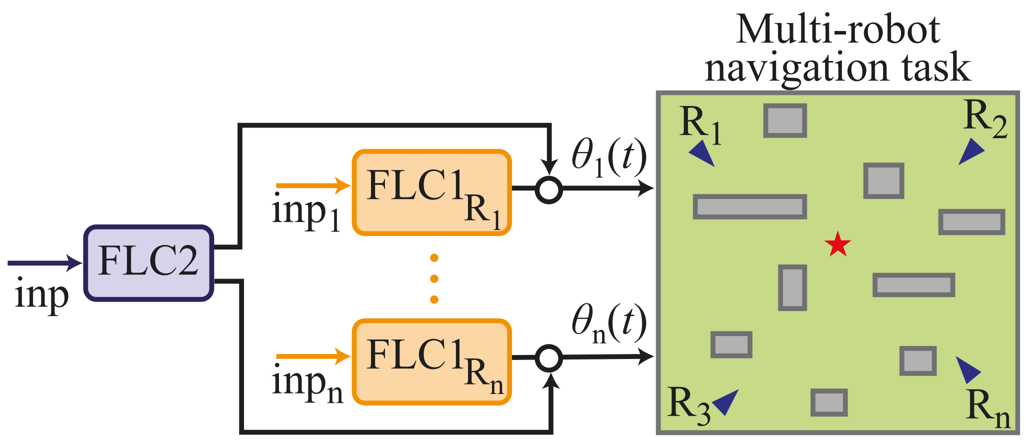


Figure 3.1: Block diagram of the control configuration for multi-robot navigation and arrival-time control.

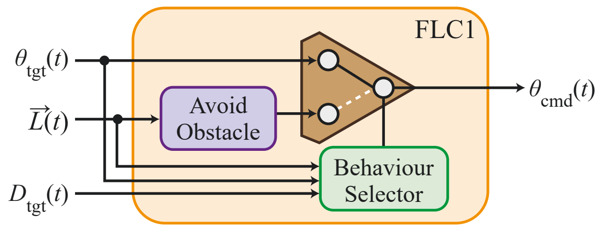


Figure 3.2: The low-level individual navigation controller.

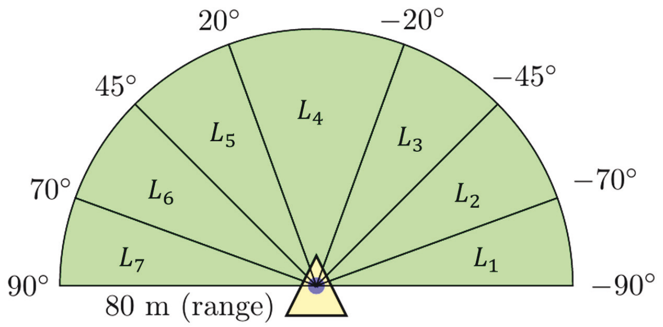


Figure 3.3: Scanning area of the 2D Lidar in the simulation setting.

3.2.1 Scalable FLC for Obstacle Avoidance

The obstacle avoidance controller is constructed as a first-order TSK-type FLC with 15 fuzzy rules of the form:

$$\begin{aligned} \textit{Rule}\ \mathrm{j}:\ & \text{IF } x_{1j} \text{ is } A_{1j} \text{ AND } x_{2j} \text{ is } A_{2j} \text{ AND } x_{3j} \text{ is } A_{3j}, \\ & \text{THEN } \theta_j = b_{0j} + \textstyle\sum_{i=1}^{3} b_{ij} x_{ij}, \end{aligned}$$

where A_{1j}, A_{2j}, A_{3j} are fuzzy sets defined by Gaussian membership functions, $j = 1, \ldots, 15$; the AND operation is implemented by the algebraic product. Inputs are provided by the Lidar sensor. For this the scanned area is divided into seven sectors (Figure 3.3). Each sector returns the shortest distance to a detected obstacle $L_1, \ldots, L_7$. To improve the scalability of the controller, we arrange scanner readings in three groups $\vec{x}_R = (L_1, L_2, L_3)$, $\vec{x}_F = (L_3, L_4, L_5)$, and $\vec{x}_L = (L_7, L_6, L_5)$ and map them into rules' inputs x_{1j}, x_{2j}, x_{3j} in the following way:

input	$j = 1, \dots, 5$	$j = 6, \dots, 10$	$j = 11, \dots, 15$
x_{1j}	L_1	L_3	L_7
x_{2j}	L_2	L_4	L_6
x_{3j}	L_3	L_5	L_5

For the last 5 rules, we make use of the left-right symmetry of the lidar sensor. The inputs from the sectors located on the sensor's left-hand side are taken in the reversed order to match the corresponding sectors from the right-hand side. This allows us to re-use membership functions of the first 5 fuzzy rules for their antecedent parts, and flip the signs of parameters defining their consequent parts and thus reduce the number of tunable parameters for this controller from 150 to 100.

3.2.2 FLC for Robot Behaviour Selection

The robot behaviour selector consists of 6 fuzzy rules that provide a switch from the default mode of going to target (Search) to obstacle avoidance (Avoid). The rules have the following structure:

$$\begin{aligned}
&\mathit{Rule}1: \textsc{if}\ \theta_{\text{tgt}} \text{ is } \theta_F \ \textsc{and}\ C_1 \text{ is True},\ \textsc{then}\ \text{Avoid};\\
&\mathit{Rule}2: \textsc{if}\ \theta_{\text{tgt}} \text{ is } \theta_R \ \textsc{and}\ C_2 \text{ is True},\ \textsc{then}\ \text{Avoid};\\
&\mathit{Rule}3: \textsc{if}\ \theta_{\text{tgt}} \text{ is } \theta_L \ \textsc{and}\ C_3 \text{ is True},\ \textsc{then}\ \text{Avoid};\\
&\mathit{Rule}4: \textsc{if}\ \theta_{\text{tgt}} \text{ is } \theta_F \ \textsc{and}\ C_4 \text{ is False},\ \textsc{then}\ \text{Search};\\
&\mathit{Rule}5: \textsc{if}\ \theta_{\text{tgt}} \text{ is } \theta_R \ \textsc{and}\ C_5 \text{ is False},\ \textsc{then}\ \text{Search}.\\
&\mathit{Rule}6: \textsc{if}\ \theta_{\text{tgt}} \text{ is } \theta_L \ \textsc{and}\ C_6 \text{ is False},\ \textsc{then}\ \text{Search};
\end{aligned}$$

Here θ_{tgt} is the scaled angle to the target; θ_F, θ_L and θ_R are fuzzy sets of direction corresponding to the scenarios that the target is at the front, at the left or right side of the drone, respectively.

$C_{[1,6]}$ are defined as:

$$\begin{aligned}
C_1 :=& mh1\_R1 \ \textsc{or}\ mh2\_R2 \ \textsc{or}\ mh3\_R1\\
C_2 :=& mh1\_R2 \ \textsc{or}\ mh2\_R2 \ \textsc{or}\ mh3\_R2\\
C_3 :=& mh1\_R3 \ \textsc{or}\ mh2\_R3 \ \textsc{or}\ mh3\_R3\\
C_4 :=& \text{Inverse of } C_1\\
C_5 :=& \text{Inverse of } C_2\\
C_6 :=& \text{Inverse of } C_3
\end{aligned} \tag{3.1}$$

where

$$\begin{aligned}
mh1\_R1 =& (L_3 \text{ is } L_\text{F} \text{ AND } d_{L_3} \text{ is } Z_d),\\
mh2\_R1 =& (L_4 \text{ is } L_\text{F} \text{ AND } d_{L_4} \text{ is } Z_d),\\
mh3\_R1 =& (L_5 \text{ is } L_\text{F} \text{ AND } d_{L_5} \text{ is } Z_d),\\
mh1\_R2 =& (L_1 \text{ is } L_\text{R} \text{ AND } d_{L_1} \text{ is } Z_d),\\
mh2\_R2 =& (L_2 \text{ is } L_\text{R} \text{ AND } d_{L_2} \text{ is } Z_d),\\
mh3\_R2 =& (L_3 \text{ is } L_\text{R} \text{ AND } d_{L_3} \text{ is } Z_d),\\
mh1\_R3 =& (L_5 \text{ is } L_\text{L} \text{ AND } d_{L_5} \text{ is } Z_d),\\
mh2\_R3 =& (L_6 \text{ is } L_\text{L} \text{ AND } d_{L_6} \text{ is } Z_d),\\
mh3\_R3 =& (L_7 \text{ is } L_\text{L} \text{ AND } d_{L_7} \text{ is } Z_d).
\end{aligned} \tag{3.2}$$

Here L_F, L_L and L_R are fuzzy sets of distance corresponding to the scenarios that the target is at the front, at the left or right side of the drone, respectively; Z_d is pre-defined by user and determines the status of reaching the target, i.e. True or False; $d_{L_i} = L_i - d_\text{tgt}$, where d_tgt is the distance to the target. The above fuzzy rules can be interpreted as: If there is an obstacle between the target and the robot, one of the rules (*Rule*1, *Rule*2 and *Rule*3) is triggered, then the robot changes the behaviour to Avoid. If the robot has a direct line to the target, one of the rules (*Rule*4, *Rule*5 and *Rule*6) is triggered, then the robot changes its behaviour to Search.

3.2.3 FLC for Time-arrival Coordination

FLC2 is a centralised controller that synchronises the robots' arrival time at the target. During each control loop, agents are ranked in the ascending order based on their (estimated) distance to the target, so that robot R_{rank_1} is the closest to the destination point and robot R_{rank_n} is the furthest from it. To accommodate teams of variable sizes, FLC2 directly controls only robots at both ends of the ranking. FLC2 takes five inputs: v_1, v_n (the moving speeds of robots R_{rank_1} and R_{rank_n}), their distances to the target D_1, D_n, and $\Delta D = D_1 - D_n$. The controller is constructed as a 0-order TSK fuzzy system with 5 inputs, 2 outputs and 10 fuzzy rules of the form:

$$\textit{Rule } j: \text{ IF } D_1 \text{ is } B_{1j} \text{ AND } \dots \text{ AND } \Delta D \text{ is } B_{5j}, \text{ THEN } \vec{y} \text{ is } \vec{\alpha}_j.$$

Here $\vec{\alpha}_j$ are 2-by-1 vectors, and $B_{1j}, \dots, B_{5j}$ are Gaussian membership functions. There are 120 tunable parameters in FLC2.

The outputs of the controller are speed scale factors α_1 and α_n for the first and last ranked robots, respectively. The scale factors for the rest of the robots are calculated as

$$\alpha_i = \frac{i-1}{n-1}(\alpha_n - \alpha_1) + \alpha_1,\ i = 2, \dots, n-1, \tag{3.3}$$

where $\alpha_i \in [0.5, 1.5]$. Velocities' updates are calculated as $\hat{v}_i = \alpha_i v_i$. The velocities are allowed to change within $\pm 50\%$ of the user-defined value.

To make the length of robots' paths roughly equal, FLC2 also adjusts each robot's heading angle. The heading adjustment for robot i at time step t is calculated as:

$$\theta^i_{\text{adj}}(t) = \theta^i_{\text{tgt}}(t) + \beta_i \theta_{\text{max\_adj}}, \tag{3.4}$$

where $\theta^i_{\text{tgt}}(t)$ is the target bearing (in body coordinates), and $\theta_{\text{max\_adj}}$ is the maximum allowed angle change, β_i is a scale factor that determines the strength of heading angle adjustment and is calculated from D_n, D_i (estimated distances to the target for robots n and i calculated at time step $(t-1)$) and α_i as

$$\beta_i = \sqrt{(1 - D_i/D_n)/\alpha_i} \tag{3.5}$$

.

3.3 Multi-objective hybrid GA and PSO algorithm

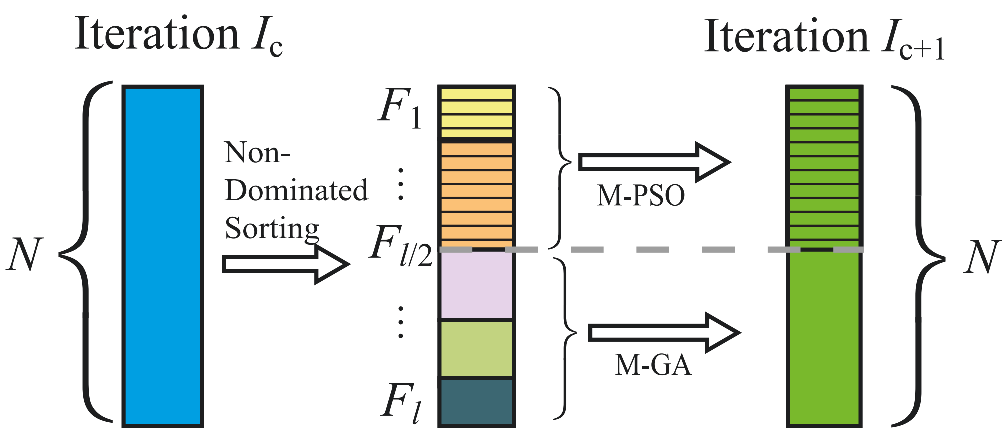


Figure 3.4: Learning configuration in the MGAPSO algorithm

In the MGAPSO, each particle or individual represents a whole solution for a specific problem. A block diagram of the procedure for creation of new individuals is

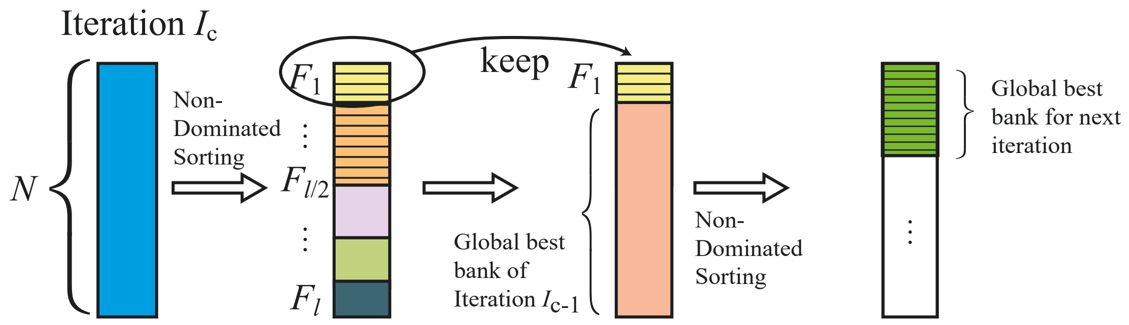


Figure 3.5: The process of updating the local-best bank.

shown in Figure 3.4. At each iteration, a non-dominated sorting categorises N new solutions into l fronts based on their performances. A front is created of individuals that cannot be dominated by each other due to incomparable fitness values. The top-half ranked individuals are updated by multi-objective PSO (M-PSO) algorithm. The rest of the solutions are updated using multi-objective GA (M-GA). At every iteration after new solutions are generated, the MGAPSO updates the local-best bank (Figure 3.5). Suppose there are N new solutions categorised into l fronts. The first-ranked solutions are kept and then mixed with those in the old local-best bank. The non-dominated sorting is applied to the mixed set of local-best, and only first-ranked individuals form a new local-best bank for the next iteration.

3.3.1 M-GA

The crossover operation of the M-GA is performed, when the local-best bank has more than two elements. A new particle vector $\vec{s}_{\text{new}} = (s^1_{\text{new}}, \ldots, s^M_{\text{new}})$ is created as follows:

$$s^j_{\text{new}} = \begin{cases} r_c P^j_1 + (1 - r_c) P^j_2, & j = 1, \ldots, \tau \\ (1 - r_c) P^j_1 + r_c P^j_2, & j = \tau + 1, \ldots, M, \end{cases} \tag{3.6}$$

where r_c is a random number from $[0, 1]$, and $\vec{P}_1$ and $\vec{P}_2$ are two parents randomly selected from the local-best bank, τ denotes a randomly selected crossover site, M is the length of the solution vector. After crossover operation, mutations occur with a predefined probability p_m. The mutation operation is implemented as

$$s^j_{\text{new}} \leftarrow s^j_{\text{new}} + c_m r^j_m (s^j_{\max} - s^j_{\min}), \quad j = 1, \ldots, M, \tag{3.7}$$

where r^j_m is a random number on $[-1, 1]$, c_m is a coefficient to restrict strength of mutation; and $\vec{s}_{\max}$ and $\vec{s}_{\min}$ are upper and lower bounds of the solution space, respectively. The initial velocity vector of each new solution is assigned by the following formula:

$$\vec{v}_{\text{new}} = c_m(\vec{r}_1(\vec{P}_1 - \vec{s}_{\text{new}}) + \vec{r}_2(\vec{P}_2 - \vec{s}_{\text{new}})), \tag{3.8}$$

here $\vec{r}_1$ and $\vec{r}_2$ are vectors of random numbers on $[0, 1]$.

If there is only one global solution in the bank, M-GA performs asexual reproduction operation to generate new solutions. It is implemented by Gaussian sampling. Suppose there is only one local-best solution P^g. Each element of P^g is set as a centre of the Gaussian density function, and the standard deviation is set to 0.1. The asexual reproduction operation is followed by a mutation to prevent new solutions from becoming too similar with each other.

3.3.2 M-PSO

In addition to M-GA, the MGAPSO incorporates velocity and position update mechanism of the M-PSO to ensure that all solutions explore in a relatively optimal area during the optimisation process. The swarm is divided into N_g groups in the same manner as in [32]. Neighbours are particles within the same group. The neighbourhood best position, $\vec{P}^g_z$, is the leading particle in group z selected as

$$\vec{P}^g_z = \underset{1 \le k \le n}{\arg\max}\, \text{SPD}(\vec{s}_i, \vec{P}^g_k). \tag{3.9}$$

Here n is the size of the global-best bank. SPD is a metric based on similarity and fitness of particles defined as

$$\text{SPD}(\vec{s}_i, \vec{P}^g_k) = \frac{\|\vec{s}_i - \vec{P}^g_k\| D_p(\vec{P}^g_k)}{\|\vec{f}(\vec{s}_i)\| - \|\vec{f}(\vec{P}^g_k)\|}, \tag{3.10}$$

where $\vec{s}_i$ is the i-th particle in the group, $\vec{P}^g_k$ is k-th local best, $\vec{f} = [f^1_i, \ldots, f^{N_{\text{obj}}}_i]$ is a N_{obj}-dimensional objective vector. To discriminate solutions in a front, we introduce a crowding distance $D_p(\cdot)$. Solutions are sorted by their crowding distance values in ascending order. The crowding distance of i-th solution $D_p(\vec{s}_i)$ is obtained as follows:

$$D_p(\vec{s}_i) = \frac{1}{N_{\text{obj}}} \sum_{j=1}^{N_{\text{obj}}} u^j_i, \tag{3.11}$$

with

$$u^j_i = \frac{|f^j_{i+1} - f^j_{i-1}|}{f^j_{\max} - f^j_{\min}}, \quad j = 1, \ldots, N_{\text{obj}}, \tag{3.12}$$

where $f_{\max}^j$ and $f_{\min}^j$ are, respectively, the maximum and minimum values of the j-th objective, i is the running index for solutions in a group. Crowding distances of two boundary solutions are set to 1.

The velocity $\vec{v}_i$ of each particle i in group z is updated using its individual best position, $\vec{P}_i$, and the neighbourhood best $\vec{P}_z^g$:

$$\begin{aligned} &\vec{v}_i(t+1) = \\ &\quad \chi((\vec{v}_i(t) + c_1\vec{r}_1(\vec{P}_i - \vec{s}_i(t)) + c_2\vec{r}_2(\vec{P}_z^g - \vec{s}_i(t))), \end{aligned} \tag{3.13}$$

where χ is a constriction factor that controls the magnitude of $\vec{v}_i$, c_1 and c_2 are positive acceleration coefficients. This method forces groups to locate different optima and thus reduces the chance of all solutions being trapped in a local minimum. Each particle changes its position according to the following formula:

$$\vec{s}_i(t+1) = \vec{s}_i(t) + \vec{v}_i(t+1). \tag{3.14}$$

3.3.3 Interpretability Improvement

This study considers the transparency of fuzzy sets to improve the interpretability of an FLC. Namely, transparent fuzzy sets should 1) be distinguishable enough; 2) represent the universe of discourse of input variables. Additionally, the antecedence of the fuzzy model should be as simple as possible. Generally, a smaller number of fuzzy sets are more desirable. In this research, we propose a grouping and merging mechanism to obtain transparent fuzzy sets and integrate the method into the training process of an FLC.

The grouping mechanism is to find similar fuzzy sets in a partition of input variable by using distance measurement and a pre-defined distance threshold δ_g. Given input x_i and assume B_{ij} is a fuzzy set of rule j, the Gaussian membership function can be described as the following form,

$$\mu_{ij}(x_i) = exp\left\{-\frac{(x_i - m_{ij})^2}{{\sigma_{ij}}^2}\right\}. \tag{3.15}$$

where μ_{ij} is the membership value of B_{ij} with input x_i. Given two fuzzy sets in input variable i, B_{ip} and B_{iq}, the distance between the two fuzzy sets is defined as,

$$\rho_m\left(B_{ip}, B_{iq}\right) = \sqrt{(m_{ip} - m_{iq})^2 + (\sigma_{ip} - \sigma_{iq})^2}. \tag{3.16}$$

A cut-off distance ρ_{cutoff} is set for grouping fuzzy sets in an input variable. The fuzzy set grouping algorithm is shown in Algorithm 2. To group fuzzy sets in input variable

i, the fuzzy sets are sorted on their centres in ascending order as $\widetilde{B}_{i1}, \ldots, \widetilde{B}_{ir}$. The grouping algorithm sequentially evaluate the distance ρ_m starting from $\widetilde{B}_{i1}$, those fuzzy sets that located within ρ_{cutoff} will be grouped in the first group G_1. If a fuzzy set B_{ij} of which $\rho_m\left(\widetilde{B}_{i1}, \widetilde{B}_{ij}\right) > \rho_{cutoff}$, then B_{ij} will be designated in the new group G_2. The algorithm continuously evaluate ρ_m between each pair of fuzzy sets until every fuzzy sets are designated to their own groups. After grouping fuzzy sets, a reference centre will be generated for each group. Every reference centre is obtained by calculating the arithmetic mean from the centres of fuzzy sets assigned in the same group. The reference centres then are used to guide those fuzzy sets locating in the same group; the fuzzy sets of input variable i are updated as follows,

$$\hat{m}_{ip} = m_{ip} + r_3(m_{i,G_j} - m_{ip}),\ \widetilde{B}_{ip} \in G_j, \tag{3.17}$$

where m_{ip} is the centre of fuzzy set $\widetilde{B}_{ip}, \hat{m}_{ip}$ is the updated centre and $r_3 \in [0, 1]$ is a uniform random number. The update of centres occurs after the position update of the M-PSO; the fuzzy sets in the same group would move closer to each other during the learning. Significant overlapping between neighbouring fuzzy sets may occur due to the change in their centres and widths. Denote the degree of overlapping as δ, and given sorted fuzzy sets $\left\{\widetilde{B}_{ip},\ \widetilde{B}_{iq}, \ldots\right\} \in G_j$, then degree of overlapping between $\widetilde{B}_{ip}$ and $\widetilde{B}_{iq}$ can be evaluated as follows,

$$\delta_{pq} = max\left(\mu_{ip}(m_{iq}),\ \mu_{iq}(m_{ip})\right), \tag{3.18}$$

where $\mu_{ip}(m_{iq})$ is the membership value of $\widetilde{B}_{ip}$ fed with m_{iq}, the centre of $\widetilde{B}_{iq}$, and $\mu_{iq}(m_{ip})$ is the membership value of $\widetilde{B}_{iq}$ fed with m_{ip}. The higher δ_{pq} indicates greater overlap between $\widetilde{B}_{ip}$ and $\widetilde{B}_{iq}$. If δ_{pq} is larger than the pre-set threshold value δ_{th}, the two neighbouring fuzzy sets will be merge together. The centre the new fuzzy set is set to the average centre of the two merged fuzzy sets, and the width is set to the average width. The parameters of two merged fuzzy sets in the solution vector will be replaced by the parameters of the new fuzzy set. Figure 3.6 shows the pipeline of the intrpretalbe fuzzy controller learning based on the MGAPSO.

3.4 Controller Training

All three controllers introduced in Section 3.2 are trained separately in a cascading manner. First, the obstacle avoidance part of FLC1 is trained to control a robot to follow the boundary of an obstacle. When this training is complete, the behaviour selector part of FLC1 is trained to navigate a robot in a cluttered environment

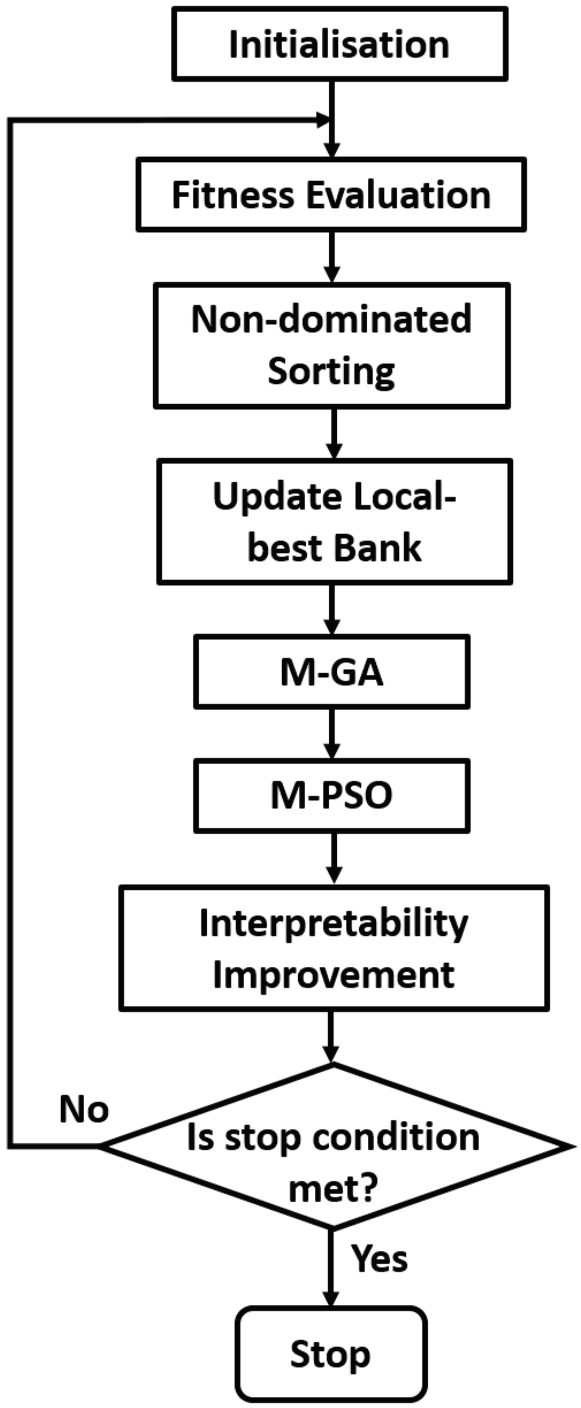


Figure 3.6: Flowchart of the MGAPSO algorithm

Algorithm 2 Fuzzy Set Grouping Algorithm

```
Require: $\widetilde{B}_{i1}, \ldots, \widetilde{B}_{ir}$, fuzzy sets sorted in ascending order.
  Initialization: Set empty groups $G_j \leftarrow \emptyset (1 \le j \le r)$
  $j \leftarrow 1$, set the index of group j to 1
  $G_j \leftarrow \widetilde{B}_{i1}$, Assign $\widetilde{B}_{i1}$ to the first group.
  for $p = 1, \ldots, r$ do
    for $q = p + 1, \ldots, r$ do
      if $\rho_m \left(\widetilde{B}_{ip}, \widetilde{B}_{iq}\right) < \rho_{cutoff}$ then
        $G_j \leftarrow G_j \cup \widetilde{B}_{iq}$ , assign $B_{iq}$ to $G_k$
      else
        $p \leftarrow q$
        $j \leftarrow j + 1$
      end if
    end for
  end for
```

towards its destination point. The obstacle avoidance controller is fixed during this phase of the training. And finally, FLC2 is trained to coordinate a group of robots (each carrying an already trained FLC1) to achieve a simultaneous arrival. The maps used in training are shown in Figure 3.7.

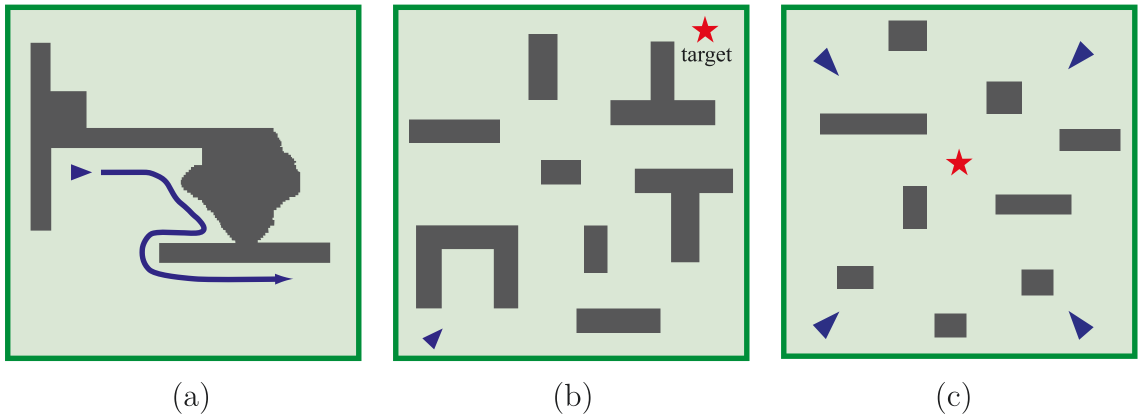


Figure 3.7: Maps for 3 phases of controller training.

3.4.1 Phase 1 – training for obstacle avoidance

During phase 1 of training, the robot moves around the obstacle shown in Figure 3.7 (a) and stops when the following constraint is violated

$$\min(L_1, L_2, L_3, L_4, L_5, L_6, L_7) > D_{\text{min}}. \tag{3.19}$$

Here D_{min} is a user set value. When the robot stops, the number of time steps taken is recorded as T_{total}. The distance from the obstacle which the robot maintains every time step is used as the other criterion for performance evaluation. Thus we have two objectives f_1 and f_2:

$$f_1 = \frac{\sum_{t=1}^{T_{\text{total}}} |L_7(t) - D_{\text{BF}}|}{T_{\text{total}}}, \quad f_2 = 1/T_{\text{total}}. \tag{3.20}$$

D_{BF} is a user set value.

3.4.2 Phase 2 – collision-free navigation

A robot equipped with a boundary-following knowledge is now placed in a more complex environment (Figure 3.7 (b)). The goal is to safely navigate from the starting point to the target location relying only on sensor data. The boundary-following behaviour is fixed. The second part of FLC1 works as a switch between two modes: going to the target and avoiding an obstacle. The fitness value of a solution is determined by running a simulation. The simulation stops either when a pre-set number of steps is reached, or when the distance to the target location becomes less than a user-defined value D_{miss}. Robot's trajectory is given a rating based on its final distance from the target D_{final}, number of collisions with obstacles N_{c}, and the total length of the trajectory L. This gives us two objective functions:

$$f_3 = N_{\text{c}} + P, \quad f_4 = L + P, \tag{3.21}$$

where P is the penalty function defined as

$$P = \begin{cases} 0, & \text{if } D_{\text{final}} \le D_{\text{miss}}, \\ 1000 D_{\text{final}}, & \text{if } D_{\text{final}} > D_{\text{miss}}. \end{cases} \tag{3.22}$$

The penalty value is added to both objective functions in (3.21) to gradually eliminate those particles that fail to reach the target.

3.4.3 Phase 3 – training for time-arrival coordination

For the time-arrival coordination, a team of 4 agents is placed in a single scenario as shown in Figure 3.7 (c). Each robot is equipped with an FLC1 and a 2D-Lidar sensor. A single destination point is selected. The scenario set-up is fixed during the training process. The evaluation of a controller performance involves a complete simulation run from start to end until either all robots arrive at their targets or the simulation time limit is reached. At the end of the run, robots are ranked again by their arrival times, and the first (T_1) and the last (T_n) of these times are used to generate two objective values as follows:

$$f_5 = |T_n - T_1|, \quad f_6 = (T_1 + T_n)/2. \tag{3.23}$$

When f_5 is at minimum, the robots arrive at the destination point at the same time; f_6 keeps all robots moving at maximum speeds by minimising the average arrival time of the two robots positioned at the opposite ends of the ranking.

3.4.4 Comparison between Multi-Objective and Single-objective

To compare the optimisation performance of the MGAPSO with that of the single-objective PSO, we conducted 10 runs for each training phase using both algorithms for statistical evaluation. When using PSO, we combined objective functions defined in (3.20)–(3.23) using cascading weightings:

$$\begin{aligned} f_{\mathrm{PSO}_1} &= 0.1 f_1 + 10 f_2, \\ f_{\mathrm{PSO}_2} &= 10 f_3 + 0.1 f_4, \\ f_{\mathrm{PSO}_3} &= 10 f_5 + 0.1 f_6. \end{aligned} \tag{3.24}$$

We also selected from each run of the MGAPSO those fitness pairs that have a smaller combined value for comparison. In the first objective function $f_{\mathrm{PSO}_1} = 0.1 f_1 + 10 f_2$, f_1 enables the robot to keep an adequate distance to avoid an obstacle, and f_2 is to evaluate the performance of the boundary-following movement. We considered f_2 more important than f_1 because we expected that the single-objective PSO could find solutions at least enabling robot's collision-free movement during the task. The second f_{PSO_2} consists of f_3 and f_4, which record the times of collisions occurring and the final distance to the target. We expected that the collision does not happen during the navigation task. Thus, f_3 has a higher weight than f_4. In the third objective function $f_{\mathrm{PSO}_3} = 10 f_5 + 0.1 f_6$, f_5 is to minimise the time difference of robots arriving at their destination point, and f_6 is to ensure that all robots move at their maximum speeds during their navigation. f_5 has a higher weight as it is the main purpose of designing the controller; to coordinate a group of robots to arrive at their destination point simultaneously. f_{PSO_3} also had been used in our previous study [49] and demonstrated its performance in designing the fuzzy controller for a variant number of robots. The results are gathered in Table 3.1. In our opinion, the MGAPSO succeeds in forcing the particles to explore a wider area and through this it finds a better solution.

Table 3.1: Comparing the MGAPSO and PSO for controller training.

	MGAPSO		PSO	
	Average	STD	Average	STD
f_1	4.2051	0.9760	5.4971	0.7080
f_2	0.0033	0.0	0.0036	0.0005
f_3	0.0	0.0	0.0	0.0
f_4	186.83	3.14	187.57	1.56
f_5	0.41	0.053	0.63	0.097
f_6	432.9	35.3	458.6	30.8

3.4.5 Simulations

3.4.6 FLC1 behaviour selector vs simple switching mechanism

To compare the performance of the FLC1 behaviour select or with that of the hand-coded switch logic, we ran 20 robots in a test environment. The results are shown in Figure 3.8. Plot (a) shows trajectories produced by robots equipped with both parts of FLC1: FLC1-A (the avoid-obstacle controller) and FLC1-B (an embedded behaviour selector). Plot (b) shows trajectories produced by robots equipped with FLC1-A and a hand-coded switching mechanism. We observe that FLC1-B more reliably achieves collision-free navigation compared to its hand-coded counterpart.

3.4.7 Arrival time coordination

In order to make FLC2 able to handle teams of variable sizes without retraining, we construct it so that it directly controls only two agents (the furthest and the nearest to the destination point) by producing two speed adjustment factors for

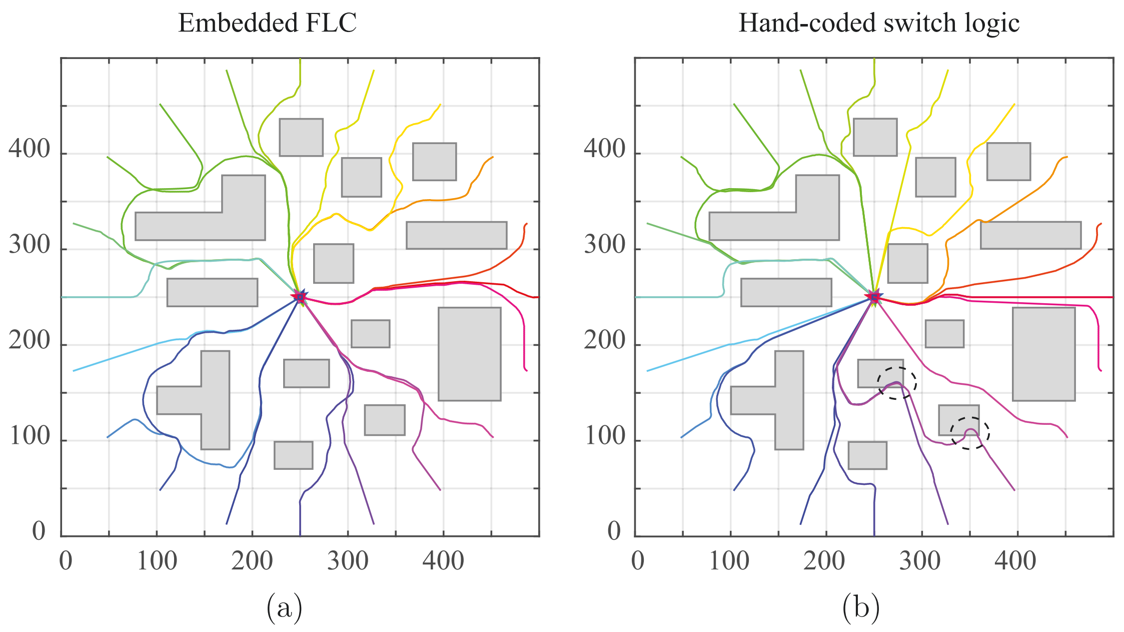


Figure 3.8: Trajectories produced by (a) the trained behaviour selector and (b) the hand-coded switching mechanism as in [32], [49]. Each line represents a robot's trajectory.

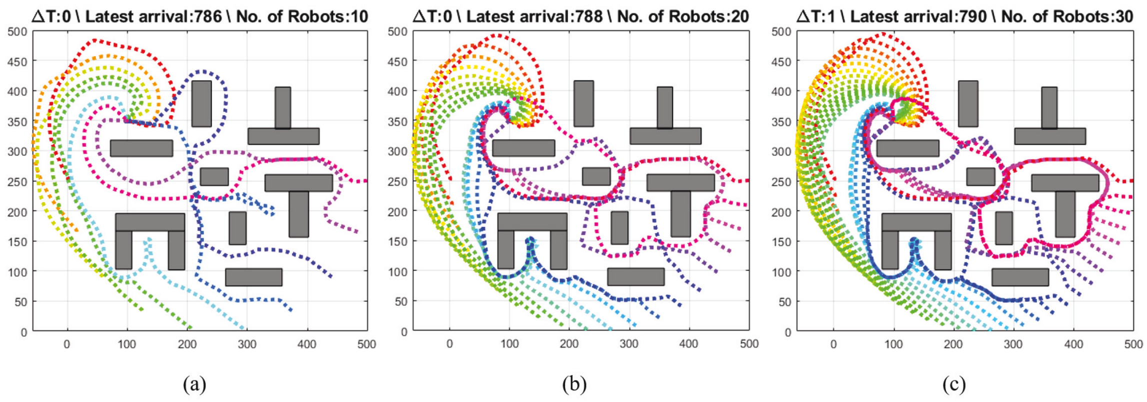
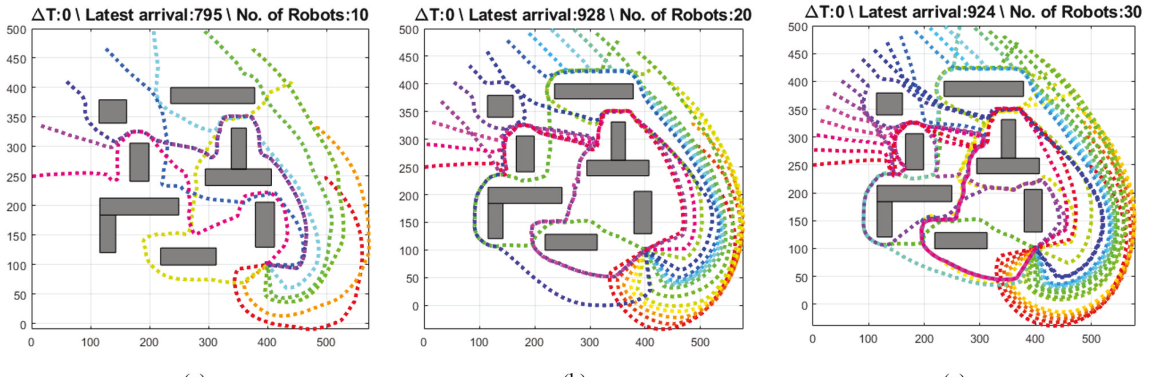


Figure 3.9: Arrival time coordination of robotic teams of various sizes. The number of robots in (a), (b) and (c) are 10, 20 and 30, respectively.



(a) (b) (c)

Figure 3.10: Arrival time coordination of robotic teams of various sizes. The number of robots in (a), (b) and (c) are 10, 20 and 30, respectively.

them. For the rest of the team, these factors are calculated using (3.3). FLC2 also controls heading angles via (3.4) and (3.5).

Figures 3.9 and 3.10 show the results of the controller managing teams of 10, 20 and 30 agents in test environments. In Figure 3.9, the target point is set at $(x, y) = (100, 350)$. The controller achieves 1-time-step time difference between the fastest robot and the slowest robot, and takes 788 time steps to complete the navigation task when it controls 30 robots. When the number of robots is reduced to 10, the controller achieves zero time difference with 786 time-steps completion time. In Figure 3.10, the target point is set at $(x, y) = (400, 100)$. FLC2 achieves perfect time control for all three teams, but the completion time increases with the size of the team. It takes a team of 10 agents 795 time-steps to complete the task, whereas a team of 30 needs 924 time-steps. The simulation results demonstrate that the proposed controller has good scalability and robustness; without any retraining it is able to coordinate varying number of robots with acceptable performance.

3.4.8 Interpretable fuzzy controller for arrival time coordination

This section presents the learned knowledge of the interpretable fuzzy controller for arrival time coordination. Table 3.2 reveals the MGAPSO-designed fuzzy rules. We set the number of fuzzy rules to 10, and there are five input variables and two consequent components in each rule. The fuzzy controller here is fed D_1, D_n, v_1, v_n, and ΔD. D_1 and D_n are the distance between the target and robot R_{rank_1} and robot R_{rank_n}, respectively. v_1 and v_n are the moving speeds of robots R_{rank_1} and R_{rank_n}. ΔD is the difference of remaining distance of robots R_{rank_1} and R_{rank_n}, that is $\Delta D = D_1 - D_n$. The outputs of the fuzzy controller are α_1 and α_n for speed regulation of robots R_{rank_1} and R_{rank_n}. Figure 3.11 shows the corresponding fuzzy sets in the five input variables. Comparing to the fuzzy sets learned without the interpretability improvement, as shown in Figure 3.12, the transparency of fuzzy sets in Figure 3.11 is improved, which allow human to understand the learned knowledge in the MGAPSO-designed fuzzy rules. For example, *Rule*1 represent the situation that R_{rank_1} is moving "Very Fast" and in "Middle Near" distance to the target and R_{rank_n} is moving "Slow" and in "Very Far". Consequently, *Rule*1 slows down the R_{rank_1} with $\alpha_1 = 0.8635$ and speeds up R_{rank_n} with $\alpha_1 = 1.9476$. If both R_{rank_1} and R_{rank_n} are locating far away from the target and moving very fast, the firing strength of *Rule*7 will become large; the controller will speed up both R_{rank_1} and R_{rank_n}. On the other side, if R_{rank_1} is approaching the target and moving fast, and R_{rank_n} is still in the middle distance and moving slowly, the firing strength of *Rule*4 will become large; the controller will reduce the moving speed of R_{rank_1} and accelerate R_{rank_n} to catch up R_{rank_1}.

This section presents the knowledge learned by the interpretable fuzzy controller for arrival-time coordination. Table 3.2 reveals the MGAPSO-designed fuzzy rules. We set the number of fuzzy rules to 10, and there are five input variables and two consequent components in each rule. The fuzzy controller here is fed D_1, D_n, v_1, v_n, and ΔD. D_1 and D_n are the distances between the target and robots R_{rank_1} and R_{rank_n}, respectively. v_1 and v_n are the movement speeds of robots R_{rank_1} and R_{rank_n}, respectively. ΔD is the difference in the remaining distances of robots R_{rank_1} and R_{rank_n} to the target, that is, $\Delta D = D_1 - D_n$. The outputs of the fuzzy controller are α_1 and α_n for the speed regulation of robots R_{rank_1} and R_{rank_n}, respectively. Fig. 3.11 shows the corresponding fuzzy sets in the five input variables. Compared to the fuzzy sets learned without the interpretability improvement, as shown in Fig. 3.12, the transparency of the fuzzy sets in Fig. 3.11 is improved, allowing humans to understand the learned knowledge captured in the MGAPSO-designed fuzzy rules. For example, *Rule* 1 represents the situation in which R_{rank_1} is moving "Very Fast" and at a "Middle Near" distance to the target and R_{rank_n} is moving at a "Slow" speed and a "Very Far" distance. Consequently, *Rule* 1 slows down R_{rank_1} with $\alpha_1 = 0.8635$ and speeds up R_{rank_n} with $\alpha_1 = 1.9476$. If both R_{rank_1} and R_{rank_n} are located far from the target and are moving very fast, then the firing strength of *Rule* 7 will become large; the controller will speed up both R_{rank_1} and R_{rank_n}. On the other hand, if R_{rank_1} is approaching the target and moving fast while R_{rank_n} is still at a middle distance and moving slowly, then the firing strength of *Rule* 4 will become large; the controller will reduce the movement speed of R_{rank_1} and accelerate R_{rank_n} to catch up with R_{rank_1}.

Table 3.2: Fuzzy rules trained by the MGAPSO with interpretability improvement

Rules	**Input 1**	**Input 2**	**Input 3**	**Input 4**	**Input 5**	α_1	α_n
Rule 1	$B_{1,2}$ (Middle Near)	$B_{2,4}$ (Very Far)	$B_{3,4}$ (Very Fast)	$B_{4,1}$ (Slow)	$B_{5,3}$ (Distant)	0.8635	1.9476
Rule 2	$B_{1,1}$ (Reaching)	$B_{2,1}$ (Very Near)	$B_{3,1}$ (Very Slow)	$B_{4,1}$ (Slow)	$B_{5,1}$ (Closing)	1.5919	1.6165
Rule 3	$B_{1,5}$ (Very Far)	$B_{2,4}$ (Very Far)	$B_{3,3}$ (Fast)	$B_{4,1}$ (Slow)	$B_{5,3}$ (Distant)	0.8785	1.8372
Rule 4	$B_{1,1}$ (Reaching)	$B_{2,2}$ (Near)	$B_{3,4}$ (Very Fast)	$B_{4,1}$ (Slow)	$B_{5,2}$ (Not Distant)	0.3147	1.7453
Rule 5	$B_{1,1}$ (Reaching)	$B_{2,1}$ (Very Near)	$B_{3,2}$ (Slow)	$B_{4,1}$ (Slow)	$B_{5,1}$ (Closing)	1.3919	1.6165
Rule 6	$B_{1,3}$ (Middle)	$B_{2,3}$ (Middle)	$B_{3,4}$ (Very Fast)	$B_{4,2}$ (Fast)	$B_{5,1}$ (Closing)	0.8568	0.7833
Rule 7	$B_{1,5}$ (Very Far)	$B_{2,4}$ (Very Far)	$B_{3,3}$ (Fast)	$B_{4,3}$ (Very Fast)	$B_{5,2}$ (Not Distant)	1.5635	1.9476
Rule 8	$B_{1,3}$ (Middle)	$B_{2,4}$ (Very Far)	$B_{3,1}$ (Very Slow)	$B_{4,1}$ (Slow)	$B_{5,3}$ (Distant)	0.5705	1.9722
Rule 9	$B_{1,2}$ (Middle Near)	$B_{2,3}$ (Middle)	$B_{3,4}$ (Very Fast)	$B_{4,1}$ (Slow)	$B_{5,4}$ (Very Distant)	0.9477	0.81038
Rule 10	$B_{1,4}$ (Far)	$B_{2,4}$ (Very Far)	$B_{3,2}$ (Slow)	$B_{4,2}$ (Fast)	$B_{5,1}$ (Closing)	1.1705	0.72542

3.5 Summary

This research investigated the problem of multi-robot navigation and simultaneous arrival coordination in an unknown environment. We presented a two-layer structure of FLCs to ensure safe navigation with coordination of the arrival times. All three proposed controllers (for obstacle avoidance, behaviour selection and arrival-time coordination) are based on fuzzy inference systems. By splitting the large-scale feature space into several smaller ones, we reduced the number of parameters to be learned without compromising the performance of the low-level controller. Additionally, in contrast to traditional behaviour switching mechanisms, we proposed a novel rule-embedded FLC to improve the navigation performance. Moreover, we developed a grouping and merging mechanism to obtain transparent fuzzy sets and integrated this mechanism into the training process for all FLCs, thus increasing the interpretability of the fuzzy models. To design these controllers automatically and efficiently, we developed the hybrid MGAPSO method, which simultaneously leverages the exploration capability of a GA and the convergence capability of PSO. Simulation results demonstrate that with our approach, various numbers of robots can be successfully controlled to safely navigate and reach their target simultaneously.

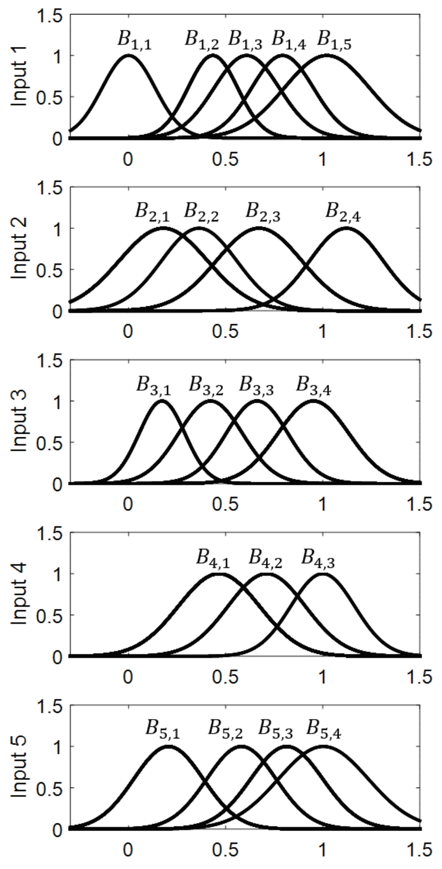


Figure 3.11: Fuzzy sets trained by the MGAPSO with interpretability improvement

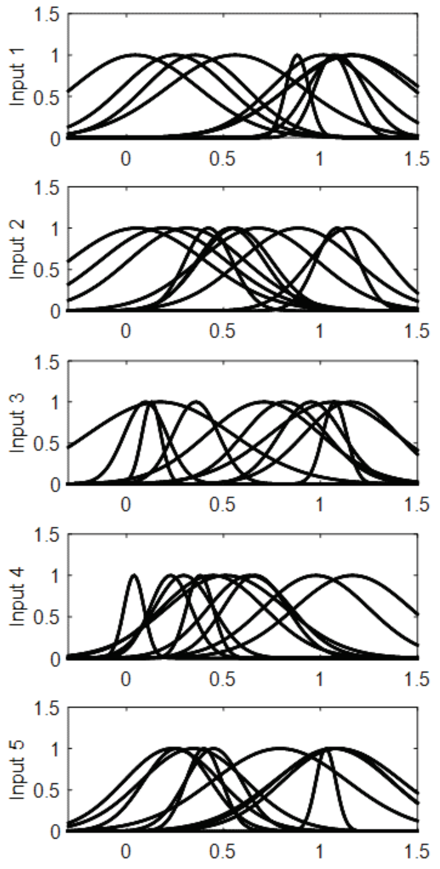


Figure 3.12: Fuzzy sets trained by the MGAPSO without interpretability improvement

Chapter 4

Fuzzy-Rule Interpretable Neural Fuzzy System

4.1 Background

The FISs convert learned knowledge into fuzzy rules that can be interpreted by the human user by optimising the transparency of the fuzzy sets. This mechanism allows human agents to understand machine agents' actions and states and then establish machine-to-human interaction in a human-machine MAT system. In addition to machine-to-human interaction, human-to-machine interaction should also be considered in a human-machine MAT system. This chapter presents an approach to identify the human states and their transition routes automatically, which allow machine agent to understand human cognitive states and their transition, and can be used to coordinate human and machine agents when making a decision.

Several approaches have been developed to identify human states. Haynes and Rees [51] proposed a decoding-based approach based on Functional Magnetic Resonance Imaging (fMRI) data to categorised unconscious and conscious mental states. In paper [52], the initial state of brain activity was identified by applying Principal Component Analysis (PCA) on magnetoencephalography (MEG) data. This initial state then was used to trace the brain activity in the state space as the subject received visual stimulus. These two researches considered spatial features that might not be able to discover time-dependent transition state. Taghia et al. [53] developed a system based on first-order Markov model to discover hidden brain state by analysing functional connectivity from fMRI data. Kringelbach et al. [54] developed a framework that models the state transition in the brain based on the information extracted from neuroimaging data. This framework also used Markov model to observe state changes. Both [53] and [54] provide time-dependent approaches that enable the transition state being observed. The approaches for brain state discovery discussed so far rely on fMRI or neuroimaging data that offer only second-level observations on the brain activity. However, the alteration in mental state occurs in millisecond-level. And EEG device can recode brain activity many times in a second, which provide more sophisticate information for brain state observation. This study therefore proposes an EEG-based approach to investigate brain state in a higher-resolution way.

To identify brain state via EEG data, the proposed approach is incorporated with unsupervised learning. Unsupervised learning has been widely used for state identification because it does not require external supervision [53], [55]–[58]. Thus, we used an unsupervised density-based clustering method to automatically identify brain states. One of the advantages of density-based clustering algorithms is that they do not require prior knowledge to determine the number of clusters, unlike centre-based clustering algorithms, such as K-means and K-medoids [59]–[62]. In addition, most centre-based clustering algorithms assign the data points to the nearest cluster centre, which might fail to identify clusters with an arbitrary shape [60], [62]. Density-based clustering algorithms, on the other hand, calculate the local density of data so that arbitrarily shaped clusters can be detected easily [60], [62], [63]

Apart from brain dynamics, the subjective difference is another factor affecting BCI performance significantly. Brain activities could have non-stationery and time-variant dynamics over time, and this addresses one more challenge for developing BCI [64]–[66]. To overcome the subjective difference, we consider the FISs to extract features from the brain dynamics. The fuzzification operation and if-then-rule architecture in FISs have been demonstrated to address uncertainty [43], [44], [67]–[69]. The fuzzy if-then-rule can model the dependency between input and output variables as an inference logic for different observable situations [43], [68]. This inference mechanism facilitates the modelling of human brain activities associated with different behaviours. The fuzzification operation translates the data into membership degrees using the Gaussian membership function and provides a tolerance for uncertainty, such as noise and variations in data [43], [44], [57], [67]–[69]. Fuzzy Neural Networks (FNNs) [43], [44], [66], [69], [70] amalgamating fuzzy inference systems into artificial neural networks are introduced in this study to model the continuous changes in brain states via a supervised learning approach. We assumed that every single fuzzy rule represents one specific brain pattern associated with a specific behaviour or latent mental process during the task. More specifically, the firing strength of a fuzzy rule can be the membership (degree) of a particular brain activity, which provides useful information to explicate the brain states [66], [69], [71]. Therefore, in this study, the firing strength of fuzzy rules is used as features to depict recorded EEG data, and then the unsupervised density-based clustering method is exploited to define external states as well as the covert states of the brain.

In addition to the identification of external and covert brain states, we introduced probability to determine the degree of change to other states by applying a Markov model [53], [72]–[75] to the captured brain states to obtain probabilities of transition between every pair of states. The transition probability of Markov model is not deterministic; it inherently cannot predict the future states from the present with certainty. However, this transition probability allows the user to find the possible feature states [53], [72]. We therefore can assign transition probability to brain states to generate moment-by-moment connectivity patterns, called the Covert State Transition Diagram (COSTD), which helps us to understand brain dynamics in relation to a cognitive task.

This study proposed a Fuzzy COSTD (FCOSTD) which is based on FNN and a density-based clustering approach. We then leverage FCOSTD to identify the dynamic changes of brain states (external and covert states) and explore the relationships between the human brain and behaviour performance.

4.2 Methodology

Figure 4.1 interprets the generation of the FCOSTD based on EEG signals. Here the FNN plays an important role, providing plentiful information to carry out unsupervised clustering, brain state representation and state change visualization. The fuzzy inference mechanism in the FNN is used to characterize frequency-domain EEG data. There are two advantages of using FNNs: 1) the antecedent components of the fuzzy rule can overcome the subjective uncertainty [76], [77]; 2) fuzzy inference mechanism in the FNN enables the neural network to extract knowledge from input patterns which can be interpreted by either machine or human [17], [36], [78].

4.2.1 Fuzzy Neural Network

To develop a brain-state-drift classifier that can overcome the subjective uncertainty of EEG signals and train the classifier with fewer segmental patterns, we exploited an FNN to classify incoming brain activity patterns. The structure of the FNN used in this study is shown in Figure 4.1. Assume the input $X(t) = [x_1(t), \ldots, x_n(t)]^T \in \mathbb{R}^{n \times 1}$, and the FNN has r rules. The layers of the network, from

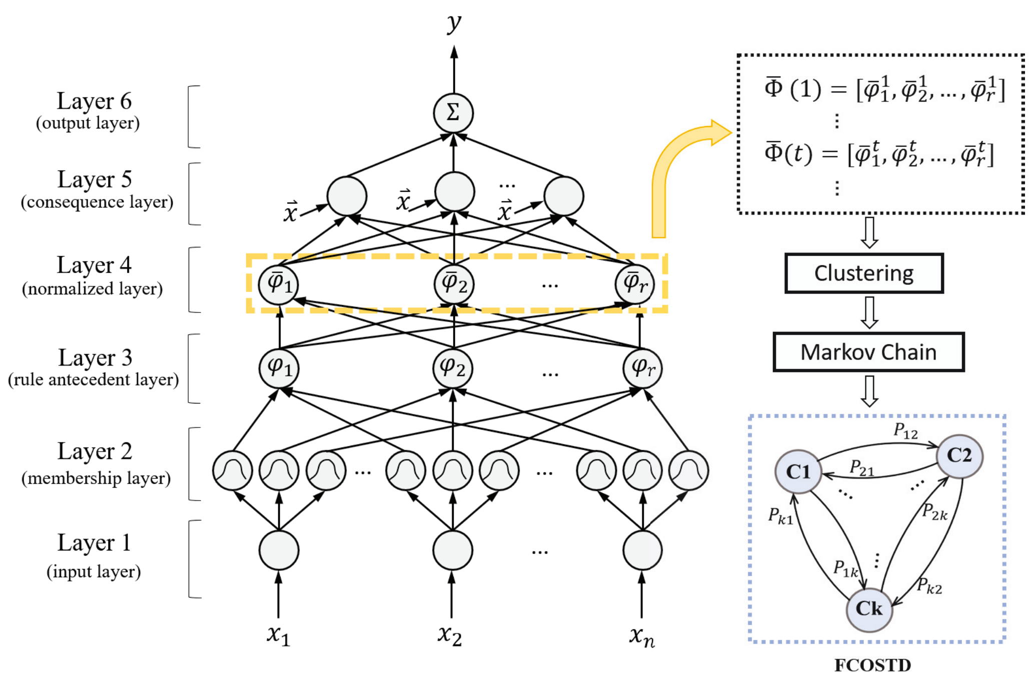

y
Layer 6
(output layer)
Layer 5
(consequence layer)
Layer 4
(normalized layer)
Layer 3
(rule antecedent layer)
Layer 2
(membership layer)
Layer 1
(input layer)
Σ
$\vec{x}$
$\bar{\varphi}_1$ $\bar{\varphi}_2$ ... $\bar{\varphi}_r$
φ_1 φ_2 ... φ_r
x_1 x_2 x_n
$\bar{\Phi}(1) = [\bar{\varphi}_1^1, \bar{\varphi}_2^1, \ldots, \bar{\varphi}_r^1]$
$\bar{\Phi}(t) = [\bar{\varphi}_1^t, \bar{\varphi}_2^t, \ldots, \bar{\varphi}_r^t]$
Clustering
Markov Chain
C1
C2
Ck
P_{12} P_{21} P_{k1} P_{1k} P_{2k} P_{k2}
FCOSTD

Figure 4.1: The proposed FCOSTD. The fuzzy state identification includes a fuzzy neural network, clustering, and Markov chain, and then the FCOSTD can be generated

the first to the fifth, realize a fuzzy-rule-based inference model of the following form:

$$\textit{Rule } j: \text{ IF } x_1 \text{ is } A_{1j} \text{ AND } \ldots \text{ AND } x_n \text{ is } A_{nj}, \text{ THEN } w_i = a_{0j} + a_{1j}x_1 + \ldots + a_{nj}x_n, \tag{4.1}$$

$$\begin{aligned} \textit{Rule } \text{j}: \text{ IF } x_{1j} \text{ is } A_{1j} \text{ AND } x_{2j} \text{ is } A_{2j}\text{AND}x_{3j} \text{ is } A_{3j}, \\ \text{THEN } \theta_j = b_{0j} + \textstyle\sum_{i=1}^{3} b_{ij}x_{ij}, \end{aligned}$$

where A_j^i is the i-th fuzzy set defined on the j-th input variable and $a_0^i, \ldots, a_n^i$ are the coefficients of the linear consequent of the rule. Note that the number of the consequent parameters corresponds to the dimension of the input variable X. To illustrate the structure of the FNN in detail, the FNN is introduced in this section layer by layer.

Layer 1 (input layer) The number of nodes in this layer equals the length of one input pattern vector. Each node, which performs no computation, transmits input data to the next layer. The outputs of the j-th node in this layer can be defined as

$$O_j^{(1)} = x_j(t). \tag{4.2}$$

Layer 2 (membership layer) In layer 2, each node represents a fuzzy set and evaluates the membership value of an input variable from layer 1 with the corresponding membership function. In other words, the nodes in layer 2 fuzzify the input data, transferring the crisp values to the linguistic levels. The membership functions can be either triangular, trapezoidal, or Gaussian. We use a Gaussian membership function in this study. Thus, the output of each layer-2 node can be calculated as

$$O_{ij}^{(2)} = \mu_j^i = exp\left[-\left(\frac{O_j^{(1)} - m_j^i}{\sigma_j^i}\right)^2\right], \tag{4.3}$$

where μ_j^i is the membership value corresponding to the j-th input variable of the i-th rule and m_j^i and σ_j^i are, respectively, the centre and the width of the Gaussian membership function.

Layer 3 (rule antecedent layer) The nodes in layer 3 are called rule nodes. A node in layer 3 represents one fuzzy rule and receives corresponding membership values from the nodes in layer 2 to compute the firing strength value of the rule by performing a fuzzy AND operation. Each node in layer 3 receives single-dimensional membership degrees from appropriate nodes in Layer 2 to form the antecedent clause of a rule. Here, fuzzy AND operation is utilized to calculate the node outputs, taking the algebraic product of the membership values of the atomic clauses associated with a rule. Given r rules, the output of each layer-3 node can be computed as

$$O_i^{(3)} = \varphi_i = \prod_j^n O_{ij}^{(2)}. \tag{4.4}$$

Layer 4 (normalization layer) The size of this layer equals the size of layer 3. Each layer-4 node represents a normalized rule antecedent whose outputs are calculated by

$$O_i^{(4)} = \bar{\varphi}_i = \frac{O_i^{(3)}}{\sum_k^r O_k^{(3)}}. \tag{4.5}$$

Layer 5 (consequence layer) Layer-5 nodes are called consequent nodes. Each node receives the output delivered from layer 4 and the input variables of layer 1. The output of layer 5 is obtained as follows:

$$O_i^{(5)} = w_i \, O_i^{(4)} = \left(\sum_j^n a_j^i x_j + a_0^i \right) \, O_i^{(4)} \ . \tag{4.6}$$

Layer 6 (output layer) A node of layer 6 corresponds to one output variable. Here, the nodes operate integration as a defuzzification process with

$$O^{(6)} = y(t) = \sum_i^r O_i^{(5)} \ . \tag{4.7}$$

4.2.2 Parameter learning of the FNN

The parameter-learning phase carries out supervised learning to optimize all the free FNN parameters. In this study, the parameters of fuzzy rules are learned via the gradient descent (GD) algorithm. Considering the single-output case for simplicity, the error function E to be minimized is defined by

$$E = \sum_{t=1}^N \frac{1}{2} \left[y\left(t\right) - y_d\left(t\right) \right]^2, \tag{4.8}$$

where N is the number of training samples and y_d is the desired output for the layer-6 nodes. Denote that $\hat{m}_j^i$ and $\hat{\sigma}_j^i$ are the new centre and width of the Gaussian membership function during the training, and can be updated as follows:

$$\hat{m}_j^i = m_j^i - \eta \frac{\partial E}{\partial m_j^i},$$

$$\frac{\partial E}{\partial m_j^i} = \sum_{t=1}^{N} \sum_{i=1}^{r} \frac{\partial E}{\partial y} \frac{\partial y}{\partial \varphi_i} \frac{\partial \varphi_i}{\partial \mu_j^i} \frac{\partial \mu_j^i}{\partial m_j^i} \tag{4.9}$$

$$= \sum_{t=1}^{N} \sum_{i \in \varphi(j)} \left[y\left(t\right) - y_d\left(t\right) \right] \frac{\left[w_i - y(t) \right] \varphi_i}{\sum_k^r \varphi_k} \frac{\left(x_j - m_j^i \right)}{\left(\sigma_j^i \right)^2},$$

where $w_i = \sum_j^n a_j^i x_j + a_0^i$. And,

$$\hat{\sigma}_j^i = \sigma_j^i - \eta \frac{\partial E}{\partial \sigma_j^i},$$

$$\frac{\partial E}{\partial \sigma_i^j} = \sum_{t=1}^{N} \sum_{i=1}^{r} \frac{\partial E}{\partial y} \frac{\partial y}{\partial \varphi_i} \frac{\partial \varphi_i}{\partial \mu_j^i} \frac{\partial \mu_j^i}{\partial \sigma_j^i} \tag{4.10}$$

$$= \sum_{t=1}^{N} \sum_{i \in \varphi(j)} \left[y\left(t\right) - y_d\left(t\right) \right] \frac{\left[w_i - y(t) \right] \varphi_i}{\sum_k^r \varphi_k} \frac{\left(x_j - m_j^i \right)^2}{\left(\sigma_j^i \right)^3}.$$

The consequent components can be updated with

$$\begin{aligned} \hat{a}_j^i &= a_j^i - \eta \frac{\partial E}{\partial a_j^i}, \\ \frac{\partial E}{\partial a_j^i} &= \sum_{t=1}^{N} \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_i} \frac{\partial w_i}{\partial a_j^i} = \sum_{t=1}^{N} \left[y\left(t\right) - y_d\left(t\right) \right] \frac{\varphi_i}{\sum_k^r \varphi_k} x_j, \end{aligned} \tag{4.11}$$

where $\hat{a}_j^i$ is new consequent component. Note that x_0 is set to 1 for updating a_0^i.

4.2.3 Covert State Transition Diagram

After the optimised FNN is obtained, the FCOSD is generated to represent the relationships among the defined states. Figure 4.1 shows the generation of the FCOSD. First, the outputs of the layer-4 nodes (normalised layer) are extracted and encoded in a vector $\bar{\Phi}(t) = [\bar{\varphi}_1^t, \bar{\varphi}_2^t \ldots, \bar{\varphi}_r^t]$. Afterwards, an unsupervised clustering approach is adopted to categorise rule-node vectors over all trials into K clusters. According to the dominant label in each cluster, the K cluster can be defined as K states and connected to corresponding real behaviours or physical meanings by external labels of each clusters. The relationships among the K states are then established to form a covert state transition diagram.

Density-based clustering algorithm We use the Density Peaks Clustering Algorithm (DPCA) [62] to cluster normalised firing strength vectors $\bar{\Phi}$. Here, we define that a strength vector $\bar{\Phi}$ is a data point. To find cluster centres for the whole set of $\bar{\Phi}$, DPCA computes the local density ρ_i and the distance δ_i to higher-density points for each datum point i. We use the Euclidean metric to calculate distances $d_{ij} = \|\bar{\Phi}_i - \bar{\Phi}_j\|$. The local density ρ_i is estimated by a Gaussian kernel function defined as

$$\rho_i = \sum_{j \neq i} exp\left[-\left(\frac{d_{ij}}{d_c}\right)^2\right], \tag{4.12}$$

where $d_{ij}$is the Euclidean distance between data points i and j,and d_c is the cut-off distance predefined by the user. Here, d_c is set according to Rodriguez's study [62], and a chosen d_c can satisfy that the average number of neighbours is approximately 20% of the total number of data points, e.g.,$d_c = 0.07$. Given $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_N$, where N is the number of data points, and then δ_i can be determined by

$$\delta_i = \max_{\{j:\rho_j > \rho_i\}} d_{ij}. \tag{4.13}$$

Here δ_i represents the maximum distance between datum point i and other higher-density points. With these two quantities, we can define a γ_i value as follows:

$$\gamma_i = \rho_i \delta_i. \tag{4.14}$$

A high γ_i value of a datum point reflects that it simultaneously has crowding neighbours and is located far from other higher-density points. We can use this γ_i value to determine cluster centres. For other on-central points, clustering assignment is implemented by the nearest-neighbour approach as follows:

$$k = \begin{cases} \underset{j:\rho_j>\rho_i}{\arg\min}\, d_{ij}, & \rho_i < \max_j \rho_j \\ \text{i} & \text{otherwise} \end{cases} \tag{4.15}$$

where k is the index of the neighbour datum point. If datum point j closest to datum point i and of which local density ρ_j is highest, then k is assigned as j; otherwise, k is assigned as i. Afterwards, each non-central point is assigned to the same cluster as its neighbour datum point of higher density.

Visualization of state changes To take a solid consideration for state representation of the proposed system, we must not only determine the covert states but also the relationships among them. We build a cover-state transition diagram taking transition probabilities into account to discover the temporal changes of the observed system via a Markov chain [72], [73], [75]. All connections among observable states are updated based on the state transition matrix. Given K states found by the DPCA, the state transition matrix can be defined as

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1K} \\ \vdots & \ddots & \vdots \\ p_{K1} & \cdots & p_{KK} \end{bmatrix}. \tag{4.16}$$

The columns of the matrix refer to the state for the next time point, and the rows of the matrix refer to the current state. Each transition probability p_{ij} between two states is estimated by the maximum likelihood approach, which is $p_{ij} = T_{ij}/L$, where T_{ij} is the number of direct transitions from state i to state j and L is the total number of times the system arrived at the state i, irrespective of how it has arrived. Thus $\sum_{j=1}^{K} p_{ij} = 1; i = 1,\ 2,\ \ldots, K$. Once the state transition matrix is obtained, we can observe the path of transition from state to state as the observed system reacts to the stimuli or events.

4.3 Experiment

4.3.1 Experimental Design

The human brain is a complex system, and brain dynamics change rapidly regarding stimuli, tasks or emotion. To demonstrate the feasibility of the proposed FCOSTD, the EEG signals recorded from the distracted driving experiment were selected because the participant must switch their attention between the designed tasks over time. All electrodes of the EEG cap were referenced to linked two mastoids of the participant, and a single ground electrode was attached to the forehead. All references and ground are built in the device. The experiment was designed in a driving simulator on a dynamic 6-degree-of-freedom motion platform with 360 °driving scenes rendered on the seven LCD projectors [79], [80]. The simulator provides authentic visual and kinesthetic stimuli of the driving situation on a highway to the participants. This means that the participants experience real-time kinematic feedback from the motion platform regarding their operations, such as turning the steering wheel. The simulator emulates car cruising at a fixed speed of 100 km/hr on a highway scene in the night throughout the entire experiment.

There are two tasks: one driving task and a mathematical problem-solving task. In the driving task, the car deviates to either the left or right from the cruising lane randomly, and the participants are asked to turn the steering wheel to control the car back to the cruising lane immediately. In the mathematical problem-solving task, the participants need to verify a two-digit addition equation (i.e., whether the equation is correct or incorrect) displayed on the front screen when the event comes in. The numbers of correct and incorrect equations are the same, and the difficulty level is set the same as well. There are two buttons mounted on the left and right sides of the steering wheel. The participants can easily use the right thumb to press the button mounted on the right side of the steering wheel to report the correctness of the mathematical equation. To evaluate behaviour performance, the latency between even onset to proper response is defined as reaction time (RT). For the driving task, the RT is the latency between the deviation onset and turning the steering wheel. Similarly, the latency between the presentation of the equation and the button press is the RT of the mathematical task.

Eleven neurologically healthy volunteers aged 20–28 years participated in a distracted driving experiment. Continuous EEG data were recorded during the time a participant was involved in the experiment. All participants were required to have a driver's license and good driving habits. This experiment was conducted at National Chiao Tung University, Hsinchu, Taiwan [79], [81]. Every participant had two training sessions before the experiment to acquaint themselves with the two tasks and the virtual-reality environment. After two training sessions, every participant had four 15-min experimental sessions separated by 10-min rests. A more detailed explanation of the designed distracted driving can be found in our previous studies [79], [80].

4.3.2 EEG Data Preprocessing

In the preprocessing phase, the continuous EEG recordings are filtered using a zero-phase finite impulse response (FIR) band-pass filter, which has been widely used for EEG signal prepossessing [82]–[84]. We set lower cut-off frequency to 1 Hz and higher cut-off frequency is 50 Hz. The selected cut-off frequencies enable the filter to capture all the variations in event-related potentials (ERPs) of collected EEG signals in the designated bandwidth, and filter out the noise from power line, movement-related artifacts, and sweat artifacts. Others wide variety of artifacts are then removed via Independent Component Analysis (ICA) [85], [86]. The ICA can recover independent components from the recorded signals maximizing the degree of statistical independence of the estimated components [79]. Once the artifact-free EEG data are produced, it was segmented into a 1.2-second interval from the time an event is on-set, and we call it a trial. Then, every trial is further divided into 17 segments with overlapping 400-ms windows advancing in 50-ms steps. Every 400-ms sample was transferred from the time domain to the frequency domain using the Fast Fourier Transform (FFT). To reduce the computational complexity, six components, including the frontal, central, parietal, occipital, left motor and right motor regions, are selected for further processing. The mean powers in the delta (1–3 Hz), theta (4–7 Hz), alpha (8–12 Hz), and low beta (13–20 Hz) bands were then extracted from the selected components. Note that previous studies [65], [79], [87], [88] have demonstrated that the brain dynamics in these selected components are highly associated with visual perception, visual processing, motor control, attention and planning. Papers [79] and [80] also suggest that mental states of subjects can change significantly within 1.2 seconds after events is on-set during the driving. For the training process of the FNN, each segment is assigned a label, either driving or math. The total number of input variables fed to the FNNs was 24 (4 bands $\times$ 6 components).

4.3.3 Experimental Results

The FNN, in this study, is rained subject-by-subject because all subjects have various brain activity patterns. The output of FNN is binary output; 0 represents driving and 1 is math. We use mean squared error (MSE) as the loss function for training. The learning rate for update rule of gradient descent algorithm is set to 0.001, and the learning epoch is 100. The single-subject performance of all experiments is shown in Table 4.1. The second and third rows, respectively, present the number of rules that FNN generated and its recognition rate. The fourth and fifth rows present the performance of clustering on firing strength vectors $\bar{\Phi}$ extracted from

Table 4.1: The performance of FNN and clustering on the distracted driving experiments. The number in the first row represents the index of the subjects. The second and third columns present the number of rules that FNN generated and its recognition rate, respectively. The performance of clustering is evaluated by using the Silhouette score and Fowlkes-Mallows Score, as shown in the fourth fifth columns, respectively.

Subjects Index	**Number of Rules**	**Recognition Rate (%)**	**Silhouette Score**	**Fowlkes-Mallows Score**
subject_1	14	93.36	0.56	0.56
subject_2	14	93.87	0.53	0.49
subject_3	15	93.74	0.59	0.58
subject_4	12	87.82	0.52	0.57
subject_5	17	96.09	0.39	0.67
subject_6	13	89.99	0.61	0.56
subject_7	16	92.73	0.43	0.53
subject_8	13	94.58	0.39	0.62
subject_9	15	93.42	0.5	0.54
subject_10	19	93.55	0.36	0.53
subject_11	15	93.4	0.47	0.55

the trained FNN. The performance of clustering is evaluated by using the Silhouette score [89] and the Fowlkes-Mallows score [90]. The Silhouette score in Table 4.1 is the mean value across the estimated compactness values of each cluster, while the Fowlkes-Mallows score shown in the next row (in its right lower box) evaluates the geometric mean of the precision and the recall. The number of clusters for all subjects is determine by γ_i value in formula 4.14, more details are described in section 4.4.1.

4.3.4 Case Study

Case study — Distracted driving (subject_1) Figure 4.2 shows the content of each clustering, and each index on the horizontal axis represents a cluster, i.e., $Cn, n = 1, 2, 3$, and 4, represent the n-th cluster. There are four clusters generated by the collected data from the first subject. Each cluster has its own dominating label, of which the occupancy rate is the greatest. The math label (the blue part of a bar) indicates that the participant doing or paying attention to the mathematical problem-solving task, while the driving label (the saffron part of a bar) represents the lane-keeping driving task. Table 4.2 reveals the distribution percentage of each label in different clusters. In Figure 4.2, the clusters $C1$ and $C2$ are dominated by math labels, which means that for most of the trails included in these two clusters this subject allocates most of the brain resources to math tasks. Particularly, there are more samples labelled as solving math in $C2$. Based on the label distribution, we define $C1$ and $C2$ as a pure calculating and a covert state from mental calculating to driving, respectively. There are more samples labelled as driving in $C3$ and $C4$ (ref. Figure 4.2). Here, the state (cluster) $C4$ is almost a pure state representing driving behaviour due to the 96.93% occupation rate in the driving label. The state $C3$ consists of 20.4% math labels and 79.60% driving labels. Thus, it is reasonable to consider $C3$ a covert state of driving but relatively close to the calculating behaviour. Figure 4.3 is a state transition diagram for subject_1. The four solid circles on the diagram correspond to the four different clusters/states, as shown in Figure 4.2. The directed connection between every two states represents the potential transition with the probability indicated therein. Table 4.3 summarises the transition probabilities between every two states. For example, between $C1$ and $C2$, there are two connection lines: one starts at $C1$ and ends at $C2$, and the other is the opposite direction. $C1$ and $C2$ exhibit bidirectional transitions. More specifically, the conditional probability of transition from $C1$ to itself ($C1$) is 0.8653 and that from $C1$ to $C2$ and $C3$ are 0.0943 and 0.0404, respectively. This probability of transition from $C1$ to any other state including itself does not depend on the state of the system before it came to $C1$ (i.e., it is a Markov chain). It is worth noting that there is no transition from $C1$ to $C4$ which is very reasonable because $C1$ is a pure Math-state and $C4$ is almost a pure driving state. For Subject 1, although there is a very low probability of transition from $C4$ to $C1$, there is no direct transition $C1$ to $C4$. A natural question comes – are there indirect transitions from $C1$ to $C4$. Further analysis of the data shows that there was no indirect transition from $C1$ to $C4$ via $C2$ or $C3$. Here, by indirect transition we refer to the situation when the system switches from $C1$ to $C2$ and then $C2$ to $C1$ in two successive time instants, i.e., without making any transition from $C2$ to itself. This again emphasizes the stability of the identified brain states. It is interesting to note that $C2$ is not a pure state, but this is a very stable internal (Covert state) because the conditional probability of transition from $C2$ to itself is 0.9343. Consequently, the conditional probability of transitions to $C1$, $C3$ and $C4$ are very small, 0.0131, 0.0488, and 0.0038, respectively. Similarly, the covert state $C4$ is a very stable one, for which the conditional probability of transition from $C4$ to itself is the highest, 0.9791.

Case study — Highly focusing on driving (subject_2) The number of clusters used for subject_2, as shown in Figure 4.4, is also four, which is the same as that of subject_1. Table 4.4 shows the distribution percentage of each label in all four clusters. The samples categorised in $C1$ are all math labels. $C2$ has math and driving labels mixed at 78.03% and 21.97%, respectively. In $C3$, the driving label

Table 4.2: The distribution percentage of each label in all clusters of subject_1.

Unit: %	$C1$	$C2$	$C3$	$C4$
Math Label	99.67	95.65	20.4	3.07
Driving Label	0.33	4.35	79.6	96.93

Table 4.3: Transition probability between each two identified states of subject_1.

	$C1$	$C2$	$C3$	$C4$
$C1$	0.8653	0.0943	0.0404	0
$C2$	0.0131	0.9343	0.0488	0.0038
$C3$	0.0052	0.0247	0.9584	0.0117
$C4$	0.0058	0.0175	0.0975	0.8791

occupies 86.91%, while the math label occupies 16.81%. $C4$ contains almost pure driving labels of approximately 96.55%. Therefore, we define $C1$ as a pure calculating behaviour, $C2$ as the covert state from calculating to driving behaviour, $C3$ as from driving to calculating to driving behaviour, and $C4$ representing pure driving behaviour. Figure 4.5 is the corresponding state transition diagram for subject_2. Table 4.5 shows the conditional transition probabilities between every two states for subject_2. Like subject_1, the high conditional probability of transition from any of the four states to itself (more than 0.92) suggests that each of them represents a valid state. Unlike sublect_1, for subject_2, there is no transition from $C4$ to $C1$ and a very weak transition probability from $C1$ to $C4$. This is just the opposite of subject_1. A possible reason for this may the fact that for subject_2, $C4$ is almost a pure driving state and $C1$ is almost a pure math sate, but $C4$ is more strongly represented (with much more number of trials mapped in $C4$) than $C1$. On the other hand, for subject_1, $C1$ a pure math state is much more strongly represented than the almost pure driving state $C4$. So from these two subjects, what we observe is that between two pure state, if one is represented by a smaller number of trials, then there is a low chance of transition from the weaker one to the stronger one. Like Subject 1, there is no indirect transition between $C4$ and $C1$ (ref. Figure4.5).

Table 4.4: The distribution percentage of each label in all clusters of subject_2.

Unit: %	$C1$	$C2$	$C3$	$C4$
Math Label	100	78.03	16.81	3.35
Driving Label	0	21.97	83.19	96.65

Table 4.5: Transition probability between each two identified states of subject_2.

	$C1$	$C2$	$C3$	$C4$
$C1$	0.9401	0.026	0.0286	0.0052
$C2$	0.0245	0.9294	0.046	0
$C3$	0.0136	0.0181	0.9456	0.0227
$C4$	0	0	0.0488	0.9512

Case study — Rapid switching (subject 3) Figure 4.6 shows four clusters obtained from the experiment of subject_3. The distribution percentage of each label in all clusters is revealed in Table 4.6. $C1$ contains an almost pure math label of approximately 98.12%. $C2$ has math and driving labels of 94.43% and 5.57%, respectively. In $C3$, the driving label occupies 88.66%, while the math label is 11.34%. $C4$ consists of 91.29% of the driving label and 8.71% of the math label. Therefore, we define $C1$ as a pure calculating behaviour, $C2$ as a covert state from calculating to driving behaviour, $C3$ as from driving to calculating to driving behaviour, and $C4$ as pure driving behaviour. Figure 4.7 is the state transition diagram for subject_3, and the corresponding transition probabilities between every two states are shown in Table 4.7. Table 4.7 reveals that the conditional transition probabilities from $C1$ to $C4$ and from $C4$ to $C1$ are zero. From Figure 4.6, we find that subject_3 exhibits similar characteristics with one major difference both of the pure states $C1$ and $C4$ are represented by similar number of trials and number of trials representing of each of the two states are much smaller than the two covet states. In this case, there is no transition between $C1$ and $C4$. If there is any transition from $C1$ to $C4$ or from $C4$ to $C1$, the transitions must be via the covert states $C2$ or $C3$. Except the transition between $C1$ and $C4$, the other pairs exhibit transition between the states, although some conditional transition probabilities are very low as expectation. For this subject there is a very low transition probability between $C1$ and $C4$ via $C2$. According to our investigation, the Transition probabilities of $C1$-$C2$-$C4$ and $C4$-$C2$-$C1$, both are very low.

Table 4.6: The distribution percentage of each label in all clusters of subject_3.

Unit: %	$C1$	$C2$	$C3$	$C4$
Math Label	98.12	94.43	11.34	8.71
Driving Label	1.88	5.57	88.66	91.29

Table 4.7: Transition probability between each two identified states of subject_3.

	$C1$	$C2$	$C3$	$C4$
$C1$	0.9189	0.0676	0.0135	0
$C2$	0.0186	0.9451	0.0275	0.0088
$C3$	0.006	0.0207	0.9396	0.0337
$C4$	0	0.0432	0.0504	0.9065

4.4 Analysis

4.4.1 Determining the number of clusters

In Figure 4.8 the panels (a), (b) and (c) depict the γ values of subject_1, subject_2, and subject_3 in decreasing order, respectively. Each solid circle represents one data

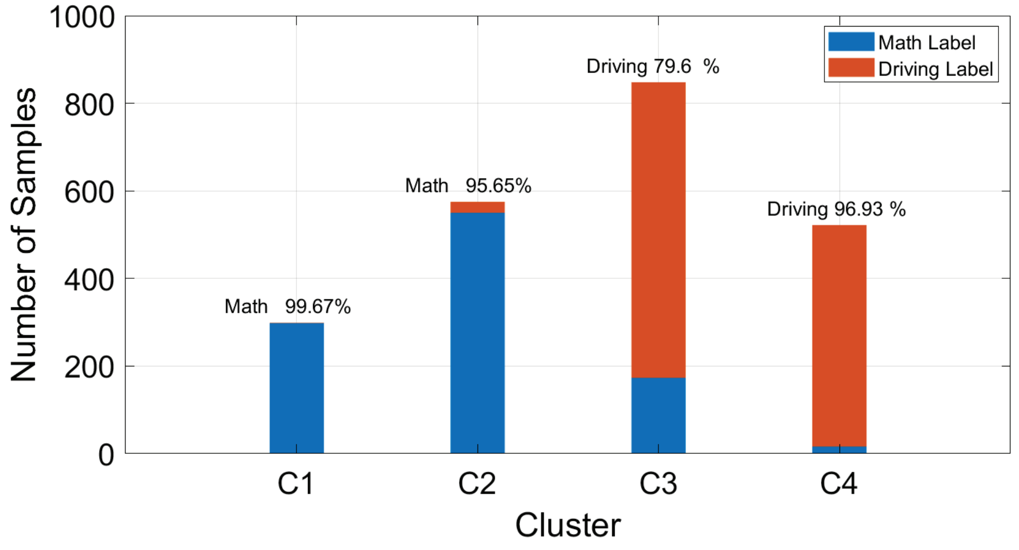


Figure 4.2: Clustering results of subject_1. The math label and driving label refer to the mathematical problem-solving task and driving task, respectively. The percentage over each bar represents the sharing rate of the dominating label.

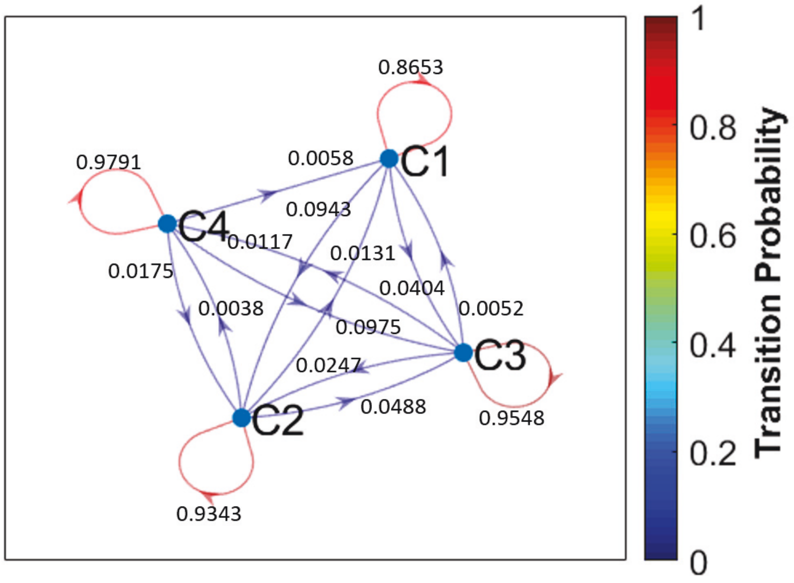


Figure 4.3: State transition diagram of subject_1. The digits adjacent to each connection line are the transition probabilities.

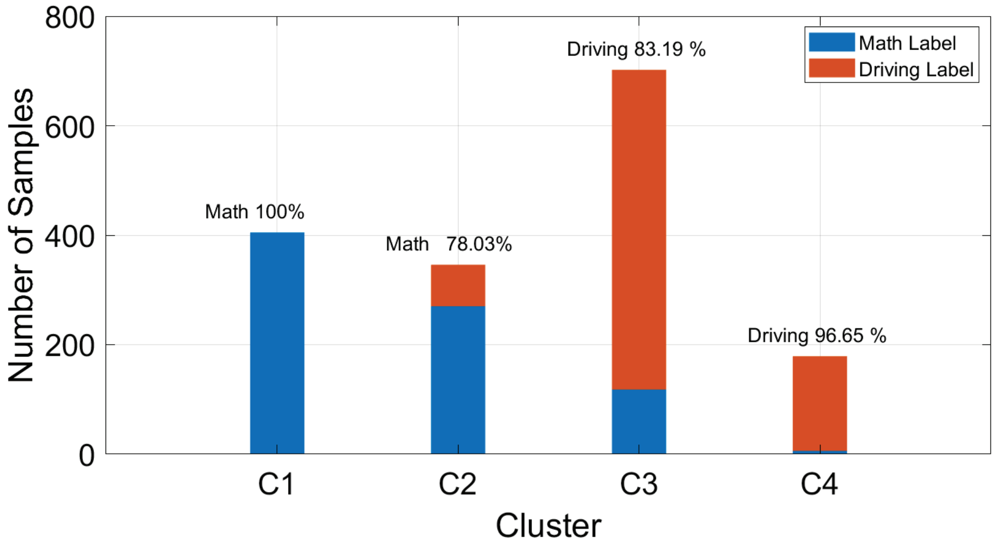


Figure 4.4: Clustering results of subject_2. The math label and driving label refer to the mathematical problem-solving task and driving task, respectively. The percentage over each bar represents the sharing rate of the dominating label.

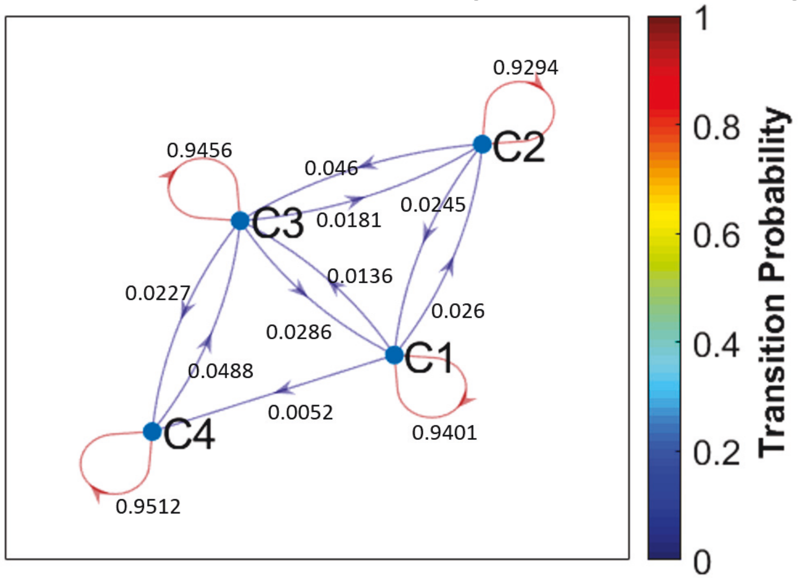


Figure 4.5: State transition diagram of subject_2. The digits adjacent to each connection line are the transition probabilities.

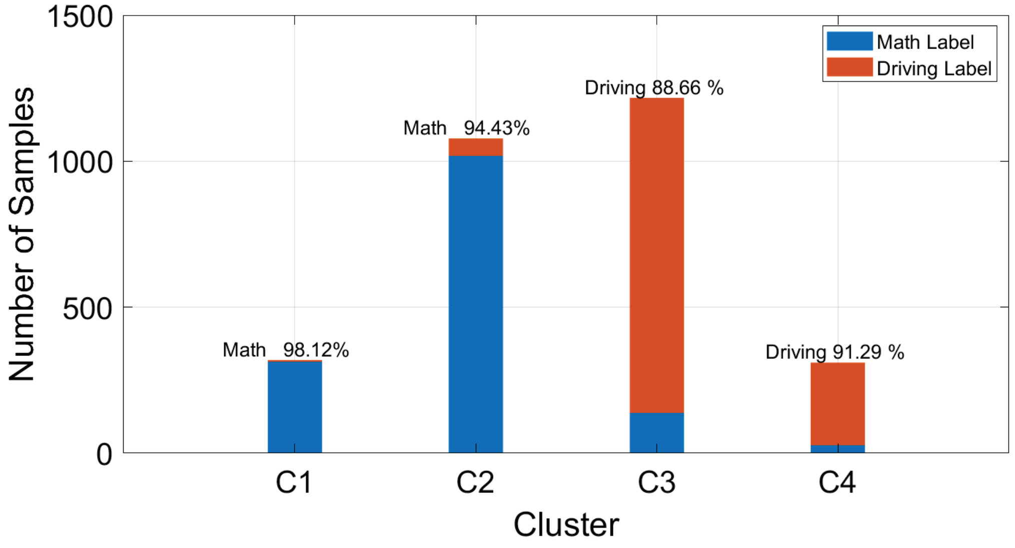


Figure 4.6: Clustering results of subject_3. The math label and driving label refer to the mathematical problem-solving task and driving task, respectively. The percentage over each bar represents the sharing rate of the dominating label.

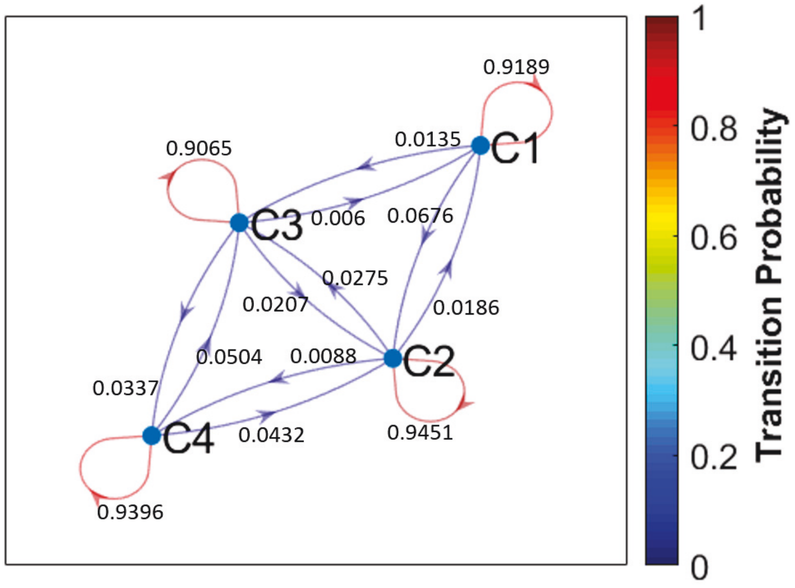


Figure 4.7: State transition diagram of subject_3. The digits adjacent to each connection line are the transition probabilities.

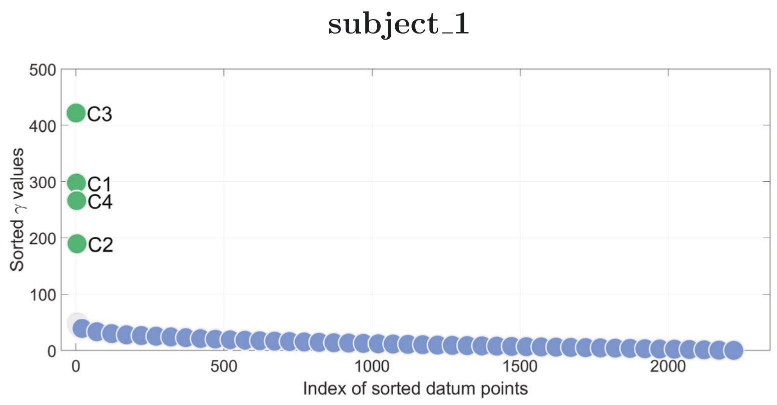
point, in which the label indicates the associated cluster. Figure 4.8 lists only samples with the top fifty percentages of sorted γ values. The γ value can be used as a criterion to select cluster centres. The sorted γ values approximately follow a power low, as shown in Figure 4.8. There are four isolated points strikingly distinctive from those who are close to the horizontal axis. As explained earlier, if a point has a high γ_i value (Ref. formula 4.14), then that point has high-density crowding neighbours and is located significantly away from other higher-density points. Therefore, these distinctive points are selected as centres for the clustering.

For example, in Figure 4.8(a), the first point on the top has the highest value of the product of highest density and longest distance from other relatively high-density points as well. Thus, this sample is defined as a centre. The rank-two sample is defined as a second centre due to its second highest γ implying that it is a high-density point as well as distant from other high-density points. Followed by the third and fourth centres, they both are high γ value samples. For the remaining non-isolated samples in Figure 4.8(a), they cannot be the centre because the γ values of these instances are relatively small than those in the selected four isolated cases. Then we take these four data points as cluster centres and form four clusters around them for subject_1, as shown in Figure 4.2.

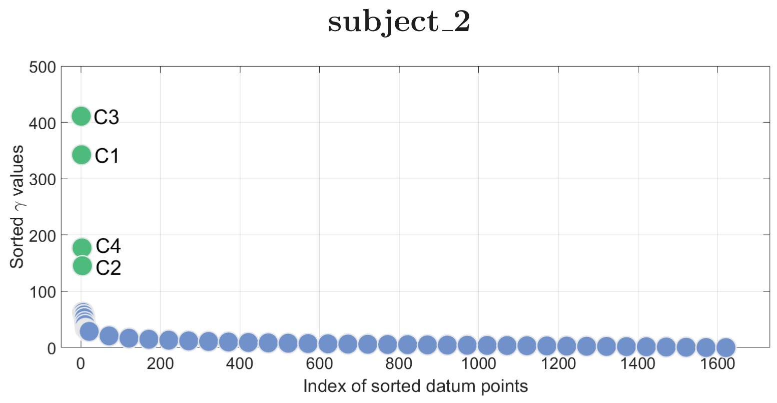
These four top-ranked samples belonging to $C3$, $C1$, $C4$ and $C2$ correspond to clusters 3, 1, 4 and 2, respectively, as labelled in Figure 4.2. Analysing the clusters, we have exhibited that these four clusters are associated with four different types of brain states associated with handling lane-keeping tasks and mathematical calculation tasks. The same strategy is also applied to subject_2 and subject_3 and surprisingly for each of them the system discovers four clusters representing four brain states.

4.4.2 State change analysis

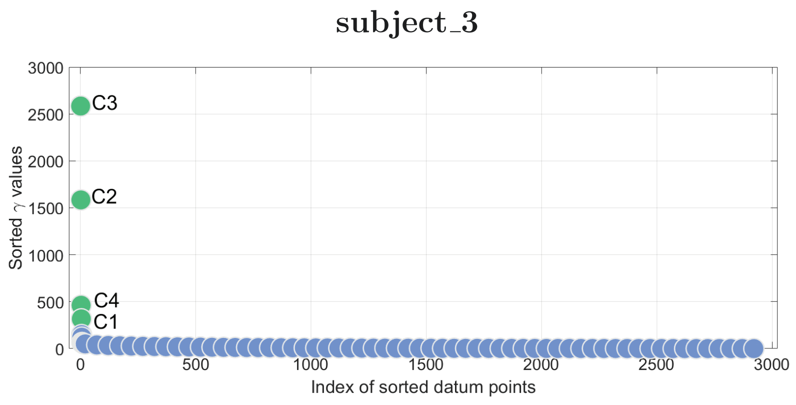
We leveraged FCOSTD to explore the changes in human states through brain dynamics during driving. Note that the changes in human state can be linked with human behaviour as humans perform lane-keeping tasks or mental calculations during distracted driving. In this study, we demonstrated the changes in human states by brain dynamics and listed two different results in which two participants applied different strategies when facing a high cognitive task (mental calculation) during driving. The brain dynamics evolve continuously, and the samples are extracted only over 1200 ms. The lane-keeping task mainly involves motor reflection in which the visual perception and motor controlling resources are mainly involved [80], [91]. Meanwhile, visual perception, visual memory, and executive functions in the human



(a)



(b)



(c)

Figure 4.8: Distribution of sorted γ values in descending order. Only those samples with the top 50 percent high γ values are shown in (a), (b)and (b). The indices on each dot indicate to which cluster belongs.

brain are highly associated with the process of mental calculation [80], [92]. Based on the findings of cognitive functions in the previous studies [79]–[81], here we investigate the state changes over 1200 ms period and leverage the behaviour performance (RTs) to infer the state changes as performing math and driving during distracted driving.

Figures 4.9 reveals behaviours and changes of the four states during the experiment; (a), (c) and (e) present the behaviour drifting in the trials of the mathematical problem-solving task, whereas Figures 4.9 (b), (d) and (f) present the trials of the lane-keeping driving task for the three subjects respectively. The left panels of each sub-figure present the sorted RTs of the subject towards the corresponding task in every trial. The right panels show the drifting of subjects' brain states during each trial when the subject deals with the corresponding task, in which every trial is sorted according to RTs. Table 4.8 illustrates the switching time of each state. Let us elaborate on how the switching time is computed. Each trail is of 1200 ms (i.e., 1.2 sec). Each trial is divided into 17 segments, each segment is of length 400ms with an overlap of 350 ms between two successive segments. Suppose the kth segment of a trial is in state Cx ($x = 1, 2, 3, 4$), and the $(k + 1)$th segment of that trial is in some other state Cy ($y \neq x$) then the switching time for that trial for state Cx is $(1.2/17) \times (k+1)$. In Table 4.8 we report the average value of the switching time for each state along with its standard deviation. Each state is counted across all trials, and the statistical data are shown in Table 4.8.

In Figure 4.9, these three selected subjects had very different behavioural performance, and they had relatively high performance in the experimental results (see Table 4.1). Subject_1 had better performance (average RT is 1.596 sec) in the mathematical problem-solving task compared with subject_2 (average RT is 1.709 sec), see Figure 4.7 (b) and (d). On the other hand, subject_2 could react to the lane-keeping driving task quickly (ref. Figure 4.9 (a) and (c)). Subject_1's fast response to mental calculation implies that subject_1 may be good at problem solving. Furthermore, subject_3 behaved better than subject_1 and subject_2 in both the mathematical problem-solving task and lane-keeping driving task (ref. Figure 4.9 (e) and (f)).

When subject_1 performs mental calculation during driving, the subject's present state switches quickly from the pure mental calculation ($C1$) to covert states ($C2$ or $C3$) in many trials, see Figure 4.9 (a), in which the average time of switching from $C1$ to other states is 0.4228 ms (Table 4.8). Apart from the mental calculation, the low switching time from $C4$ to other states implies that subject_1 can quickly allocate brain resources to finish the lane-keeping driving task and re-allocate the whole brain resource for the next task of mathematical calculation. The average switching time from $C4$ to other states is 0.4291 ms. Figure 4.3 implies the same phenomena. The diagram shows a fully connected diagram except for the connection from $C1$ to $C4$; the brain states of subject_1 can switch from $C1$ to $C2$ or to $C3$ and from $C4$ to other states but from $C1$ to $C4$. Subject_1, therefore, needs more brain resources to switch the brain state to take action when the car deviates from the lane.

In contrast, subject_2 exhibits significantly different behaviour and states compared to those of subject_1, see Figure 4.4 (c) and (d). The states of subject_2 change less frequently, and the average RT is lower when performing mathematical problem-solving. Subject_2 allocates higher brain resources for mental calculation and has a delay in switching between states compared to subject_1. Its transition diagram also reflects this observation, as shown in Figure 4.5. The transition probabilities from $C4$ to $C1$ or to $C2$ as well as that from $C2$ to $C4$ are all zero, which implies that the state cannot change directly from $C4$ to $C1$; it needs to move to $C3$ or $C2$ first and then from there to reach $C1$. More specifically, subject_2 needs more time to prepare to perform mental calculations. Thus, subject_2 exhibited a lower reaction time performance than subject_1. Due to the high resources used for mental calculations, it is relatively slow when the states change from $C1$ to $C4$ while subject_2 performs the lane-keeping task. This slow change results in many states of subject_2 appearing in the distractive driving state ($C3$); see Figure 4.9 (d). However, the average value of subject_2's RT is smaller than that of subject_1's RT. This is because turning the steering wheel is sensory decision-making; it uses fewer brain resources and can take action in response to car deviation. This phenomenon can be observed in the driving task of subject_3. Subject_3 prepares to correct the deviation of the car faster than other subjects; hence, the distractive driving state ($C3$) appears at an average switch time of 0.5904 sec, which is earlier than that of subject_2 and subject_1. This earlier switching of the driving state results in faster average reaction time in the driving task. Apart from the driving task, subject_3 performed mental calculations with a 1.53-sec average RT (ref. Figure 4.9 (e)); the states remained in $C2$ in many trials. This is suggestive of the fact that subject_3 did not allocate much brain resources for mental calculations and has a good capability to address driving and mental calculations simultaneously. In Figure 4.7, the transition diagram shows that the transition probabilities from $C1$ to $C4$ and $C4$ to $C1$ are both zero, which indicates that the two pure states, the mental calculation ($C1$) and driving task ($C4$), cannot directly switch to each other. However, the transition probability from $C4$ to $C2$ is 0.0432, higher than that of both subject_1's and subject_2's. This phenomenon implies that subject_3 can quickly allocate brain resources for mental calculations from driving tasks. Accordingly, we may infer that the load on the brain of subject_3's is not as heavy as that of subject_2 and subject_1 during the whole experiment. Consequently, subject_3 can switch the brain states faster than others.

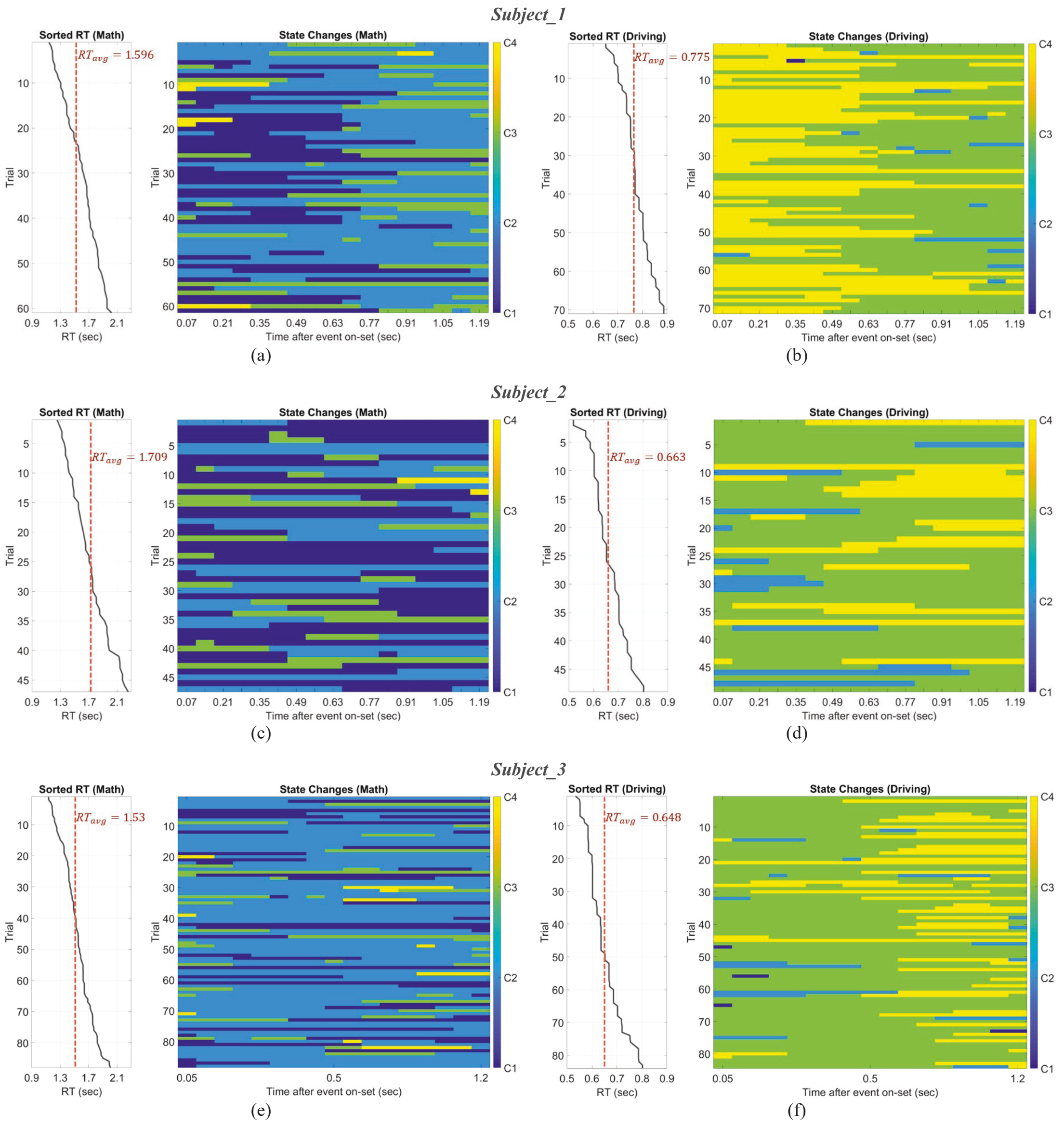


Figure 4.9: The RT and state changes of each subject during the mathematical problem-solving task and driving task. (a) and (b) present the states of subject_1, (c) and (d) are subject_2, and (e) and (f) are subject_3, while performing Math and Driving, respectively. Every trial contains 1.2-second-length data (17 segmentations) and is sorted based on the corresponding RT in ascending order. The red vertical line indicates the mean value of RT in the mathematical problem-solving task and driving task.

Table 4.8: The statistical data regarding each single state switching to another state.

Unit: sec	**Subject_1**		**Subject_2**		**Subject_3**	
Switching Time	**Average**	**STD**	**Average**	**STD**	**Average**	**STD**
$C1$	0.4228	0.2562	0.6258	0.3458	0.6476	0.3597
$C2$	0.7168	0.3297	0.6048	0.3451	0.6188	0.3402
$C3$	0.7672	0.3346	0.6118	0.3395	0.5904	0.3379
$C4$	0.4291	0.2919	0.7581	0.3164	0.8572	0.296

4.5 Summary

In the current study, we proposed FCOSTD which provides an effective mechanism for discovery and representation of the covert states and the transitions between states. We applied FCOSTD to one distracted driving experiment. The proposed system is capable of extracting useful features from the dynamic patterns of brain signals using an FNN. The FNN, leveraged by the fuzzy-inference mechanism, was successfully used for feature encoding from the EEG data. We adopted this approach because the inference mechanism of an FNN is capable of representing/modelling human brain activities associated with varying behaviours (RTs). These fuzzy-inference-based features are then clustered to identify external and covert states of human brain during the experiment involving distracted driving. The clustering results are used to interpret the states in terms of physical behaviours. Afterwards, the relationship between different states is expressed by the transition probability diagram. Because of the links between pairs of the states along with the link weights representing the conditional transition probabilities, brain dynamics can be easily visualized. We found that covert states do exist in the brain when the subject is responding to on-set events during the experiment. Moreover, the FCOSTD also provides a mechanism for describing state changes with their corresponding probabilities and the Markov chain. Through the Markov chain, the state changes can be easily observed, which allows us to understand how the brain resource was allocated as different types of actions were taken up by a subject.

Chapter 5

Distributed Neural Fuzzy System

5.1 Background

For some MAT systems applications, such as hospital service systems and bank networks [6], data privacy needs to be taken into consideration. In such MAT systems, the massive amount of data comes with a large number of agents and may be not available on a single controller, but distributed throughout a network of interconnected agents[93]. Unlimited data transition between agents may lead to serious security and privacy issues [94]–[96]. Therefore, a centralized algorithm that must be implemented with all the data in a centralized agent is neither practical nor safe especially in a MAT system considering data privacy. A distributed algorithm relied on the local data and limited communication among agents is necessary for the big data environment. In the proposed consensus learning, a single model must be agreed by all the agents based on some consensus protocols after the distributed learning process. Thus, the proposed consensus learning for Distributed Fuzzy Neural Network (D-FNN) is quite expected in the big data environment.

Recently, several distributed algorithms for FNN were proposed to decentralise the learning process [27], [28]. The authors in [27] proposed a decentralized algorithm for random-weights FNN, where the parameters in the antecedent layer are randomly selected instead of being estimated. An online implementation for the same FNN structure in [27] is further proposed in [28]. There's no doubt that such a random method for parameter identification can result in very large deviations during the learning process. In addition, it suffers from the curse of dimensionality as the number of fuzzy rules increases exponentially with the increase of input space. Moreover, the proposed distributed algorithms can only assure the consensus on the consequent layer instead of both antecedent and consequent layers of the FNN. In other words, such distributed algorithms are not really distributed since a single model can not be agreed by multiple agents.

To overcome the aforementioned issues in distributed algorithm for FNN, this research proposes a D-FNN model, which considers the consensus for its antecedent and consequent layers and hence is really distributed. A novel consensus learning algorithm, which consists of consensus-based structure learning and parameter learning, is proposed for the D-FNN model. Particularly, the consensus-based structure learning for the antecedent layer is built on a distributed clustering method, which can alleviate the dimensionality problem of the random-weights FNN. The consensus-based parameter learning for the consequent layer is realized by a distributed least square algorithm. The consensus-based structure learning and parameter learning are implemented sequentially with the latter employed after the former. Both of them are built on the well-known Alternating Direction Method of Multipliers (ADMM), which has been shown as an efficient solution for the consensus-based problems[26], [97]. Although many centralized algorithms for the FNN with clustering method for structure learning were studied in [98]–[100], to the authors' best knowledge, its distributed counterparts were not quite considered in the literature.

The contribution of this study is three-fold:

- This study proposes a new D-FNN model to address the inherent issues in the big data environment, including the uncertainty and distributed challenges. Note that existing D-FNN models always avoided the consensus for its antecedent layer due to computational difficulty. The proposed real D-FNN exploits distributed structure learning and parameter learning sequentially for the antecedent layer and consequent layer, respectively.

- A novel consensus learning algorithm is proposed to address the proposed D-FNN model. The consensus learning algorithm that consists of consensus-based structure learning and parameter learning is built on the well-known ADMM. It's worth noting that the consensus learning algorithm is very scalable and does not suffer from slow training speed or gradient vanishing problems compared with back-propagation-based methods.

- This study provides comprehensive simulations of various structures of the D-FNN. Simulation results of the proposed consensus learning algorithm outperform all existing D-FNN algorithms in terms of both generalization accuracy and training speed. Thus the superiority and effectiveness of the consensus learning algorithm are clear.

The rest of the paper is organized as follows. Section II is devoted to modelling of the structure learning and parameter learning for centralized FNN. Section III extends the centralized FNN to D-FNN, for which the consensus learning algorithm is proposed. A comprehensive simulation is conducted in Section IV to confirm the superiority and effectiveness of the proposed D-FNN model and consensus learning algorithm. Conclusions are drawn in Section V.

5.2 Structure learning and parameter learning for centralized FNN

Let us briefly recall the structure of existing FNN, which employs the first-order of the Takagi-Sugeno method of fuzzy inference system. Suppose estimating a scar output $y \in \mathbb{R}$ from a D-dimensional input $x = [x_1, x_2, \cdots, x_D]$, then the k-th fuzzy rule of the T-S system is

Rule k: IF x_1 is A_{k1} and $\cdots$ and x_D is A_{kd}
Then $y = w_{k0} + \sum_{j=1}^{D} w_{kj} x_j$

where A_{kj} is a Gaussian membership fuzzy set whose membership function is described by,

$$\varphi_{kj}(x_j) = \exp\left[-\left(\frac{x_j - m_{kj}}{\sigma_{kj}}\right)^2\right] \tag{5.1}$$

where m_{kj} and σ_{kj} are the mean and standard variance of the Gaussian membership function, respectively. Usually, the FNN is constituted by five feed-forward layers, whose structure is provided in Fig.5.1.

Layer 1 is the input layer, where each node corresponds to one input variable, i.e. x_d, and transmits the scaled input value to the next layer.

Layer 2 is the antecedent layer, where each node corresponds to one fuzzy set and outputs a membership value according to (5.1).

Layer 3 is the rule layer, where each node represents one fuzzy logic rule and performs antecedent matching of this rule using the following AND operation:

$$\phi_k(x) = \prod_{j=1}^{D} \varphi_{kj}(x_j), \tag{5.2}$$

which is the firing strength of fuzzy rule k. The obtained firing strength is then normalized by

$$\bar{\phi}_k(x) = \frac{\phi_k(x)}{\sum_{k=1}^{K} \phi_k(x)}. \tag{5.3}$$

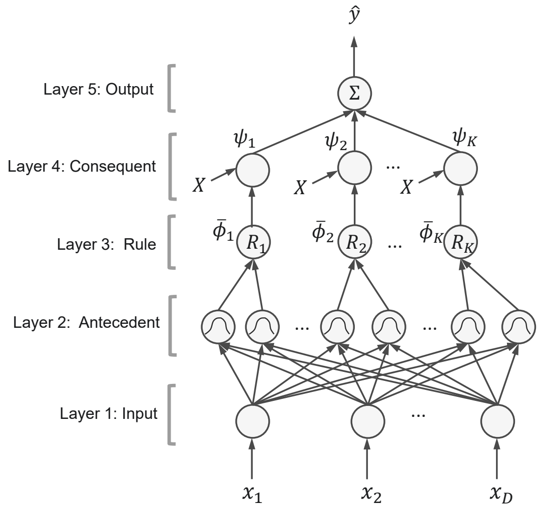


Figure 5.1: The structure of FNN

where K is the total number of fuzzy rules.

Layer 4 is called the consequent layer, where each node performs a defuzzification process for each fuzzy rule k using a weighted average operation as follows:

$$\psi_k(x) = \bar{\phi}_k(x)(w_{k0} + \sum_{j=1}^{D} w_{kj}x_j), \tag{5.4}$$

Layer 5 is the output layer, which calculates the overall output by summing the output of each fuzzy rules in Layer 4 as follows,

$$\hat{y} = \sum_{k=1}^{K} \psi_k(x), \tag{5.5}$$

Generally, FNN involves identification of structure and parameters. The structure learning is to identify the parameter of Gaussian membership function in the antecedent layer for each fuzzy rule k, i.e. m_{kj} and σ_{kj}, $j \in \{1, \cdots, D\}$; the parameter learning is to identify the output weights $w_{k0}, \cdots, w_{kD}$ in the consequent layer. Both of the structure learning and parameter learning can be addressed by the use of back propagation method [44]. Note that the back-propagation learning process often suffers from slow training speed and gradient vanishing issues. Thus, it is neither practical nor efficient especially in the big data environment, where the data possibly have very large samples and dimensions. To avoid the aforementioned issues, we considered employing the clustering method [98] for the structure learning and the least square algorithm [99] for the parameter learning.

The fuzzy rules are determined by the input-output pairs of the training samples. A basic idea is to group the input-output pairs into clusters and use one rule for each cluster. In this research, the K-means algorithm [101], which is one of the most popular and efficient clustering algorithm, is employed for structure learning to identify parameters in the antecedent layer of the FNN. As shown in Fig.5.2, the clustering centers obtained by the K-means algorithm will be the centers of Gaussian membership function in the antecedent layer of the FNN.

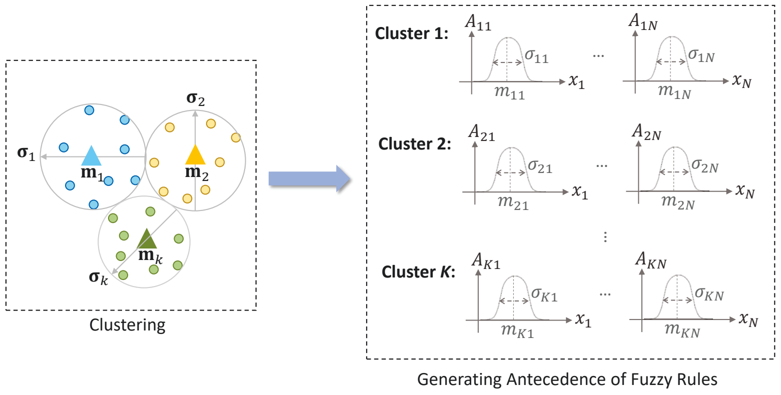


Figure 5.2: The clustering method to identify Gaussian parameters in the antecedent layer of the FNN

Let $\mathcal{D} := \{(X_i, Y_i)|X_i \in \mathbb{R}^D, Y_i \in \mathbb{R},\ i \in \mathcal{N} : \{1, \cdots, N\}\}$ denote the set of training data, x_{ij} denote the j-th component of the i-th observation X_i. The K-means algorithm aims at partitioning the N observations into K sets $\mathcal{C} := \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$. It should be noted that the total number of clusters K, which is assumed to be known a priori, is also the total number of fuzzy rules. Let $\mathcal{K} := \{1, \cdots, K\}$, the clustering problem amounts to identifying the centers of each cluster $k \in \mathcal{K}$ by minimizing the squared error distortion, i.e.,

$$\mathbf{m}_k = \arg\min_{\mathbf{m}_k} \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{|\mathcal{C}_k|} ||X_i - \mathbf{m}_k||^2 \tag{5.6}$$

where $\mathbf{m}_k$ is the center of cluster $\mathcal{C}_k$. The K-means algorithm uses an iterative refinement technique. Starting from an initial set of K centers at iteration $t = 0$, i.e. $\{\mathbf{m}_1(0), \cdots, \mathbf{m}_K(0)\}$, the algorithm alternates between an assignment step and an update step at iteration $t + 1$ as follows:

- Assignment step: Assign each observation X_i to the cluster $\mathcal{C}_k(t)$, whose center is closest to X_i.
- Update step: Update the center of each cluster by

$$\mathbf{m}_k(t+1) = \frac{1}{|\mathcal{C}_k(t)|} \sum_{X_i \in \mathcal{C}_k(t)} X_i.$$

The K-means algorithm converges if the centers are unchanged during the iteration. Once the centers of each cluster are obtained, the center of each fuzzy set is thus obtained. Accordingly, the standard variance σ_{kj} of each fuzzy set can be obtained by

$$\sigma_{kj} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - m_{kj})^2}. \tag{5.7}$$

By now, the parameters of Gaussian membership function in the antecedent layer are determined. In the following, a closed-form solution to identify the output weights in the consequent layer will be introduced.

Define a hidden matrix $H := [H_1, \cdots, H_K] \in \mathbb{R}^{N\times K(D+1)}$, where

$$H_k = \begin{bmatrix} \bar{\phi}_k(X_1) & \bar{\phi}_k(X_1)x_{11} & \cdots & \bar{\phi}_k(X_1)x_{1D} \\ \bar{\phi}_k(X_2) & \bar{\phi}_k(X_2)x_{21} & \cdots & \bar{\phi}_k(X_2)x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\phi}_k(X_N) & \bar{\phi}_k(X_N)x_{N1} & \cdots & \bar{\phi}_k(X_N)x_{ND}, \end{bmatrix} \tag{5.8}$$

and the output vector $Y := [Y_1, \cdots, Y_N]$. The output weight $\mathbf{w} \in \mathbb{R}^{K(D+1)}$ in the consequent layer of FNN can be learned by solving the following optimization problem,

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{2}||Y - H\mathbf{w}||^2 + \frac{\mu}{2}||\mathbf{w}||^2, \tag{5.9}$$

where $\mu > 0$ is a trade-off factor between the training error and regularization, the weight $\mathbf{w}$ has the following form:

$$\mathbf{w} = [w_{10}, \cdots, w_{1d}, \cdots, w_{K0}, \cdots, w_{KD}]^T, \tag{5.10}$$

It should be noted that selecting the value of μ appropriately can make the solution much more stable and have better generalization performance[102]. Since (5.9) is a standard least square optimization problem, its closed form solution can be easily obtained by:

$$\mathbf{w} = (H^T H + \mu I)^{-1} H^T Y, \tag{5.11}$$

where I is the identity matrix with dimension of $K(D+1)$.

5.3 Consensus learning for D-FNN

In the big data environment, large-scale data may exist in different locations and machines, referred as multiple agents in this research. Note that the centralized FNN must implement all data in a single agent. It may not be possible to perform centralized FNN in the big data environment due to several reasons. First, the communication load and storage resources of a single agent may limit its implementation of large-scale data. In addition, transiting the data between multiple agents may result in serious data security and privacy issues, which have attracted increasing public attention recently. Moreover, the whole system will fail if the centralized agent loses or disconnects due to contingency. Therefore, there is a great demand for D-FNN, where the global training process can be performed in each individual agent with limited information exchange. Moreover, the D-FNN is more flexible and robust against the contingencies compared with the centralized FNN.

In this section, the centralized FNN is extended to its distributed version D-FNN to deal with the big data. A novel consensus learning is proposed for the D-FNN, which integrates multiple FNNs corresponding to multiple agents in the big data environment and agrees on a single FNN based on consensus protocols. The consensus learning algorithm consists of consensus-based structure learning and parameter learning. Both of them are built on ADMM, which is widely employed in consensus-based distributed problems[26].

Generally, the distributed algorithms can be classified into two types based on network topology. The first one is implemented in a star network, where a fusion center is required to communicate with all agents. The second one does not need such fusion center. The agents are only allowed to communicate with their neighbouring agents based on the underlying network topology. Although the former has better convergence performance, the latter is more preferred in the big data environment due to the practicability and security issues. The fusion center may not exist in the big data environment. In addition, the requirement that all agents must communicate with the fusion center instead of with their neighbouring ones, will increase the potential risk of data leakage. Therefore, the D-FNN proposed in this research will employ the second type of distributed algorithm.

We consider a network with L nodes connected with E edges and each node is assumed as an agent. This network can be described as an undirected graph $\mathcal{G} = \{\mathcal{L}, \xi\}$, where $\mathcal{L}$ and ξ are the sets of vertexes and edges, respectively. A simple network consisting of five agents and six edges is shown in Fig.5.3 for illustration purpose. For the dataset $\mathcal{D} := \{(X_i, Y_i)|i \in \mathcal{N}\}$, Let $\{\mathcal{D}^1, \cdots, \mathcal{D}^L\}$ be a decomposition of the entire dataset $\mathcal{D}$. For ease of representation, each subset $\mathcal{D}^l$ is assumed to be the data located on each agent $l \in \mathcal{L}$. The set of neighbouring agents of agent l is defined as $\mathcal{N}_l$. Similarly, we introduce subset $\mathcal{C}^l$ to represent the observation subset of agent l. In each subset $\mathcal{C}^l$, let $\mathcal{C}^l_k$ denote the subset of cluster k in agent l such that $\bigcup_{k=1}^{K} \mathcal{C}^l_k = \mathcal{C}^l$. Based on the consensus strategy, the structure learning problem

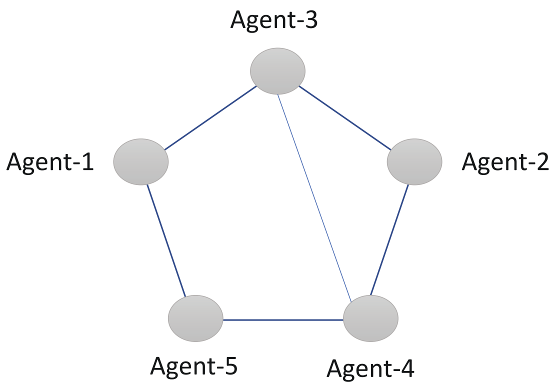


Figure 5.3: A simple network with five agents

for the D-FNN can be formulated as follows,

$$\min_{\mathbf{m}^l_k} \frac{1}{2} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{X^l_i \in \mathcal{C}^l_k} ||X^l_i - \mathbf{m}^l_k||^2 \tag{5.12a}$$

$$\text{s.t.} \quad \mathbf{m}^l_k = \mathbf{r}_k, \; l \in \mathcal{L}, \; k \in \mathcal{K}, \tag{5.12b}$$

where $\mathbf{m}^l_k$ is a local variable, representing the center of fuzzy set k of agent l, $\mathbf{r}_k$ is a global variable to integrate all local centers and $|\mathcal{C}^l_k|$ represents the cardinality operation for a set. The constraint (5.12b) employs the consensus strategy, which assures all the local centers coincide at one global vector of center. It is worth noting that the local variable $\mathbf{m}^l_k$ can be computed in parallel for each agent l.

Once the global center $\mathbf{r}_k$ of each fuzzy rule k is determined, the global standard variance can be easily calculated by:

$$\bar{\sigma}_{kj} = \sqrt{\frac{1}{N} \sum_{l=1}^{L} |\mathcal{C}^l| (\sigma^l_{kj})^2} \tag{5.13}$$

where $\bar{\sigma}_{kj}$ is the j-th component of the global standard variance of the k-th fuzzy rule, σ^l_{kj} is the corresponding standard variance of local agent l, $|\mathcal{C}^l|$ represents the cardinality of subset $\mathcal{C}^l$. It should be noted that the standard variance σ^l_{kj} of rule k dimension j is calculated in a component-wise manner, i.e.

$$\sigma^l_{kj} = \sqrt{\frac{1}{|\mathcal{C}^l_k|} \sum_{X^l_i \in \mathcal{C}^l_k} (X^l_{ij} - \mathbf{m}^l_{kj})^2} \tag{5.14}$$

where X^l_{ij} and $\mathbf{m}^l_{kj}$ are the j-th component of X^l_i and $\mathbf{m}^l_k$, respectively.

The above distributed clustering by consensus learning for antecedent layer identification of the D-FNN is shown in Fig.5.4

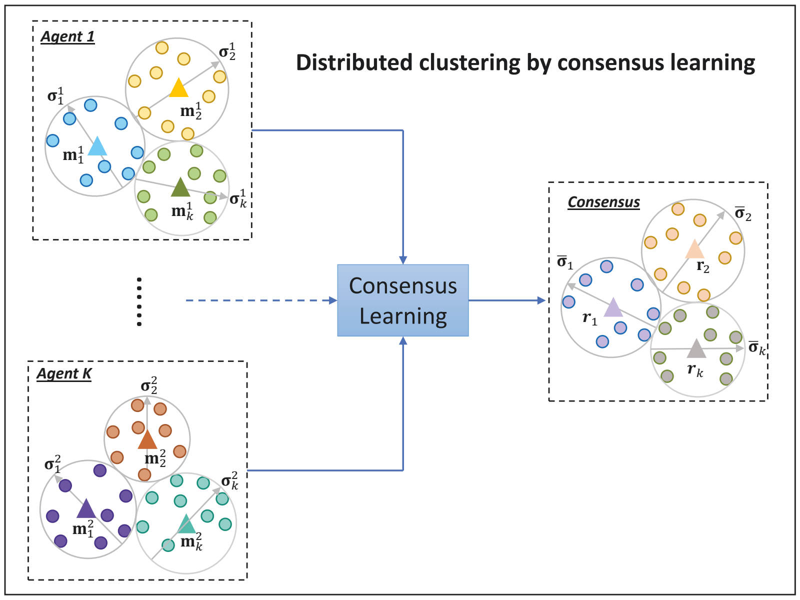


Figure 5.4: The procedure of distributed clustering by consensus learning for antecedent layer identification of the D-FNN

Similarly, the parameter learning problem for the D-FNN based on the consensus strategy is formulated as follows,

$$\min_{\mathbf{w}^l} \frac{1}{2} \sum_{l=1}^{L} (||Y^l - H\mathbf{w}^l||^2 + \mu||\mathbf{w}^l||^2), \tag{5.15a}$$

$$\text{s.t.} \quad \mathbf{w}^l = \mathbf{z}, \; l \in \mathcal{L}, \; q \in \mathcal{N}_l, \tag{5.15b}$$

where $\mathbf{w}^l$ is a local variable, representing the output weight of agent l, $\mathbf{z}$ is a common global weight to integrate all local weights.

The procedure of the proposed consensus learning algorithm, which consists of distributed structure learning for the antecedent layer and distributed parameter learning for the consequent layer, is described in Fig.5.5.

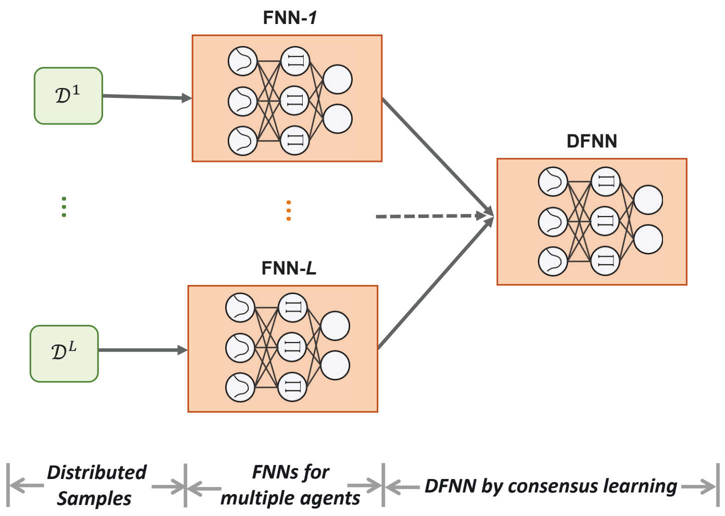


Figure 5.5: The procedure of consensus learning for the D-FNN

5.3.1 Overview of ADMM

In this section, we first introduce the preliminary knowledge of ADMM[26], which will be used to solve the proposed optimization problem (5.12) and (5.15) for the consensus learning. The standard form of ADMM solves the following problem:

$$\min f(\mathbf{x}) + g(\mathbf{y}) \tag{5.16}$$

$$\text{s.t.} \quad A\mathbf{x} + B\mathbf{y} = c, \ \mathbf{x} \in \mathcal{C}_f, \ \mathbf{y} \in \mathcal{C}_g \tag{5.17}$$

Then the following augmented Lagrangian is constructed,

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = \quad & f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^T (A\mathbf{x} + B\mathbf{y} - c) \\ & + \frac{\rho}{2} ||A\mathbf{x} + B\mathbf{y} - c||^2 \end{aligned} \tag{5.18}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier and ρ is a positive constant to trade-off the convergence rate and numerical accuracy. The ADMM solves (5.18) iteratively via an alternating procedure, which starts from arbitrary initial points $\mathbf{y}(0)$ and $\boldsymbol{\lambda}(0)$

and then updates the variable $\mathbf{x}$ and $\mathbf{y}$ and multipliers $\boldsymbol{\lambda}$ sequentially by the following procedure:

$$\mathbf{x}(t+1) = \arg\min_{\mathbf{x}\in\mathcal{C}_f} L(\mathbf{x}, \mathbf{y}(t), \boldsymbol{\lambda}(t)), \tag{5.19}$$

$$\mathbf{y}(t+1) = \arg\min_{\mathbf{y}\in\mathcal{C}_g} L(\mathbf{x}(t+1), \mathbf{y}, \boldsymbol{\lambda}(t)), \tag{5.20}$$

$$\boldsymbol{\lambda}(t+1) = \boldsymbol{\lambda}(t) + \rho(A\mathbf{x}(t+1) + B\mathbf{y}(t+1) - c), \tag{5.21}$$

until convergence. The convergence behavior at iteration t can be inspected by analyzing the primal residual and dual residual with given tolerances as follows,

$$||A\mathbf{x}(t) + B\mathbf{y}(t) - c||^2 \leq \epsilon_1, \tag{5.22}$$

$$||\boldsymbol{\lambda}(t) - \boldsymbol{\lambda}(t-1)||^2 \leq \epsilon_2, \tag{5.23}$$

where ϵ_1 and ϵ_2 are the values of convergence tolerance of the ADMM procedure.

Under the convexity condition of function $f(\cdot)$ and $g(\cdot)$ and domain set $\mathcal{C}_f$ and $\mathcal{C}_g$, ADMM is known to converge to a unique stable point[26]. A recent study also proves the convergence of ADMM for a variety of nonconvex and possibly nonsmooth functions given some sufficient conditions[97].

5.3.2 Consensus-based distributed structure learning

The optimization problem (5.12) is nonconvex, which can not be efficiently solved by any exhaustive search methods. Meanwhile, the centralized clustering methods such as the K-means algorithm still suffer from the issues of communication load, privacy concerns and contingencies in the big data environment. Thus, a distributed clustering method is crucial to address the optimization problem (5.12). In this study, a distributed K-means algorithm is proposed to address the structure learning optimization problem (5.12). This method is motivated by [103], where a distributed clustering scheme is developed for wireless sensor networks.

First, we construct the following augmented Lagrangian for (5.12):

$$\begin{aligned}\mathcal{L}_s(\mathbf{m}, \mathbf{r}, \boldsymbol{\lambda}_s) = \quad & \frac{1}{2}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{X_i^l\in\mathcal{C}_k^l} ||X_i^l - \mathbf{m}_k^l||^2 \\ & + \sum_{l=1}^{L}\sum_{k=1}^{K} \boldsymbol{\lambda}_{s,kl}^T(\mathbf{m}_k^l - \mathbf{r}_k) \\ & + \frac{1}{2}\rho_s \sum_{l=1}^{L}\sum_{k=1}^{K} ||\mathbf{m}_k^l - \mathbf{r}_k||^2 \end{aligned} \tag{5.24}$$

where $\boldsymbol{\lambda}_{s,kl}$ is the Lagrange multiplier and $\rho_s > 0$ is a penalty parameter. Starting from the initial points $\mathbf{r}(0)$ and $\boldsymbol{\lambda}_s(0)$, the variables at each iteration t are updated iteratively by the following procedures based on the ADMM:

$$\mathbf{m}^l(t+1) = \arg\min_{\mathbf{m}} \mathcal{L}(\mathbf{m}^l, \mathbf{r}(t), \boldsymbol{\lambda}_s(t)), \tag{5.25}$$

$$\mathbf{r}(t+1) = \arg\min_{\mathbf{r}} \mathcal{L}(\mathbf{m}^l(t+1), \mathbf{r}, \boldsymbol{\lambda}_s(t)), \tag{5.26}$$

$$\boldsymbol{\lambda}_{s,kl}(t+1) = \boldsymbol{\lambda}_{s,kl}(t) + \rho_s(\mathbf{m}_k^l(t+1) - \mathbf{r}_k(t+1)). \tag{5.27}$$

It should be noted that (5.25) can be solved in parallel for each agent l. Similarly, we can also employ the assignment step and update step in the K-means algorithm to update the cluster centers $\mathbf{m}^l(t+1)$ for each agent l. On the other hand, since the optimization problem (5.26) is linear on $\mathbf{r}$, the closed-form solution can be easily solved by setting the partial derivatives with respect to $\mathbf{r}$ to zero. The closed-form solution of (5.26) is given directly as follows,

$$\mathbf{r}_k(t+1) = \frac{1}{\rho_s}\bar{\boldsymbol{\lambda}}_{s,kl}(t) + \bar{\mathbf{m}}_k^l(t+1), \tag{5.28}$$

where

$$\bar{\mathbf{m}}_k^l(t+1) = \frac{1}{L}\sum_{l=1}^{L} \mathbf{m}_k^l(t+1), \tag{5.29}$$

$$\bar{\boldsymbol{\lambda}}_{s,kl}(t) = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{\lambda}_{s,kl}(t). \tag{5.30}$$

From (5.28), we can see updating $\mathbf{r}_k(t+1)$ requires the communication between all the agents. However, it may not possible in the big data environment. Therefore, a more practical updating procedure is more preferred here. Similarly with the idea in [104], this study employs the well-known distributed average consensus (DAC) strategy [105] to update $\mathbf{r}_k(t+1)$. The DAC is an iterative strategy to compute the global average requiring data only exchanged in a local neighborhood. At iteration t, the local DAC update is given by:

$$\alpha^l(t+1) = \sum_{q\in\mathcal{L}_l} W_{lq}\alpha^l(t), \tag{5.31}$$

where α^l is the local variable corresponding to each agent l, W_{lq} is a weighted connectivity matrix. Suppose d_l is the number of neighboring agents of agent l and

$d = \max_{l \in \mathcal{L}} d_l$. According to [105], given the following matrix-degree weight,

$$W_{lq} = \begin{cases} \dfrac{1}{d+1}, & \text{if} \quad q \in \mathcal{L}_l; \\ 1 - \dfrac{d_l}{d+1}, & \text{if} \quad q = l; \\ 0, & \text{otherwise.} \end{cases} \tag{5.32}$$

the following procedure converges to the global average:

$$\lim_{t \to +\infty} \alpha^l(t) = \frac{1}{L} \sum_{l=1}^{L} \alpha^l(0), \ \forall l \in \mathcal{L}. \tag{5.33}$$

Therefore, $\bar{\mathbf{m}}_k^l(t+1)$ and $\bar{\boldsymbol{\lambda}}_{s,kl}(t)$ can be easily obtained by using the DAC iteration (5.31) with the matrix-degree weight (5.32). It is worth noting that it is not necessary to match the cluster ordering of each agent before the consensus procedure. The consensus algorithm will still converge with random initial cluster ordering.

Based on the analysis above, we can summarize the algorithm for distributed structure learning in Algorithm 3. The convergence behavior of Algorithm 3 can be inspected by checking the norms of the following two residuals:

$$||\mathbf{m}_k^l(t) - \mathbf{m}_k^q(t)||^2 \le \epsilon_1, \tag{5.34}$$

$$||\boldsymbol{\lambda}_{s,k}^l(t) - \boldsymbol{\lambda}_{s,k}^l(t-1)||^2 \le \epsilon_2. \tag{5.35}$$

Computational complexity: The computation complexity of Algorithm 1 relies on the local variables update and global variables update. The local variables update is similar to the one-step of centralized K-means algorithm. Thus its computational complexity is $\mathcal{O}(NKD)$. The global variable update is based on the distributed average consensus method, the computational complexity of which is $\mathcal{O}(KDLT_2)$, where T_3 is iterations required for the convergence of the distributed average consensus method. Generally, T_3 is within a few tens. Suppose Algorithm 1 requires T_1 iterations for the update of local and global variables. The total computation complexity for Algorithm 1 is $\mathcal{O}((NKD + KDLT_3)T_1)$. Since $LT_3 \ll N$, we can conclude the computation complexity for Algorithm 1 is $\mathcal{O}(NKDT_1)$. Note that the distributed K-means algorithm is built on the well-known ADMM algorithm, which generally converges to modest accuracy (such as $\epsilon < 10^{-3}$ set in our paper) within a few tens of iterations. Therefore, the total memory burden of Algorithm 1 is quite limited.

Remark 1: In Algorithm 1, an intuitive issue is that how to properly match the clusters between the agents as each agent has many clusters. There are generally

Algorithm 3 ADMM-based distributed structure learning (5.12)

```
Initialization: Set $t = 0$ and the Lagrange multipliers $\boldsymbol{\lambda}_{s,kl}(t) = \mathbf{0}$. Initialize the
cluster centers $\mathbf{m}_k^l(t)$ by using K-means algorithm for each agent $l$.
for $t = 0, 1, 2, \cdots$, do
    Update the local variables $\mathbf{m}^l(t+1)$:
    Assignment step: Each agent $l$ assigns each $X_i^l$ to the cluster $\mathcal{C}_k^l(t)$, whose center
    $\mathbf{r}_k^l(t)$ is closest to $X_i^l$.
    Update step: Each agent $l$ updates the center of each cluster $\mathcal{C}_k^l(t)$ by
        $$\mathbf{m}_k^l(t+1) = \frac{1}{|\mathcal{C}_k^l(t)|} \sum_{X_i^l \in \mathcal{C}_k^l(t)} X_i^l \qquad (5.36)$$
    Update the global variables $\mathbf{r}(t+1)$ by (5.28) and broadcast it to each agent
    $l$.
    Update the dual variables $\boldsymbol{\lambda}_s(t+1)$ by (5.27) and broadcast it to each agent
    $l$
end for
```

two simple ways to match the clusters. The first one is randomly matching the clusters. The second one is matching the clusters according to their Euclidean distance. Both the two clustering matching methods were tested and they lead to a consistent NRMSE value for the FNN. Let us briefly analyze this phenomenon. We can see that starting from any initial cluster centers, the centralized K-means algorithm can still converge. Its convergence takes the merits of the Assignment step and Update step. Though different initial points may lead to different clustering performance, seeking a better initialization for the K-means algorithm is not the main work in this study. In the distributed K-means algorithm, the initial cluster match will only affect the initial update of the global variables, which are just starting points for local agents to perform the local update in the next iteration. Note that during the local update, each agent will assign data samples again and update the local centers based on the assignment. This procedure will eliminate the impacts of an improper match. Although different initial points may lead to different clustering results, the general convergence of the K-means algorithm still holds regardless to initial points. Therefore, the distributed K-means algorithm can still converge benefiting from the Assignment step and Update step during the update of local variables. It's also worth noting that the distributed K-means algorithm is built on the ADMM algorithm and K-means algorithm, and these two algorithms do not rely on initial points for general convergence. A similar distributed clustering method with random cluster match was also employed in [32]. Thus, we believe the initial cluster match will not be an issue for the distributed K-means algorithm.

Remark 2: The number of cluster K is an important hyperparameter for the K-means-based algorithm. In the centralized K-means algorithm, K can be tuned using a cross-validation procedure. In the distributed case, each agent can not access others' original data. Therefore, the cross-validation procedure for centralized K-means is not suitable anymore. Here, we use the following procedure to tune the hyperparameter. For each agent l, we implement the centralized FNN with its own data for $K = 1 : K_{max}$ and obtain K_{max} NRMSE values. Then we select the index K_l, whose corresponding NRMSE is minimal, as the hyperparameter for agent l. If the hyperparameter K_l of each agent are the same, then Algorithm 1 can be used directly. Otherwise, we apply a zero-filling procedure to ensure the consistency for the number of clusters. Without loss of generality, we set $K_1 \leq K_2 \leq K_L$. As shown in Fig. 5.6, if a cluster is missing, the corresponding position is filled with zero-vector $\mathbf{0}$. After the filling procedure, the cluster number of each agent becomes the same. Then the Algorithm 1 can be used directly for the distributed structure learning of the D-FNN. As we discussed in **Remark 1**, the initial cluster match does not affect the convergence of the distributed K-means algorithm, thus we can use the above zero-filling procedure to handle the cluster mismatch issue among the agents. If the data distribution of each agent are totally different, then the knowledge of domain adaptation [106] should be considered. However, this goes beyond the scope of this research.

5.3.3 Consensus-based distributed parameter learning

Similarly, we solve the optimization problem (5.15) in a distributed manner by the use of ADMM. The augmented Lagrangian for (5.15) is as follow,

$$\begin{aligned}\mathcal{L}(\mathbf{w}, \mathbf{z}, \boldsymbol{\lambda}_p) = \quad & \frac{1}{2}\sum_{l=1}^{L} ||Y^l - H^l \mathbf{w}^l||^2 + \frac{\mu}{2}||\mathbf{z}||^2 \\ & + \sum_{l=1}^{L} \boldsymbol{\lambda}_{p,l}^T (\mathbf{w}^l - \mathbf{z}) \\ & + \frac{1}{2}\rho_p \sum_{l=1}^{L} ||\mathbf{w}^l - \mathbf{z}||^2 \qquad (5.37)\end{aligned}$$

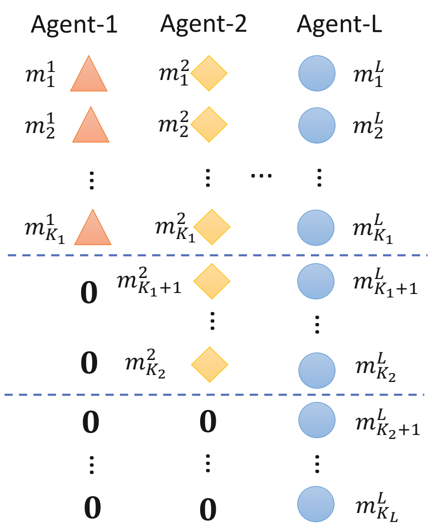


Figure 5.6: The cluster centers of each agent in the mismatching scenario

where $\boldsymbol{\lambda}_{p,l}$ is the Lagrange multiplier and $\rho_p > 0$ is a penalty parameter. The ADMM-based procedures for distributed parameter learning are as follows,

$$\mathbf{w}^l(t+1) = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}^l, \mathbf{z}(t), \boldsymbol{\lambda}_p(t)), \tag{5.38}$$

$$\mathbf{z}(t+1) = \arg\min_{\mathbf{z}} \mathcal{L}(\mathbf{w}^l(t+1), \mathbf{z}, \boldsymbol{\lambda}_p(t)), \tag{5.39}$$

$$\boldsymbol{\lambda}_p(t+1) = \boldsymbol{\lambda}_p(t) + \rho_s(\mathbf{w}^l(t+1) - \mathbf{z}(t+1)). \tag{5.40}$$

Since (5.38) is a standard least square problem, its closed form solution can be easily obtained by:

$$\mathbf{w}^l(t+1) = ((H^l)^T H^l + \mu I)^{-1}((H^l)^T Y^l - \boldsymbol{\lambda}_p(t) + \rho_p \mathbf{z}(t)), \tag{5.41}$$

where I is the identity matrix with dimension of $K(D+1)$. The closed-form solution of (5.39) can be obtained by:

$$\mathbf{z}(t+1) = \frac{\bar{\boldsymbol{\lambda}}_{p,l}(t) + \rho_p \bar{\mathbf{w}}^l(t+1)}{\mu/L + \rho_p}, \tag{5.42}$$

where

$$\bar{\mathbf{w}}^l(t+1) = \frac{1}{L}\sum_{l=1}^{L} \mathbf{w}^l(t+1),$$

$$\bar{\boldsymbol{\lambda}}_{p,l}(t) = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{\lambda}_{p,l}(t).$$

Similarly with the procedure in the distributed structure learning, we also employ the DAC strategy in order to avoid communications among all agents. The $\bar{\mathbf{w}}^l(t+1)$ and $\bar{\boldsymbol{\lambda}}_{p,l}(t)$ are obtained using the DAC iteration (5.31) with the matrix-degree weight (5.32).

Similarly, we summarize the algorithm for distributed parameter learning in Algorithm 4. The convergence behavior of the Algorithm 4 can be inspected by checking the norms of the following two residuals:

$$||\mathbf{w}^l(t) - \mathbf{z}(t)||^2 \leq \epsilon_1, \tag{5.43}$$

$$||\boldsymbol{\lambda}_p(t) - \boldsymbol{\lambda}_p(t-1)||^2 \leq \epsilon_2. \tag{5.44}$$

Algorithm 4 ADMM-based distributed parameter learning

```
**Initialization:** Set $t = 0$ and initialize global weight $\mathbf{z}(t)$ and Lagrange multipliers $\boldsymbol{\lambda}_p(t)$ for each agent.
**for** $t = 0, 1, 2, \cdots,$ **do**
    **Update the local variables** $\mathbf{w}^l(t+1)$ by (5.41)
    **Update the global variables** $\mathbf{r}(t+1)$ by (5.42) and broadcast it to each agent $l$.
    **Update the dual variables** $\boldsymbol{\lambda}_p(t+1)$ by (5.40) and broadcast it to each agent $l$
**end for**
```

Computational complexity: The computation complexity for Algorithm 2 also relies on the local variables update and global variables update. One may think that the update of local variables by (5.41) is costly due to the matrix inverse operation on a large dimension matrix. However, the dimension of $(H^l)^T H^l$ is actually $K(D+1) \times K(D+1)$, which is quite smaller compared with the dimension of H^l as $N_l \times K(D+1)$. In addition, the value of $((H^l)^T H^l + \mu I)^{-1}$ is constant during the iterations. It can be stored after being computed at the initial step. If we ignore the cost at the initialization, we can conclude the computation complexity for Algorithm 2 is $\mathcal{O}(KDLT_4T_2)$, where T_4 and T_2 are the iterations required for the distributed average consensus method and the ADMM algorithm, respectively. As we analyzed before, both T_4 and T_2 are within a few tens. Therefore, the total memory burden of Algorithm 2 is small.

Table 5.1: Numerical information of the tested datasets

Dataset	Samples	Dimensions	Desired Output
CCPP[107]	9,568	4	Electrical energy
KC-house[108]	21,613	15	House price
CASP[109]	45,730	9	Deviation
Motor[110]	998,070	7	Motor temperature

5.4 Simulations

In this section, the performance of the proposed consensus learning algorithm for D-FNN in the big data environment is evaluated by numerical simulations on several widespread datasets, which are available on UCI Machine Learning Repository [1] or Kaggle Datasets [2]. The simulation results based on the proposed consensus learning algorithm are compared with several state-of-the-art FNN algorithms.

The datasets are selected to represent various domains in the big data environment though not all of them have very large-scale samples and dimensions. Here we briefly summarize the input and output information for these datasets in Table 5.1. In all these cases, input variables are normalized between $[-1, 1]$ before the experiments.

To stimulate the distributed nature of the big data, a network of agents is randomly generated using a random topology model[104] with each agent's connectivity probability as 25%. Accordingly, each dataset is randomly decomposed for each agent. To evaluate the accuracy of all the models, we perform a 10-fold cross-validation for each dataset. In each fold, the following FNN algorithms are compared:

- Random-weight-FNN (R-FNN): This is the distributed FNN algorithm reported in [27], which randomly generates the Gaussian parameters in the antecedent layer and then employs the least square algorithm to identify the parameters in the consequent layer. As we mentioned in the Introduction, such a random method for parameter identification could result in very large deviations during the learning process. In addition, it suffered from the curse of dimensionality as the number of fuzzy rules increases exponentially with the increase of input space. Moreover, the distributed algorithm assured consensus only for the consequent layer instead of both the antecedent and consequent layers. Thus it is not really distributed and thus not practical in the big data environment.

[1]http://archive.ics.uci.edu/ml
[2]https://www.kaggle.com/datasets

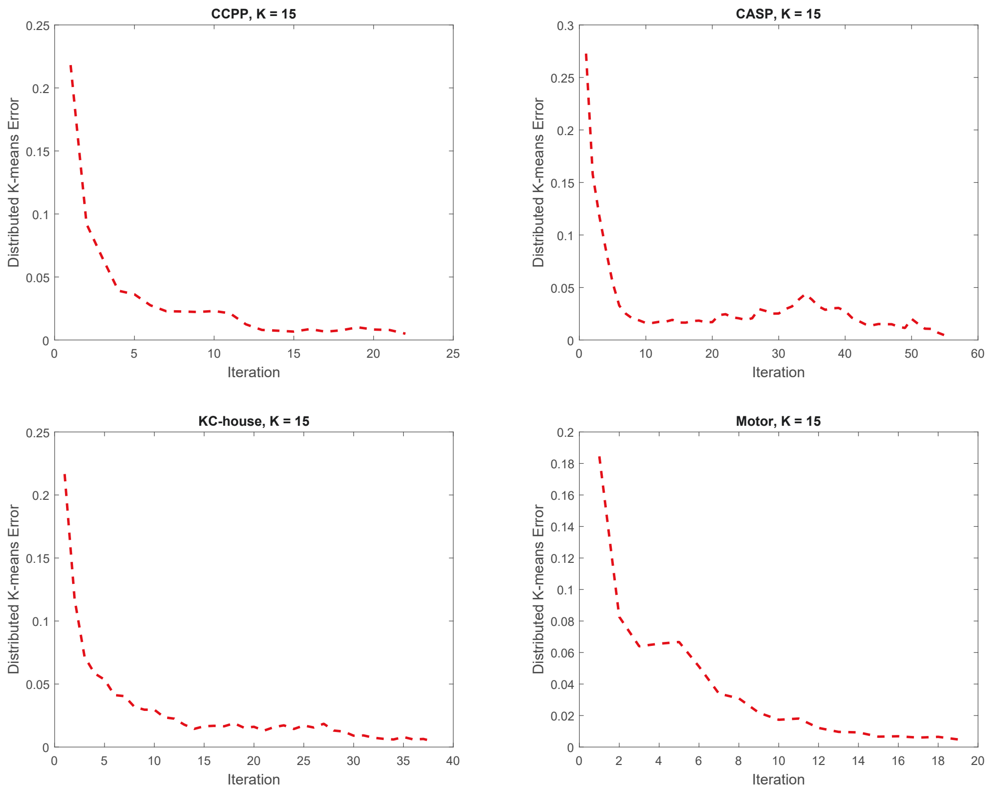


Figure 5.7: Convergence bahavior of the distributed K-means algorithm for various datasets

- Centralized K-means-FNN (C-FNN): This is the centralized FNN algorithm provided in this research. It employs the K-means algorithm to identify the Gaussian parameters in the antecedent layer and least square algorithm to identify the parameters in the consequent layer. Specifically, C-FNN solves the optimization problem (5.6) by the centralized K-means algorithm to obtain the parameters $\mathbf{m}_k$ and σ_{kj} and then fix them to solve the optimization problem (5.9) by the close-form solution (5.11). Note that by the use of the K-means algorithm for identifying parameters in the antecedent layer, it does not suffer from the curse of dimensionality as R-FNN does.

- Half-consensus learning D-FNN (H-FNN): This is the distributed algorithm for the same structure of C-FNN but employing the consensus protocol only for its consequent layer. Thus, the term ”half-consensus learning” is used for the H-FNN, Note that by H-FNN, agents can not agree on a single FNN model after the learning procedure. Specifically, H-FNN solves the optimization problem (5.6) by the centralized K-means algorithm to obtain the parameters $\mathbf{m}_k$ and σ_{kj}. Then H-FNN broadcasts them to each agent and solves optimization problem (5.15) in a distributed manner by Algorithm 4.

- Consensus learning D-FNN (D-FNN): This is the consensus learning algorithm proposed by this research, which employs the consensus protocol for both the antecedent and consequent layers. It's worth noting that this algorithm is the really distributed and practical one in the big data environment. The agreement among various agents on a single FNN model can be obtained after the consensus learning procedure. Particularly, D-FNN employs Algorithm 3 and Algorithm 4 sequentially to respectively solve the optimization problem (5.12) and optimization problem (5.15) in a fully distributed manner.

In this research, all simulations are implemented using Matlab R2019b on a laptop with Intel i7 @ 4.0 GHz processor and 16 GB of memory. The convergence criteria of the ADMM procedure in Algorithm 3 and Algorithm 4 are set as $\epsilon_1 = \epsilon_2 = 10^{-3}$. The normalized root mean square error (NRMSE) defined by

$$\text{NRMSE} = \sqrt{\frac{1}{N\hat{\sigma}_Y^2}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2}, \tag{5.45}$$

is used to evaluate the performance of the models.

The convergence behavior of distributed K-means algorithm and distributed parameter learning are provide by Fig.5.7 and Fig.5.8, respectively. It can be seen, both of the two algorithms converge quite fast for these datasets.

Table 5.2 summarizes the simulation results for the tested datasets by implementing R-FNN, C-FNN, H-FNN and D-FNN, respectively. The first column of Table 5.2 is the dataset name, the second column gives the total number of agents, the third column provides the trade-off factor in (5.15a). The fourth, fifth and sixth column of Table 5.2 present the simulation results of R-FNN including total number of rules, obtained NRMSE value and training time, respectively. The seventh column of Table 5.2 gives the K value, which is also the total number of clustering and fuzzy rules for implementing the C-FNN, H-FNN and D-FNN. Their simulation results are provided in the remaining columns. It should be noted that the NRMSE value obtained by the C-FNN is a lower bound of the one obtained by H-FNN. It can be seen from Table 5.2, R-FNN can not handle the datasets KC-house and Motor due to the large-scale rule numbers and samples. The NRMSE value of CCPP obtained by R-FNN is much worse than the ones obtained by C-FNN, H-FNN and D-FNN. As for CASP,

Table 5.2: Convergence behaviour of the distributed parameter learning algorithm for various datasets

Dataset	L	u	R-FNN		
			Rules	NRMSE	Time (sec)
CCPP	5	0.01	16	4.178	0.23
KC-house	5	0.01	32768	-	-
CASP	5	0.001	512	0.7682	457.4
Motor	25	0.001	128	-	-

Dataset	K	H-FNN	
		NRMSE	Time (sec)
CCPP	15	0.2313	0.15
KC-house	15	0.5299	0.45
CASP	15	0.7745	0.94
Motor	15	0.623	13.31

Dataset	K	H-FNN	
		NRMSE	Time (sec)
CCPP	15	0.232	0.26
KC-house	15	0.5304	0.51
CASP	15	0.7748	0.83
Motor	15	0.6248	6.0

Dataset	K	D-FNN	
		NRMSE	Time (sec)
CCPP	15	0.2331	0.61
KC-house	15	0.5251	2.31
CASP	15	0.7827	1.43
Motor	15	0.6079	9.82

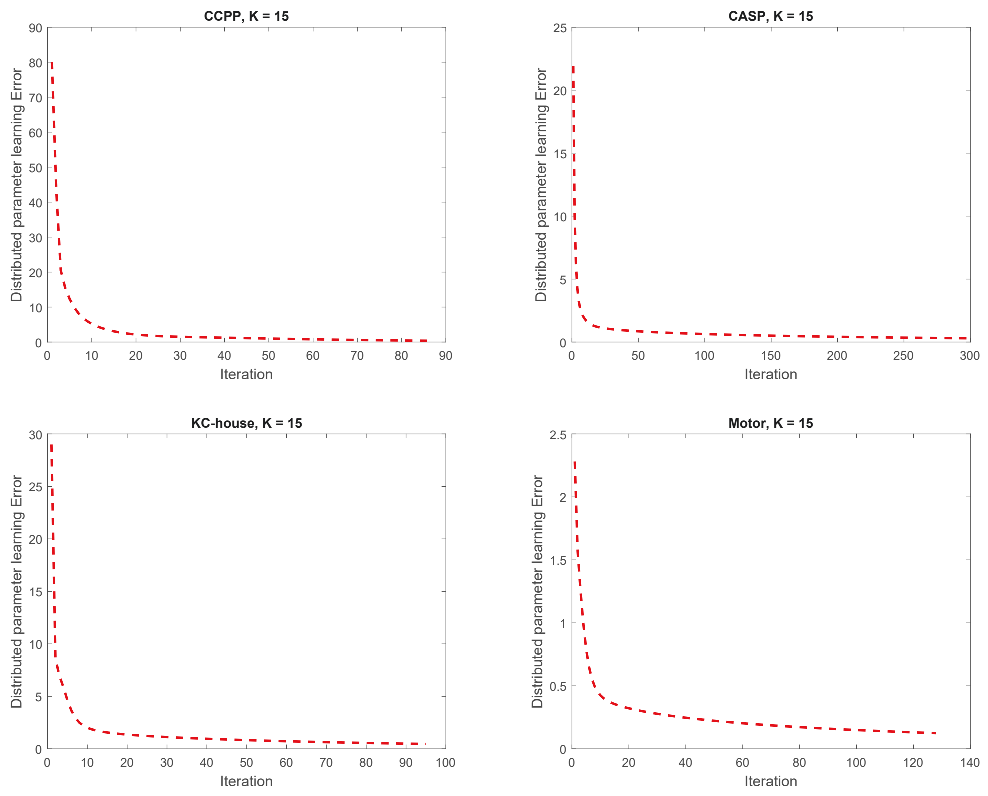


Figure 5.8: Convergence bahavior of the distributed parameter learning algorithm for various datasets

the NRMSE value obtained by R-FNN is a bit better than other three algorithms since R-FNN uses much more rules (512 vs 15). Clearly, R-FNN is not scalable and can not be used in the big data environment. We also test the D-FNN by using various values of K for CASP. Fig. 5.9 provides the NRMSE of CASP by setting various K for the D-FNN. Generally, larger value of K will lead to smaller value of NRMSE. However, we still suggest to select a moderate value of K. Surprisingly, the NRMSE value of Motor by D-FNN is much better than the other algorithms since the clustering results (fuzzy rules) of D-FNN is different from the others'. For such a large-scale dataset, the distributed clustering results can outperform the centralized ones. This phenomenon also verifies the effectiveness of the proposed consensus learning algorithm. In addition, we would like to re-emphasize that the superiority of D-FNN compared with other three methods are as follows: 1) The D-FNN is able to handle data in multiple agents. This capability becomes more significant in big

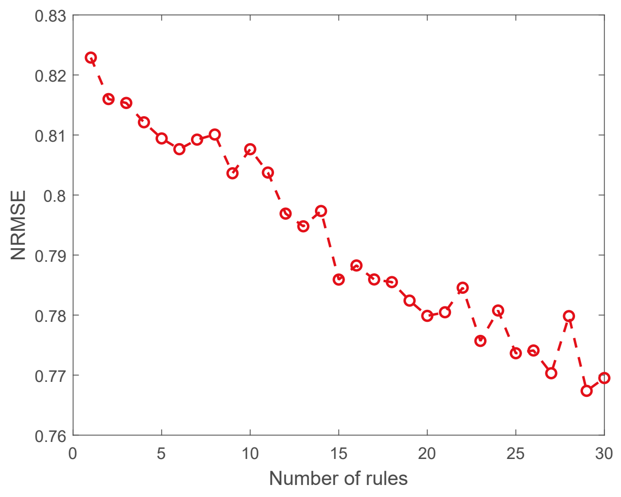


Figure 5.9: NRMSE of CASP by setting various K

data environment as the big data may exist in different locations and machines. 2) The D-FNN can alleviate the burden of communication load and storage resources in a single agent. 3) The D-FNN can preserve the users' data privacy by limiting the data transiting between multiple agents.

To verify the proposed distributed K-means method can work well in the case that each agent has different cluster numbers, we implemented it on the CASP dataset with different sample number and different cluster number of each agent, specifically, the cluster number are respectively $K_1 = 11, K_2 = 12, K_3 = 13, K_4 = 14, K_5 = 15$. The distributed K-means method converge within 50 iterations and achieves the NRMSE value 0.7887, which is consistent with the NRMSE value 0.7827 obtained by the distributed K-means method but with the same cluster number $K = 15$ of each agent.

5.5 Summary

This study has proposed a D-FNN model to deal with the inherent issues of the big data environment including the uncertainty and distributed challenge. The proposed D-FNN employed a sentential manner to exploit distributed structure learning and parameter learning based on distributed optimization methods. It's worth noting that the D-FNN is very scalable and does not suffer from slow training speed and gradient vanishing problems compared with back-propagation-based methods.

The consensus learning algorithm has been proposed for the D-FNN in the big data environment. The consensus learning algorithm, which consists of consensus-based distributed structure learning and parameter learning is built on the well-known ADMM. Simulation results have verified the superiority and effectiveness of the proposed consensus learning algorithm for D-FNN.

Chapter 6

Conclusion and Future Work

This research proposed four various fuzzy models to resolve the problems of designing MAT systems, such as the coordination of agents, the interpretability of actions and states, high-dimensional data and data privacy. First, a hierarchical fuzzy logic system was developed for safely navigating multiple robots in complex environments, where the lower-level fuzzy controller is for obstacle avoidance, and the higher-level controller regulates the speeds of robots to enable simultaneous arrival of the targets. The simulation results demonstrate that the proposed models can be adapted to different numbers of robots for navigating the environment and reaching the target on time safely. We further extended this model to improve the transparency of fuzzy sets. A grouping and merging mechanism was developed to optimise transparent fuzzy sets and integrated this mechanism into the training process, thus increasing the fuzzy models' interpretability. The simulation results showed that the optimised fuzzy sets can be interpreted by explaining the fuzzy sets and the consequent components.

Moreover, this research proposed a FCOSTD that is an effective mechanism for discovering and representing the covert states and the transitions between states. The fuzzy-inference mechanism was used to represent the activities of the human brain associated with varying behaviours. Through the distracted driving experiment and visualising the fuzzy-inference-based features, we found that covert states exist in the brain when the subject responds to onset events during distracted driving. The proposed state transition diagram also provides a mechanism for describing state changes with their corresponding probabilities and the Markov chain. The experimental results demonstrate that different subjects have similar states and inter-state transition behaviour but different methods for allocating brain resources as different actions are being taken. The discovery of covert brain states offers machine agents the possibility to understand human cognitive states in a human-autonomous MAT system.

Finally, a D-FNN model was developed to address data privacy and high-dimensional data. The D-FNN used a sentential manner to exploit distributed structure learning and parameter learning based on distributed optimisation methods. The proposed consensus learning, which involves distributed structure learning and distributed parameter learning, can handle the D-FNN model and restrict data exchange among agents. The simulation results on popular datasets demonstrate the proposed consensus learning algorithm's superiority and effectiveness for the D-FNN.

In the future, the fuzzy models proposed in this research can be used to develop advanced multiagent teaming systems that consider the coordination of agents, interpretation of actions and states, and data privacy for different applications and can be used in big-data environments. For example, a human and a machine arm can cooperatively perform assembly tasks in a production line. In such a MAT system, FCOSTD is used to predict the human agent's cognitive state and the machine arm control by a fuzzy-set interpretable fuzzy controller, which can establish a mutual interaction between the human agent and the machine arm to improve task efficiency.

Bibliography

[1] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[2] N. Vlassis, "A concise introduction to multiagent systems and distributed artificial intelligence," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 1, no. 1, pp. 1–71, 2007.

[3] J. Wang, K. Chen, and F. L. Lewis, "Coordination of multi-agent systems on interacting physical and communication topologies," *Systems & Control Letters*, vol. 100, pp. 56–65, 2017.

[4] G. Sadler, H. Battiste, N. Ho, L. Hoffmann, W. Johnson, R. Shively, J. Lyons, and D. Smith, "Effects of transparency on pilot trust and agreement in the autonomous constrained flight planner," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–9.

[5] H. Madduri and N. Gomes, "Designing a cognitive concierge service for hospitals," in *International Conference on Applied Human Factors and Ergonomics*, Springer, 2017, pp. 447–456.

[6] O. Vermesan and J. Bacquet, *Next generation Internet of Things: Distributed intelligence at the edge and human machine-to-machine cooperation*. River Publishers, 2019.

[7] M. Payab and M. Khanzadi, "State of the art and a new methodology based on multi-agent fuzzy system for concrete crack detection and type classification," *Archives of Computational Methods in Engineering*, pp. 1–34, 2020.

[8] S. E. Lyshevski, "Multi-agent distributed coordination and control for aerial systems," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2018, pp. 964–969.

[9] P. Yao and S. Qi, "Obstacle-avoiding path planning for multiple autonomous underwater vehicles with simultaneous arrival," *Science China Technological Sciences*, vol. 62, no. 1, pp. 121–132, 2019.

[10] S. Misra, A. Chakraborty, R. Sharma, and K. Brink, “Cooperative simultaneous arrival of unmanned vehicles onto a moving target in gps-denied environment,” in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 5409–5414.

[11] L. Babel, “Coordinated target assignment and uav path planning with timing constraints,” *Journal of Intelligent & Robotic Systems*, vol. 94, no. 3-4, pp. 857–869, 2019.

[12] H. Ishibuchi, T. Murata, and I. B. Türkşen, “Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems,” *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135–150, 1997.

[13] M. Cococcioni, P. Ducange, B. Lazzerini, and F. Marcelloni, “A pareto-based multi-objective evolutionary approach to the identification of mamdani fuzzy systems,” *Soft Comput*, vol. 11, no. 11, pp. 1013–1031, 2007.

[14] M. J. Gacto, R. Alcalá, and F. Herrera, “Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems,” *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 3, pp. 515–531, 2010.

[15] P. Pulkkinen and H. Koivisto, “A dynamically constrained multiobjective genetic fuzzy system for regression problems,” *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 1, pp. 161–177, 2010.

[16] R. Alcala, M. J. Gacto, and ”. F. Herrera, “Fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems”,” *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 4, pp. 666–681, 2011.

[17] C. Juang, T. Jeng, and Y. Chang, “An interpretable fuzzy system learned through online rule generation and multiobjective aco with a mobile robot control application,” *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2706–2718, 2016.

[18] F. Aghaeipoor and M. M. Javidi, “Mokbl+moms: An interpretable multi-objective evolutionary fuzzy system for learning high-dimensional regression data,” *Information Sciences*, vol. 496, pp. 1–24, 2019.

[19] M. I. Rey, M. Galende, M. J. Fuente, and G. I. Sainz-Palmero, “Multi-objective based fuzzy rule based systems (frbss) for trade-off improvement in accuracy and interpretability: A rule relevance point of view,” *Knowledge-Based Systems*, vol. 127, pp. 67–84, 2017.

[20] X. Bi, X. Zhao, G. Wang, P. Zhang, and C. Wang, "Distributed extreme learning machine with kernels based on mapreduce," *Neurocomputing*, vol. 149, pp. 456–463, 2015.

[21] Y. Ye, M. Xiao, and M. Skoglund, "Decentralized multi-task learning based on extreme learning machines," *arXiv preprint*, pp. 1–11, 2019. arXiv: `1904.11366`.

[22] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. 55, pp. 1663–1707, May 2010.

[23] Y.-Q. Bai, Y.-J. Shen, and K.-J. Shen, "Consensus proximal support vector machine for classification problems with sparse solutions," *Journal of the Operations Research Society of China*, vol. 2, no. 1, pp. 57–74, 2014.

[24] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *International conference on machine learning*, 2016, pp. 2722–2731.

[25] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with admm," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 3466–3473.

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *"Distributed optimization and statistical learning via the alternating direction method of multipliers,"*, vol. 3, no. 1, pp. 1–122, 2011.

[27] R. Fierimonte, M. Barbato, A. Rosato, and M. Panella, "Distributed learning of random weights fuzzy neural networks," in *2016 IEEE International Conference on Fuzzy Systems*, IEEE, 2016, pp. 2287–2294.

[28] R. Fierimonte, R. Altilio, and M. Panella, "Distributed on-line learning for random-weight fuzzy neural networks," in *2017 IEEE International Conference on Fuzzy Systems*, IEEE, 2017, pp. 1–6.

[29] J. K. Pothal and D. R. Parhi, "Navigation of multiple mobile robots in a highly clutter terrains using adaptive neuro-fuzzy inference system," *Robotics and Autonomous Systems*, vol. 72, pp. 48–58, 2015.

[30] D. Chandrasekhar Rao and M. Kabat, “A study on cooperation and navigation planning for multi-robot using intelligent water drops algorithm,” en, in *Emerging Research in Computing, Information, Communication and Applications*, N. Shetty, L. Patnaik, H. Nagaraj, P. Hamsavath, and N. Nalini, Eds., Singapore: Springer, 2019, pp. 577–590.

[31] A. Zhu and S. Yang, “Neurofuzzy-based approach to mobile robot navigation in unknown environments,” en, *IEEE Trans. Syst. Man Cybern. C (Appl. Rev*, vol. 37, pp. 610–621, 2007. DOI: `10.1109/tsmcc.2007.897499`.

[32] C.-F. Juang and Y.-C. Chang, “Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments,” *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 2, pp. 379–392, 2011.

[33] M.-G. Lai, W.-T. Zeng, and C.-F. Juang, “Navigation for two fuzzy controlled cooperative object-carrying robots in concave maps with the consideration of dead-cycle problem,” en, in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE*, Vancouver, BC, Canada: IEEE, 2016.

[34] A. Din, M. Jabeen, K. Zia, A. Khalid, and D. Saini, “Behavior-based swarm robotic search and rescue using fuzzy controller,” es, *Comput. Electr. Eng*, vol. 70, pp. 53–65, 2018. DOI: `10.1016/j.compeleceng.2018.06.003`.

[35] J.-Y. Jhang, C.-J. Lin, and K.-Y. Young, “Cooperative carrying control for multi-evolutionary mobile robots in unknown environments,” en, *Electronics*, vol. 8, p. 298, 2019. DOI: `10.3390/electronics8030298`.

[36] J. Mohanta and A. Keshari, “A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation,” en, *Appl. Soft Comput*, vol. 79, pp. 391–409, 2019. DOI: `10.1016/j.asoc.2019.03.055`.

[37] B. Pradhan, D. Roy, and N. Hui, “Multi-agent navigation and coordination using ga-fuzzy approach,” da, in *Soft Computing for Problem Solving*, J. Bansal, K. Das, A. Nagar, K. Deep, and A. Ojha, Eds., Singapore: Springer, 2019, pp. 793–805.

[38] J. Chia-Feng, “Combination of online clustering and q-value based ga for reinforcement fuzzy system design,” en, *IEEE Trans. Fuzzy Syst*, vol. 13, pp. 289–302, 2005. DOI: `10.1109/tfuzz.2004.841726`.

[39] E. Mansoori, M. Zolghadri, and S. Katebi, “Sgerd: A steady-state genetic algorithm for extracting fuzzy classification rules from data,” en, *IEEE Trans. Fuzzy Syst*, vol. 16, pp. 1061–1071, 2008. DOI: `10.1109/tfuzz.2008.915790`.

[40] S. Nantogma, W. Ran, X. Yang, and H. Xiaoqin, “Behavior-based genetic fuzzy control system for multiple usvs cooperative target protection,” en, in *2019 3rd International Symposium on Autonomous Systems (ISAS*, Shanghai, China: IEEE, 2019, pp. 181–186.

[41] C.-F. Juang and C. Lo, “Zero-order tsk-type fuzzy system learning using a two-phase swarm intelligence algorithm,” en, *Fuzzy Sets Syst*, vol. 159, pp. 2910–2926, 2008. DOI: `10.1016/j.fss.2008.02.003`.

[42] W. Ding, C. Lin, and Z. Cao, “Deep neuro-cognitive co-evolution for fuzzy attribute reduction by quantum leaping pso with nearest-neighbor memeplexes,” en, *IEEE Trans. Cybernetics*, vol. 49, pp. 2744–2757, 2019. DOI: `10.1109/TCYB.2018.2834390`.

[43] C.-T. Lin and C. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, no. Upper Saddle River, NJ: Prentice Hall, 1996.

[44] C.-F. Juang and C.-T. Lin, “An online self-constructing neural fuzzy inference network and its applications,” *IEEE transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.

[45] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” en, in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360*, Anchorage, AK, USA: IEEE, 1998, pp. 69–73.

[46] K. Greff, R. Srivastava, J. Koutnik, B. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” en, *IEEE Trans. Neural Netw. Learn. Syst*, vol. 28, pp. 2222–2232, 2017. DOI: `10.1109/tnnls.2016.2582924`.

[47] Cyberbotics, *Webots: Robot simulator*, en, Available: [Online]. Available: `https://cyberbotics.com/`.

[48] W. Zhang and Y. Zhang, “Behavior switch for DRL-based robot navigation,” in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, IEEE, 2019, pp. 284–288.

[49] Y.-C. Chang, A. Dostovalova, C.-T. Lin, and J. Kim, “Intelligent multi-robot navigation and time-arrival control using a scalable pso-optimised hierarchical controller,” *Frontiers in Artificial Intelligence*, vol. 3, no. 50, pp. 1–12, 2020.

[50] Z. Wang, C. Cheng, P. Blunsom, A. Markham, L. Xie, N. Trigoni, Y. Miao, and W. Sen, *Learning with stochastic guidance for robot navigation.* IEEE Transactions on Neural Networks and Learning Systems, 2020.

[51] J.-D. Haynes and G. Rees, “Decoding mental states from brain activity in humans,” en, *Nature Reviews Neuroscience*, vol. 7, no. 7, pp. 523–534, Jul. 2006, number: 7 publisher: Nature Publishing Group, ISSN: 1471-0048. DOI: 10.1038/nrn1931.

[52] A. T. Baria, B. Maniscalco, and B. J. He, “Initial-state-dependent, robust, transient neural dynamics encode conscious visual perception,” *PLOS Computational Biology*, vol. 13, no. 11, pp. 1–29, Nov. 2017. DOI: 10.1371/journal.pcbi.1005806. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1005806.

[53] J. Taghia, W. Cai, S. Ryali, J. Kochalka, J. Nicholas, T. Chen, and V. Menon, “Uncovering hidden brain state dynamics that regulate performance and decision-making during cognition,” en, *Nature Communications*, vol. 9, no. 1, pp. 1–19, Jun. 27, 2018, ISSN: 2041-1723. DOI: 10.1038/s41467-018-04723-6.

[54] M. L. Kringelbach, J. Cruzat, J. Cabral, G. M. Knudsen, R. Carhart-Harris, P. C. Whybrow, N. K. Logothetis, and G. Deco, “Dynamic coupling of whole-brain neuronal and neurotransmitter systems,” en, *Proceedings of the National Academy of Sciences*, vol. 117, no. 17, pp. 9566–9576, Apr. 28, 2020, ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1921475117.

[55] H. B. Celikoglu and M. A. Silgu, “Extension of traffic flow pattern dynamic classification by a macroscopic model using multivariate clustering,” *Transportation Science*, vol. 50, no. 3, pp. 966–981, Feb. 22, 2016, ISSN: 0041-1655. DOI: 10.1287/trsc.2015.0653.

[56] A. Krause, A. Smailagic, and D. Siewiorek, “Context-aware mobile computing: Learning context- dependent personal preferences from a wearable sensor array,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 113–127, Feb. 2006, ISSN: 2161-9875. DOI: 10.1109/TMC.2006.18.

[57] M. A. Silgu and H. B. Celikoglu, “Clustering traffic flow patterns by fuzzy c-means method: Some preliminary findings,” en, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2015, pp. 756–764, ISBN: 978-3-319-27340-2. DOI: 10.1007/978-3-319-27340-2_93.

[58] M. Xing, O. Ajilore, O. E. Wolfson, C. Abbott, A. MacNamara, R. Tadayonnejad, A. Forbes, K. L. Phan, H. Klumpp, and A. Leow, “Thought chart: Tracking dynamic eeg brain connectivity with unsupervised manifold learning,” en, G. A. Ascoli, M. Hawrylycz, H. Ali, D. Khazanchi, and Y. Shi, Eds.,

ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2016, pp. 149–157, ISBN: 978-3-319-47103-7. DOI: 10.1007/978-3-319-47103-7_15.

[59] L. C. C. Heredia and A. R. Mor, "Density-based clustering methods for unsupervised separation of partial discharge sources," *International Journal of Electrical Power & Energy Systems*, vol. 107, pp. 224–230, 2019.

[60] A. K. Jain, "Data clustering: 50 years beyond k-means," en, *Pattern Recognition Letters*, Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), vol. 31, no. 8, pp. 651–666, Jun. 1, 2010, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2009.09.011.

[61] P. Liu, D. Zhou, and N. Wu, "Vdbscan: Varied density based spatial clustering of applications with noise," ISSN: 2161-1904, 2007 International Conference on Service Systems and Service Management, Jun. 2007, pp. 1–4. DOI: 10.1109/ICSSSM.2007.4280175.

[62] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," en, *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 27, 2014, PMID: 24970081, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1242072.

[63] K. Pal, N. R. Pal, J. M. Keller, and J. C. Bezdek, "Relational mountain (density) clustering method and web log analysis," en, *International Journal of Intelligent Systems*, vol. 20, no. 3, pp. 375–392, 2005, ISSN: 1098-111X. DOI: 10.1002/int.20071.

[64] R. H. Hariri, E. M. Fredericks, and K. M. Bowers, "Uncertainty in big data analytics: Survey, opportunities, and challenges," *Journal of Big Data*, vol. 6, no. 1, p. 44, Jun. 4, 2019, ISSN: 2196-1115. DOI: 10.1186/s40537-019-0206-3.

[65] C.-T. Lin, C.-H. Chuang, Y.-K. Wang, S.-F. Tsai, T.-C. Chiu, and L.-W. Ko, "Neurocognitive characteristics of the driver: A review on drowsiness, distraction, navigation, and motion sickness," en, *Journal of Neuroscience and Neuroengineering*, vol. 1, no. 1, pp. 61–81, Jun. 2012, DOI: info:doi/10.1166/jnsne.2012.1010.

[66] Y.-T. Liu, Y.-Y. Lin, S.-L. Wu, C.-H. Chuang, and C.-T. Lin, "Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 347–360, Feb. 2016, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2015.2496330.

[67] I. Caylak, E. Penner, and R. Mahnken, “A fuzzy uncertainty model for analytical and numerical homogenization of transversely fiber reinforced plastics,” en, *PAMM*, vol. 19, no. 1, e201900356, 2019, ISSN: 1617-7061. DOI: 10.1002/pamm.201900356.

[68] I. Couso, C. Borgelt, E. Hullermeier, and R. Kruse, “Fuzzy sets in data analysis: From statistical foundations to machine learning,” *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 31–44, 2019.

[69] F.-C. Lin, L.-W. Ko, C.-H. Chuang, T.-P. Su, and C.-T. Lin, “Generalized eeg-based drowsiness prediction system by using a self-organizing neural fuzzy system,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 9, pp. 2044–2055, Sep. 2012, ISSN: 1558-0806. DOI: 10.1109/TCSI.2012.2185290.

[70] J.-S. Jang, “ANFIS: Adaptive-network-based fuzzy inference system,” *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[71] A. Jafarifarmand, M. A. Badamchizadeh, S. Khanmohammadi, M. A. Nazari, and B. M. Tazehkand, “A new self-regulated neuro-fuzzy framework for classification of eeg signals in motor imagery bci,” *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1485–1497, Jun. 2018, ISSN: 1941-0034. DOI: 10.1109/TFUZZ.2017.2728521.

[72] W. Jamal, S. Das, I.-A. Oprescu, and K. Maharatna, “Prediction of synchrostate transitions in eeg signals using markov chain models,” *IEEE Signal Processing Letters*, vol. 22, no. 2, pp. 149–152, Feb. 2015, ISSN: 1558-2361. DOI: 10.1109/LSP.2014.2352251.

[73] B. Kemp and H. A. C. Kamphuisen, “Simulation of human hypnograms using a markov chain model,” en, *Sleep*, vol. 9, no. 3, pp. 405–414, Sep. 1986, ISSN: 0161-8105, 1550-9109. DOI: 10.1093/sleep/9.3.405.

[74] S. Ryali, K. Supekar, T. Chen, J. Kochalka, W. Cai, J. Nicholas, A. Padmanabhan, and V. Menon, “Temporal dynamics and developmental maturation of salience, default and central-executive network interactions revealed by variational bayes hidden markov modeling,” en, *PLOS Computational Biology*, vol. 12, no. 12, e1005138, Dec. 13, 2016, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005138.

[75] P. Salvador, A. Nogueira, and R. Valadas, “Markovian models for medical signals on wireless sensor networks,” ISSN: 2164-7038, 2009 IEEE International Conference on Communications Workshops, Jun. 2009, pp. 1–5. DOI: 10.1109/ICCW.2009.5208090.

[76] U. Bodenhofer and P. Bauer, “Interpretability of linguistic variables: A formal account,” en, *Kybernetika*, vol. 41, no. 2, pp. 227–248, 2005.

[77] C. Wagner, M. Smith, K. Wallace, and A. Pourabdollah, “Generating uncertain fuzzy logic rules from surveys: Capturing subjective relationships between variables from human experts,” ISSN: null, 2015 IEEE International Conference on Systems, Man, and Cybernetics, Oct. 2015, pp. 2033–2038. DOI: 10.1109/SMC.2015.355.

[78] H. Iyatomi and M. Hagiwara, “Knowledge extraction from scenery images and the recognition using fuzzy inference neural networks,” ISSN: 1062-922X, SMC’98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), vol. 5, Oct. 1998, 4486–4491 vol.5. DOI: 10.1109/ICSMC.1998.727557.

[79] Y.-K. Wang, T.-P. Jung, and C.-T. Lin, “Eeg-based attention tracking during distracted driving,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 6, pp. 1085–1094, Nov. 2015, ISSN: 1558-0210. DOI: 10.1109/TNSRE.2015.2415520.

[80] ——, “Theta and alpha oscillations in attentional interaction during distracted driving,” en, *Frontiers in Behavioral Neuroscience*, vol. 12, p. 3, Feb. 9, 2018, ISSN: 1662-5153. DOI: 10.3389/fnbeh.2018.00003.

[81] Y.-K. Wang, S.-A. Chen, and C.-T. Lin, “An eeg-based brain–computer interface for dual task driving detection,” en, *Neurocomputing*, vol. 129, pp. 85–93, Apr. 10, 2014, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2012.10.041.

[82] E. Butkevičiūtė, L. Bikulčienė, T. Sidekerskienė, T. Blažauskas, R. Maskeliūnas, R. Damaševičius, and W. Wei, “Removal of movement artefact for mobile eeg analysis in sports exercises,” *IEEE Access*, vol. 7, pp. 7206–7217, 2019, event: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2890335.

[83] A. Kumar, Q. Fang, J. Fu, E. Pirogova, and X. Gu, “Error-related neural responses recorded by electroencephalography during post-stroke rehabilitation movements,” English, *Frontiers in Neurorobotics*, vol. 13, 2019, publisher: Frontiers, ISSN: 1662-5218. DOI: 10.3389/fnbot.2019.00107. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2019.00107/full.

[84] S. Shirinpour, I. Alekseichuk, K. Mantell, and A. Opitz, "Experimental evaluation of methods for real-time eeg phase-specific transcranial magnetic stimulation," en, *Journal of Neural Engineering*, vol. 17, no. 4, p. 046 002, Jul. 2020, publisher: IOP Publishing, ISSN: 1741-2552. DOI: 10.1088/1741-2552/ab9dba.

[85] T.-P. Jung, S. Makeig, M. McKeown, A. Bell, T.-W. Lee, and T. Sejnowski, "Imaging brain dynamics using independent component analysis," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1107–1122, Jul. 2001, ISSN: 1558-2256. DOI: 10.1109/5.939827.

[86] T.-P. Jung, S. Makeig, A. J. Bell, and T. J. Sejnowski, "Independent component analysis of electroencephalographic and event-related potential data," in *Central auditory processing and neural modeling*, Springer, 1998, pp. 189–197.

[87] C.-T. Lin, S.-A. Chen, T.-T. Chiu, H.-Z. Lin, and L.-W. Ko, "Spatial and temporal eeg dynamics of dual-task driving performance," en, *Journal of NeuroEngineering and Rehabilitation*, vol. 8, no. 1, p. 11, 2011, ISSN: 1743-0003. DOI: 10.1186/1743-0003-8-11.

[88] C.-T. Lin, Y.-T. Liu, S.-L. Wu, Z. Cao, Y.-K. Wang, C.-S. Huang, J.-T. King, S.-A. Chen, S.-W. Lu, and C.-H. Chuang, "Eeg-based brain-computer interfaces: A novel neurotechnology and computational intelligence method," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 3, no. 4, pp. 16–26, Oct. 2017, ISSN: 2380-1298. DOI: 10.1109/MSMC.2017.2702378.

[89] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," en, *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1, 1987, ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7.

[90] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American statistical association*, vol. 78, no. 383, pp. 553–569, 1983.

[91] L. Jäncke, B. Brunner, and M. Esslen, "Brain activation during fast driving in a driving simulator: The role of the lateral prefrontal cortex:" en, *NeuroReport*, vol. 19, no. 11, pp. 1127–1130, Jul. 2008, ISSN: 0959-4965. DOI: 10.1097/WNR.0b013e3283056521.

[92] G. Sammer, C. Blecker, H. Gebhardt, M. Bischoff, R. Stark, K. Morgen, and D. Vaitl, "Relationship between regional hemodynamic activity and simultaneously recorded eeg-theta associated with mental arithmetic-induced workload," vol. 28, no. 8, pp. 793–803, Aug. 2007.

[93] L. Georgopoulos and M. Hasler, “Distributed machine learning in networks by consensus,” *Neurocomputing*, vol. 124, pp. 2–12, 2014.

[94] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, “State-of-the-art in privacy preserving data mining,” *ACM Sigmod Record*, vol. 33, no. 1, pp. 50–57, 2004.

[95] D. Chen and H. Zhao, “Data security and privacy protection issues in cloud computing,” in *2012 International Conference on Computer Science and Electronics Engineering*, IEEE: vol. 1, 2012, pp. 647–651.

[96] M. Li, W. Lou, and K. Ren, “Data security and privacy in wireless body area networks,” *IEEE Wireless communications*, vol. 17, no. 1, pp. 51–58, 2010.

[97] Y. Wang, W. Yin, and J. Zeng, “Global convergence of ADMM in nonconvex nonsmooth optimization,” *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[98] S. L. Chiu, “Fuzzy model identification based on cluster estimation,” *Journal of Intelligent & fuzzy systems*, vol. 2, no. 3, pp. 267–278, 1994.

[99] J. de Jesús Rubio, “SOFMLS: Online self-organizing fuzzy modified least-squares network,” *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[100] J. A. M. Hernández, F. G. Casta neda, and J. A. M. Cadenas, “An evolving fuzzy neural network based on the mapping of similarities,” *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1379–1396, 2009.

[101] J. MacQueen, “Et al,” in *Some methods for classification and analysis of multivariate observations*, vol. 1, , Oakland, CA, USA: in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, pp. 281–297.

[102] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[103] P. A. Forero, A. Cano, and G. B. Giannakis, “Distributed clustering using wireless sensor networks,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, 2011.

[104] S. Scardapane, D. Wang, and M. Panella, “A decentralized training algorithm for echo state networks in distributed big data applications,” *Neural Networks*, vol. 78, pp. 65–74, 2016.

[105] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[106] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2010.

[107] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014.

[108] *Predict house price in king county using regression.* `https://www.kaggle.com/harlfoxem/housesalesprediction`, [Online; accessed 18-August-2019].

[109] *Physicochemical properties of protein tertiary structure data set.* `https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure`, [Online; accessed 20-July-2019].

[110] W. Kirchgässner, O. Wallscheid, and J. Böcker, "Deep residual convolutional and recurrent neural networks for temperature estimation in permanent magnet synchronous motors," *2019 IEEE International Electric Machines Drives Conference (IEMDC)*, pp. 1439–1446, 2019.