VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Solving Vehicle Platooning Problem with Neural Networks

Řešení problému s četou vozidel

pomocí neuronových sítí

Duy Quy Vo

Bachelor Thesis

Supervisor: Hossein Barghi Jond, Ph.D.

Ostrava, 2022

VSB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Bachelor Thesis Assignment

Student: **Duy Quy Vo**

Study Programme: B2647 Information and Communication Technology

Study Branch: 2612R025 Computer Science and Technology

Title: Solving Vehicle Platooning Problem with Neural Networks
Řešení problému s četou vozidel pomocí neuronových sítí

The thesis language: English

Description:

The main idea behind a platoon (or string) is the longitudinal control of a group of autonomous vehicles moving in the same lane of the highway. The first vehicle usually leads the platoon and specifies the trajectory and speed for the follower vehicles. To act the platoon as a whole a control mechanism must be employed. Optimal control is a common approach to design a controller for vehicle control problems. In this work, the vehicle platooning will be formulated as an optimal control problem. Due to the complexity of finding an analytic solution to the problem, a neural network will be employed to solve the formulated control problem. Simulations will be carried out to verify the validity of the models and solutions.

The thesis should be organized as follows:
1. Introduction to vehicle platooning and its application and importance in the future intelligent transportation systems.
2. Formulation of the vehicle platooning as an optimal control problem.
3. Designing a neural network to solve the formulated problem.
4. Simulation results on the validity of the models and solutions.
5. Conclusion.

References:

[1] https://www.youtube.com/watch?v=X7vziDnNXEY&ab_channel=ScaniaNederland
[2] M. A. Huijzer, Truck platooning for a convoy of heterogeneous trucks, Bachelor Thesis, University of Groningen, 2017. [3] S. Effati, M. Pakdaman, Optimal control problem via neural networks, Neural Computing and Applications, Vol. 23, pp. 2093–2100, 2013.

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: **Hossein Barghi Jond, Ph.D.**

Date of issue: 01.09.2021

Date of submission: 30.04.2022

doc. Ing. Petr Gajdoš, Ph.D.	prof. Ing. Jan Platoš, Ph.D.
Head of Department	*Dean*

Abstrakt

Tento výzkum se snaží vyřešit problém kontroly čety pomocí schopností umělých neuronových sítí. Protože je obtížné získat analytické řešení problému řízení čety, je před hledáním přibližných řešení upřednostňována schopnost aproximace funkcí neuronových sítí. V této práci vytváříme chybovou funkci, která zahrnuje všechny podmínky odvozené z Pontryaginova minimálního principu (PMP) pro řízení formace čety v topologii navazující na předchůdce. Experimentální řešení pro stavovou funkci, Řídicí funkci a Lagrangeovy multiplikátory je odvozeno z chybové funkce. Funkce stavu, řízení a nákladů jsou implementovány pomocí neuronových sítí. Pro optimalizaci váhy pro chybovou funkci se používají optimalizační techniky, jako je genetický algoritmus (GA), diferenciální evoluce (DE) a variace algoritmu samoorganizujícího se migračního algoritmu (SOMA). Nahradíme optimalizované váhy v aproximovaných funkcích a získáme řešení problému řízení formace čety.

Klíčová slova

Pontryaginův minimální princip; problém optimálního řízení; umělé neuronové sítě; připojené a automatizované vozidlo (CAV); řízení formace čety

Abstract

This research seeks to solve the platoon control problem using the abilities of Artificial Neural Networks. Because it is difficult to obtain an analytical solution to the platoon control problem, the function approximation ability of Neural Networks is preferred to search for approximate solutions. In this thesis, we create an error function that encompasses all of the Pontryagin minimum principle (PMP) derived conditions for the platoon formation control under the Predecessor-following topology. The experimental solution for the state function, control function, and Lagrange multipliers is derived from the error function. State, control, and cost functions are implemented using Neural Networks. To optimize weights for the error function, optimization techniques such as the Genetic Algorithm (GA), Differential Evolution (DE), and variations of the self-organizing migrating algorithm (SOMA) algorithm are used. We substitute the optimized weights in the approximated functions and get the solutions to the platoon formation control problem.

Keywords

Pontryagin minimum principle; Optimal control problem; Artificial Neural Networks; Connected and automated vehicle (CAV); Platoon formation control

Acknowledgement

It is my pleasure to express my gratitude to my supervisor, Hossein Barghi Jond, Ph.D., for his essential counsel, unwavering support, and patience during my studies; his vast knowledge and expertise have inspired me throughout my academic career. Czech Republic is a country in Central Europe. For those who don't know, I'd want to convey my heartfelt thanks to my parents. Thank you to my parents and family for their inspiration and support.

Contents

List of symbols and abbreviations

PF	–	Predecessor-following
PMP	–	Pontryagin Minimum Principle
LFA	–	Lane Following Assist
V2V	–	Vehicle-to-Vehicle Communication
CAVs	–	Connected and Automated Vehicles
CNN	–	Convolutional Neural Network
RNN	–	Recurrent Neural Network
ODEs	–	Ordinary Differential Equations
DE	–	Differential Evolution
FEs	–	Function Evaluations
GA	–	Genetic Algorithm
iSOMA	–	Self-Organizing Migrating Algorithm with Narrowing Search Space Strategy
MaxFEs	–	Maximum of Function Evaluations
SOMA	–	Self-Organizing Migrating Algorithm
SOMA ATO	–	Self-Organizing Migrating Algorithm All To One Version
SOMA T3A	–	Self-Organizing Migrating Algorithm Team To Team Adaptive

List of Figures

List of Tables

Chapter 1

Introduction

In this world of technology , autonomous vehicles using Artificial Intelligence is among the most important and fast-growing fields of study of Deep Learning. Autonomous vehicle is a ground-based vehicle with integrated vehicle automation. These vehicles senses its surroundings so it can move safely without human intervention. Facing the problem of increasing traffic and traffic safety, autonomous vehicles help respond to increased traffic, density, efficiency, safety and comfort when traveling. Many self-driving vehicles solutions have been tested around the world. Research efforts continue to be made to improve the performance, reliability and cut costs of autonomous vehicles.

Autonomous vehicles are integrated with many sensors to collect information from the surrounding environment such as thermographic cameras, Global Positioning System (GPS), and so on. Advanced control systems receive information from sensors and analyze them to determine suitable navigation for autonomous vehicles. Autonomous vehicle navigation [1] consists of three main steps: Perception and localization – Planning – Control. Among these steps, the motion control task plays an important role in determining the overall performance in navigation systems. Recently, researchers proposed methods for optimizing this task via advancements in Artificial Intelligence, especially Deep Neural Networks. The goal of the motion controller is to ensure that the vehicle follows a desired path by minimizing the error between the vehicle and the reference path. At the same time, the speed of the vehicle must also be stable, to avoid collisions with other vehicles [2, 3].

In transportation, there is rarely a single but a group of autonomous or manned vehicles joining traffic together. This group is usually preferred to as a platoon. And platooning problem is the process of routing self-driving vehicles based on a leading vehicle given a set of starting points and deadlines. The leading vehicle can be an autonomous vehicle or a manned vehicle. This vehicle decides the route, speed and status of the whole platoon. Platooning brings great potential benefits such as: reduce fuel consumption because of the decrease in the need for acceleration, deceleration, and stopping to maintain traffic flow; reduce congestion; substantially shorter commutes during rush hours; and fewer traffic collisions. For example, when the distance between vehicles in the platoon is close enough, it makes better use of the road, saving time, energy and reducing emissions[4, 5].

The primary objective of this research is to optimize a platoon of autonomous vehicles with a leading vehicle using Neural Networks. We suggests to use the Predecessor-following topology (PF) to get an error function containing all PMP conditions [6]. The error function proposes an experimental solution for state, control, and cost functions. The Neural Networks are implemented for each function in the experimental solution: state function, cost function, control function, respectively. The main optimization technique used in this research is to apply Heuristic algorithms to optimize the weights of Neural Networks. When obtained the Neural Network with optimized weights, we substitute the optimal weights into the approximation solution and get the solution to the platoon control problem.

The motivation behind this approach is the ability of Computational Artificial Intelligence to locate vehicles in a platoon. Computational Artificial Intelligence using computer technology allows solving problems with very high accuracy to meet the requirements of increasing accuracy of platoon control problem. Artificial Neural Network is an outstanding application of Computational Artificial Intelligence with the ability to work like human being, learn from experiential data, and have high fault tolerance. The Artificial Neural Network [7] determines the relationship between input and output parameters for high accuracy combined with optimization algorithms to optimize weights to achieve the lowest error value.

In section 2, we introduce the platoon control problem under the PF topology [8] and the model of Neural Network. The construction of a Neural Network to solve the problem is covered in section 3. Section 4 tells us how to set up to get the Neural Network Solver. The simulation results as well as the efficiency of the optimization algorithms shown in Section 5 and Section 6 gives the final conclusions and possible development directions.

Chapter 2

Problem Formation

In this part, we look at different techniques to solve the problem of vehicle platooning. A platooning control problem emerges to control the collective motions of vehicles as a platoon on the roads. These vehicles are autonomously operated and linked to one another via a communication architecture (CT). By this association, the location information of the vehicles is provided to find the relative distance between the vehicles and optimize it to solve the platoon problem. To find the optimal controller [9] for each vehicle in the platoon, it is assumed that vehicles form a platoon according to the Predecessor-following topology (2.1). The purpose of this part is to apply the optimal control principles to formulate the platoon formation control as an optimal control problem. This problem has been discussed in [10, 11] under the framework of differential games and individual trajectories for the PF topology have been obtained. The optimal control problem in this thesis is a simplified version of the proposed differential game problem in [10, 11].

Generating the individual trajectories of the PF topology requires high computational performance. Meeting this need, Neural Networks not only have high computational power but also large capacity to model the platoon problem. The approach of the Neural Network to generic topology is the major aim of section (2.2). Additionally, to optimize the results obtained from the Neural Network, numerical approaches were used for the problem and the Hamilton [12] function.

2.1 Platoon Formulation

A control problem is presented to imitate the real-world challenge we face in vehicle platooning. In this problem, we consider the control a platoon of CAVs going in the PF topology, as shown in Figure 2.1.

There are two types of links in the platoon that we need to pay attention to in terms of connectivity. First, a sensor link to get information of the predecessor vehicle. The source of information is its predecessor vehicle. This implies that information can only be transmitted from the vehicle in front to the vehicle near it in order to maintain a single and one-way communication flow through-

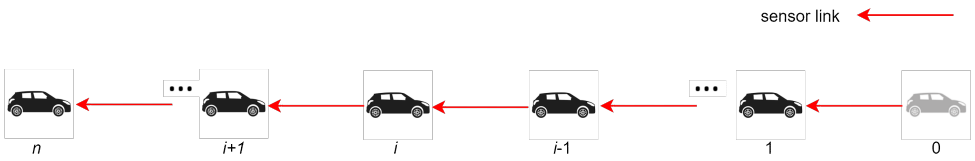

Figure 2.1: A vehicle platoon with predecessor-following (PF) topology. The figure is reprinted from [11].

out the platoon. On the other hand, a V2V connection is the link between two sequential vehicle. The direction of information transmission of V2V connection between two of these vehicles is not restricted, just as with a sensor link. The V2V connection allows us to connect the car in front to the car behind and vice versa. More information on this link can be found in [13].

In order to better understand the vehicle platoon arrangement, assume a platoon of n CAVs traveling with a vehicle in the lead. This platoon is follow the traffic lane in which the lead vehicle is traveling. According to the PF topology, depending on information received from the sensor connection, this platoon automatically changes the distance between each vehicle and the vehicle ahead of it in the platoon. As a result, when the vehicle leading the platoon changes traffic lanes or the speed causes a change in distance with the platoon's vehicle following it, the vehicles behind it immediately alter their behavior according to the PF topology.

As indicated in Figure 2.1, the platoon control problem has been concretized. The reference (x_0) is a moving vehicle. The reference (x_0) can be manned or unmanned, but we don't need to worry about it because it won't link the V2V to any of the vehicles in our platoon. Figure 1 shows that the vehicle leading the platoon is numbered 1 if it is a real vehicle and 0 if it is a virtual vehicle, from right to left. Lane Following Assist (LFA) [14], for example, produces a reference trajectory that supports the lead vehicle and vehicles in platoons. The platoon's real vehicles are indexed as $1, \ldots, n$.

Let x_i and u_i indicate the longitudinal position and control input of the i^{th} vehicle ($i = 1, \ldots, n$), respectively. The longitudinal dynamics of inter-vehicle distance policies can be characterized as

$$\dot{x}_i - \dot{x}_{i-1} = u_i. \tag{2.1}$$

The platoon formation control cost function is defined as:

$$J(x_0, \cdots, x_n, u_1, \cdots, u_n) = \frac{1}{2} \int_0^{t_f} \sum_{i=1}^{n} ((x_i - x_{i-1} - d_i)^2 + u_i^2)\ dt, \tag{2.2}$$

where t_f is a limited time horizon.

To ensure traffic safety, the PF topology used for platoons must also ensure that no collision occurs. In other words $x_i - x_{i-1-d_i} < 0$. Using information from the sensor, each vehicle perceives its relative distance from its predecessor. From there, the vehicle will adjust the appropriate speed to

avoid collisions causing traffic accidents. The platoon does not need a communication environment between autonomous vehicles because there is no V2V connection. Using information from the distance sensor with the predecessor vehicle avoids the situation when the information exchange network is hacked or the information sent is faulty.

Assumption 1: (Collision-avoidance) The initial positions hold $x_0(0) > x_1(0) > \ldots > x_n(0)$ and $d_i < 0$ $(i = 1, \ldots, n)$ since the its direction is opposite to the traffic flow [11].

Under **Assumption 1**, collision-avoidance is embedded in the platoon formation control given by (2.1) and (2.2).

Assumption 2: (Reference trajectory) We assume $\dot{x}_0 = 0$, i.e., the reference has a constant speed [11].

To make the dynamics and cost functions in (2.1) and (2.2) easier to calculate, we may substitute the individual location of each vehicle with a value of relative distance between two vehicles through sensor links.

Let's define the relative distance $y_i = x_i - x_{i-1}$ that can be directly monitored via the sensor connection for vehicle i $(i = 1, \ldots, n)$. The platoon formation control in (2.1) and (2.2) may be recast as the minimizing of the following optimization:

$$\min_{u_1, \cdots, u_n} J(y_1, \cdots, y_n, u_1, \cdots, u_n) = \frac{1}{2} \int_0^{t_f} \sum_{i=1}^{n} ((y_i - d_i)^2 + u_i^2) dt, \tag{2.3}$$

s.t.

$$\dot{y}_i = u_i, \quad y_i(0) = x_i(0) - x_{i-1}(0), \quad i = 1, \ldots, n.$$

We quote the closed-form solution of above optimal control problem from [11] in the following:

Theorem 1 *For a n-vehicle platoon defined as the optimal control problem (2.3) the relative distances trajectories y_i's are obtained as [11]*

$$y_i(t) = \alpha(t) y_i(0) + (\alpha(t) - 1) d_i, \tag{2.4}$$

where

$$\alpha(t) = \frac{\cosh(t_f - t)}{\cosh(t_f)}. \tag{2.5}$$

Proof Define the Hamiltonian

$$H = \frac{1}{2} \sum_{i=1}^{n} ((y_i - d_i)^2 + u_i^2 + \lambda_i u_i) \tag{2.6}$$

where λ_i is the costate.

According to the Pontryagin's minimum principle, the necessary conditions for optimality are $\frac{\partial H}{\partial y_i} = -\dot{\lambda}_i$ and $\frac{\partial H}{\partial \lambda_i} = \dot{y}_i$, $\frac{\partial H_i}{\partial u_i} = 0$. Applying the necessary conditions on (2.6) yields:

$$\dot{\lambda}_i = -y_i + d_i, \quad , \lambda_i(t_f) = 0 \tag{2.7}$$

$$u_i = \dot{y}_i \tag{2.8}$$

$$u_i = -\lambda_i \tag{2.9}$$

for $i = 1, \ldots, n$.

Substituting (2.7) in relative dynamics results in

$$\dot{y}_i = -\lambda_i, \quad y_i(0) = x_i(0) - x_{i-1}(0), \quad i = 1, \ldots, n. \tag{2.10}$$

Let $\boldsymbol{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$, $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^\top \in \mathbb{R}^n$, and $\boldsymbol{d} = [d_1, \ldots, d_n]^\top \in \mathbb{R}^n$. Also, let $\mathbf{0}$ and $\boldsymbol{I}$ denote the zero and identity matrix of appropriate dimension. The equations (2.7), (2.9) and (2.10) can be unified into the following differential equation

$$\begin{bmatrix} \dot{\boldsymbol{y}} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & -\boldsymbol{I} \\ -\boldsymbol{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{d} \end{bmatrix}, \tag{2.11}$$

with the initial condition vector $\boldsymbol{y}(0)$ where $y_i(0) = x_i(0) - x_{i-1}(0)$ $(i = 1, \ldots, n)$ and terminal condition vector $\boldsymbol{\lambda}(t_f)$ where $\lambda_i(t_f) = 0$ $(i = 1, \ldots, n)$.

If there is a solution, then equation (2.11) has a solution for each $\boldsymbol{y}(0)$ and $\boldsymbol{\lambda}(t_f)$. Matrix analyses in Laplace Transform domain [15] (which transforms equation (2.11) into a more solvable algebraic equation) then shows that (2.11) has a solution as follows

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\lambda} \end{bmatrix} = \boldsymbol{\Phi}(t) \begin{bmatrix} \boldsymbol{y}(0) \\ \boldsymbol{\lambda}(0) \end{bmatrix} + \boldsymbol{\Psi}(t, 0)\boldsymbol{d}, \tag{2.12}$$

where

$$\boldsymbol{\Phi}(t) = \begin{bmatrix} \boldsymbol{\Phi}_{11}(t) & \boldsymbol{\Phi}_{12}(t) \\ \boldsymbol{\Phi}_{21}(t) & \boldsymbol{\Phi}_{22}(t) \end{bmatrix} = \mathcal{L}^{-1}\{ \begin{bmatrix} s\boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & s\boldsymbol{I} \end{bmatrix}^{-1} \} = \tag{2.13}$$

$$\mathcal{L}^{-1}\{ \begin{bmatrix} s(s^2\boldsymbol{I} - \boldsymbol{I})^{-1} & -(s^2\boldsymbol{I} - \boldsymbol{I})^{-1} \\ -\boldsymbol{I}(s^2\boldsymbol{I} - \boldsymbol{I})^{-1} & s(s^2\boldsymbol{I} - \boldsymbol{I})^{-1} \end{bmatrix} \} = \begin{bmatrix} \cosh(t)\boldsymbol{I} & -\sinh(t)\boldsymbol{I} \\ -\sinh(t)\boldsymbol{I} & \cosh(t)\boldsymbol{I} \end{bmatrix},$$

and

$$\boldsymbol{\Psi}(t, 0) = \begin{bmatrix} \boldsymbol{\Psi}_1(t, 0) \\ \boldsymbol{\Psi}_2(t, 0) \end{bmatrix} = \int_0^t \begin{bmatrix} \boldsymbol{\Phi}_{12}(t - \tau) \\ \boldsymbol{\Phi}_{22}(t - \tau) \end{bmatrix} d\tau = \begin{bmatrix} -(1 - \cosh(t))\boldsymbol{I} \\ -\sinh(t)\boldsymbol{I} \end{bmatrix}. \tag{2.14}$$

From (2.12), $\boldsymbol{y}$ and $\boldsymbol{\lambda}$ are obtained as

$$\boldsymbol{y} = \boldsymbol{\Phi}_{11}(t)\boldsymbol{y}(0) + \boldsymbol{\Phi}_{12}(t)\boldsymbol{\lambda}(0) + \boldsymbol{\Psi}_1(t,0)\boldsymbol{d}, \tag{2.15}$$

$$\boldsymbol{\lambda} = \boldsymbol{\Phi}_{21}(t)\boldsymbol{y}(0) + \boldsymbol{\Phi}_{22}(t)\boldsymbol{\lambda}(0) + \boldsymbol{\Psi}_2(t,0)\boldsymbol{d}. \tag{2.16}$$

which can be simplified as:

$$\boldsymbol{y} = \cosh(t)\boldsymbol{y}(0) - \sinh(t)\boldsymbol{\lambda}(0) - (1-\cosh(t))\boldsymbol{d}, \tag{2.17}$$

$$\boldsymbol{\lambda} = -\sinh(t)\boldsymbol{y}(0) + \cosh(t)\boldsymbol{\lambda}(0) - \sinh(t)\boldsymbol{d}. \tag{2.18}$$

Applying the terminal condition $\boldsymbol{\lambda}(t_f) = \mathbf{0}$, we obtain vector $\boldsymbol{\lambda}(0)$ as

$$\boldsymbol{\lambda}(0) = \boldsymbol{\Phi}_{22}^{-1}(t_f)\left[-\boldsymbol{\Phi}_{21}(t_f)\boldsymbol{y}(0) - \boldsymbol{\Psi}_2(t_f,0)\boldsymbol{d}\right]. \tag{2.19}$$

which can be simplified as:

$$\boldsymbol{\lambda}(0) = \cosh(t_f)^{-1}\sinh(t_f)\left[\boldsymbol{y}(0) + \boldsymbol{d}\right]. \tag{2.20}$$

To conclude the *Proof* 2.1, what we need to do is substitute (2.20) by (2.17). After replacing and simplifying the expression, we get the relative distancing policies trajectories y_i as in (2.4). ■

The control actions u_i's and the individual vehicle trajectories x_i's can be calculated from (2.1) and the relative distances given by (2.4).

2.2 Neural Networks

A Neural Network (or Artificial Neural Network) is a beautiful mathematical-based programming model inspired by the human biological brain network. Artificial Intelligence, particularly Deep Learning, makes extensive use of it. Its variants are used to solve typical real-world issues such as image processing with CNN [16], natural language processing with RNN [17], and so on.

An Artificial Neural Network has a structure that is comparable to that of a biological Neural Network. In computer science, an Artificial Neural Network has three layers: input layer, hidden layers, and output layer. Nodes make up the layers of an Artificial Neural Network. Figure 2.2 depicts a nice example of an Artificial Neural Network.

There are one or more nodes in each of the input and output layers. The number of hidden layers in an Artificial Neural Network is limitless, or there are no hidden layers at all. The more hidden layers there are, the more complicated the problem becomes and the Artificial Neural Network becomes tough to calculate and train. As a result, it's critical to think about creating Artificial Neural Network models.

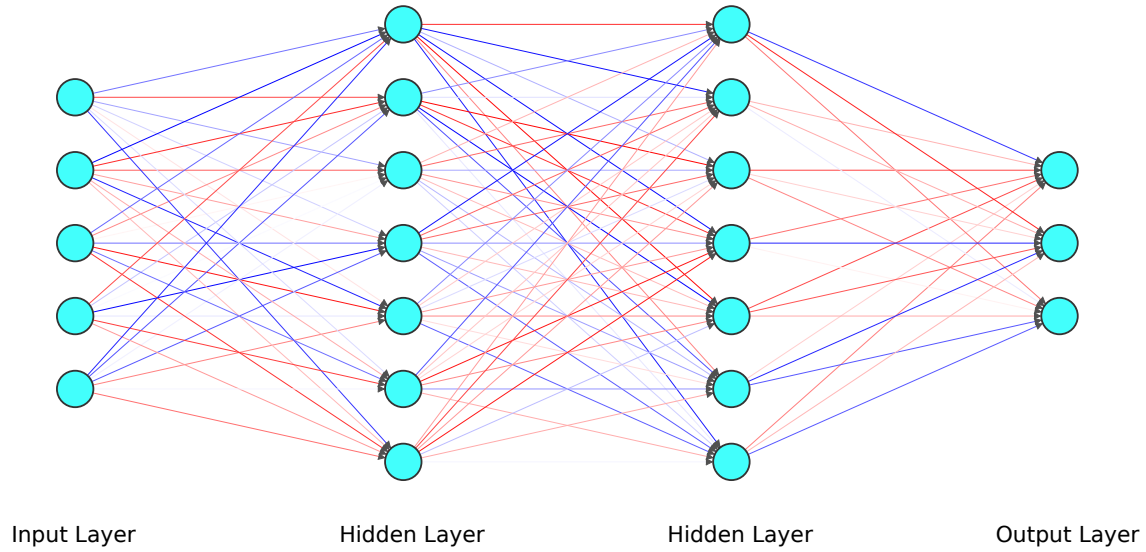

Figure 2.2: An Artificial Neural Network example structure.

Layers are connected to one another via weights. The value for each node is calculated using the input weight group for that node. The activation function is applied to the node values to generate the output value.

Activation functions are non-linear functions because the calculations used before are linear functions resulting in the whole Artificial Neural Network being linear. The activation function is often used as *Sigmoid* function, *tanh* function, *ReLU* function and so on.

After the Artificial Neural Network has calculated all of the nodes, a cost function is required to regulate the computation's cost as efficiently as possible. The optimum Artificial Neural Network is formed by adjusting the weights to minimize the cost value to the lowest feasible level.

Chapter 3

Solving Platooning Problem with Neural Networks

Figure 3.1 illustrates our overall approach to the platooning problem (2.3). First of all, it is established that the cost function in (2.2), along with the dynamic constraint in (2.3) must be determined. Secondly, the aspects of (2.2) and (2.3) are converted into Hamiltonian terms in (2.6). Then, the Pontryagin principle is operated on the Hamiltonian in (2.6). The resulting ordinary differential equations in (2.7),(2.8) and (2.9) are then converted to the state equation in (2.11). Once an initial value problem is determined, the linear time invariant structure is exploited to solve the system. The next step is to build Neural Networks to approximately solve the problem.

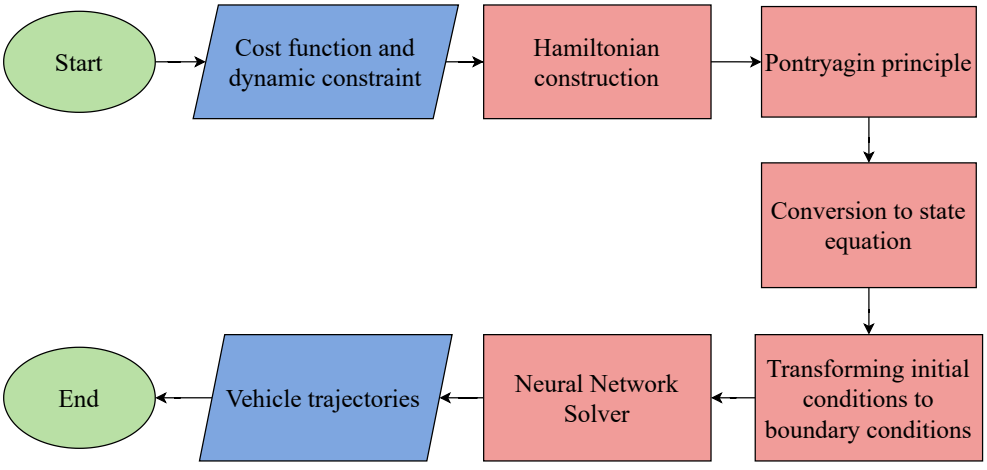

Figure 3.1: Solution steps for the optimal pairwise distances in (2.3).

3.1 Boundary Conditions

Procedures 1-4 have been completed in subsection 2.1 based on the solution steps presented in Figure 3.1. We proceed with the next step in solving the challenge of converting initial conditions to boundary conditions in this subsection. In Artificial Neural Networks, boundary conditions are required for identifying waste solutions.

Because the simulation is based on a genuine issue, we may assume for the Eq. (2.3) that the platoon is continue along the road without interruption.

Additionally, we have the well-known Hamiltonian (2.6). The spacing between vehicles must be optimized as a solution to problem (2.3). It also indicates that the value of the function Hamiltonian (2.6) must be optimized as follows for $t \in [t_0, t_f]$ and all controls admissible:

$$\boldsymbol{H}(\boldsymbol{y}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t) \leq \boldsymbol{H}(\boldsymbol{y}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t), t) \tag{3.1}$$

An optimum control equation (3.1), according to the Pontryagin minimum principle, must optimize the function the Hamiltonian (see [18]). We obtain an optimum argument by demonstrating the requirement of the Pontryagin minimal principle. These values is optimum if the $y(t)$ state, $\lambda(t)$ costate, $u(t)$ control fulfill the following requirements:

$$\begin{cases} \frac{\partial H(y,u,t,\lambda)}{\partial y} = -\dot{\lambda}(t) \\ \\ \frac{\partial H(y,u,t,\lambda)}{\partial \lambda} = \dot{y}(t) \\ \\ \frac{\partial H(y,u,t,\lambda)}{\partial u} = 0 \end{cases} \tag{3.2}$$

The issue may be solved fast and easily using a system of ordinary differential equations to solve (2.3) and the Hamiltonian function. However, in practice, obtaining the formula for ordinary differential equations might be challenging. Only the value of the function at certain values of the independent variables $y(t)$, $\lambda(t)$, and $u(t)$ are of relevance. There are some applications where even actual value is difficult to come by. As a result, utilizing an Artificial Neural Network with an approximation function to compute values within a specific approximation is a reasonable approach.

3.2 Preliminaries of the Neural Network Solver

As we can see in Figure 3.2, one of the major components of an Artificial Neural Network is the perceptron, designed after a living neuron. With the help of perceptrons, it is possible to approximate the nonlinear function to an arbitrary level of precision.

Taking a look at Figure 3.2, we can see that w is the weight vector of the input layer, v represents the output layers weights, while b contains the bias weights.

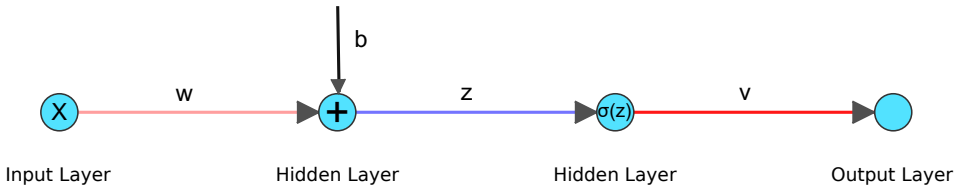

Figure 3.2: Neural Network perceptrons.

Based on these formulation, we arrives at the following result based on the Neural Network:

$$\begin{cases} Output = \sum_{i=1}^{k} v_i \sigma(z_i) \\ \\ z_i = \sum_{i=1}^{k} w_i x + b_i \end{cases} \tag{3.3}$$

where k is number is the number of sigmoid units. The activation function here is the sigmoid function in the following formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

For optimal control problem (2.3), we utilizes Neural Networks for function approximation so that we can approximate the state, co-state, and control function. We discuss this in more detail in the following subsection.

3.3 Neural Network Solver

Our goal in this section is to approximate the proposed equations for the optimal control problem (2.3) using a Neural Network. We interested in three functions: state, costate, and control. Specifically, for each function, we need to build 3 separate Neural Networks: a state Neural Network is called n_y, a costate Neural Network is called n_λ, and a control Neural Network is called n_u. The fine-tuning parameters of each Neural Network is different, as shown in Figure 3.2. In this regard, it is important to note that the structures of Neural Network models must be constructed in order to satisfy the initial or boundary conditions. The Neural Network model can be expressed as follows:

$$\begin{cases} n_y = \sum_{n=1}^{m} v_y^n \sigma(z_y^n), & z_y^n = w_y^n t + b_y^n \\ n_\lambda = \sum_{n=1}^{m} v_\lambda^n \sigma(z^n \lambda), & z_\lambda^n = w_\lambda^n t + b_\lambda^n \\ n_u = \sum_{n=1}^{m} v_u^n \sigma(z_u^n), & z_u^n = w_y^u t + b_u^n \end{cases} \tag{3.5}$$

for $n = 1, \ldots, m$. where m is the number of possible neurons different for each Neural Network. As an example, Figure (3.3) depicts the results when $m = 5$.

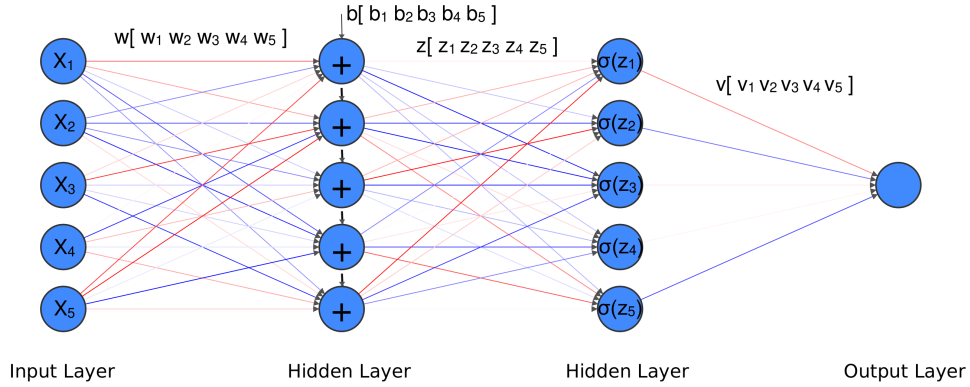

Figure 3.3: Neural Network architecture for $m = 5$.

The 3 Neural Networks are being built now based on expressions (3.5), and the trail has been demodulated. Clearly, the state, cost and control main trail solution that includes Neural Networks need to satisfy the initial and boundary conditions. A solution to the main trail approximation problem is also determined based on the initial and boundary conditions:

$$\begin{cases} y_T = y_0 + t n_y \\ \lambda_T = (t - t_f) n_\lambda \\ u_T = n_u \end{cases} \tag{3.6}$$

Hamiltonian trail solutions can be obtained by substituting the trail solutions into Hamilyonian . With its trail solution, it replaces y, u, λ functions. As a result, the Hamiltonian function contains the weights of the Neural Network. Since the trial solutions (3.6) must satisfy conditions (3.2), they are substituted into the equation (3.2):

$$\begin{cases} \frac{\partial H_T}{\partial y_T} + \dot{\lambda}_T = 0 \\ \\ \frac{\partial H_T}{\partial \lambda_T} - \dot{y}_T = 0 \\ \\ \frac{\partial H_T}{\partial u_T} = 0 \end{cases} \tag{3.7}$$

As a result of this definition [19], we have three error functions corresponding to each equation in order to solve system (3.7) as follows:

$$\begin{cases} E_1(\phi, t) = \left[\frac{\partial H_T}{\partial y_T} + \dot{\lambda}_T\right]^2 \\ \\ E_2(\phi, t) = \left[\frac{\partial H_T}{\partial \lambda_T} - \dot{y}_T\right]^2 \\ \\ E_3(\phi, t) = \left[\frac{\partial H_T}{\partial u_T}\right]^2 \end{cases} \tag{3.8}$$

The input layer weight vector, the bias weight vector, and the output layer weight vector are contained within vector ϕ. Now all that is left for us to do is to differentiate the interval $[t_0, t_f]$ into m points and solve the following unconstrained optimization problem [20]:

$$\min_{\phi} \sum_{i=1}^{m} E_1(\phi, t_i) + E_2(\phi, t_i) + E_3(\phi, t_i) \tag{3.9}$$

As we discussed in our earlier article, it is possible to solve (3.9) with any optimization algorithm. For example, we can use a metaheuristic algorithm such as Genetic Algorithm (GA) or Particle Swarm Optimization algorithm [21], etc.

The optimization results are then used to substitute the value of the ϕ constant in equation (3.6) and determine the components of the state, costate, and control functions.The main advantage to this method is that it is not an extremely complicated algorithm to implement for beginners, and to improve the accuracy of the approximations, we can use a large number of hidden layers or training points within the interval $[t_0, t_f]$ to get more accurate approximations.

Last but not least is that the solution for the state, co-state, control functions is presented as a function of time (t), so that we can compute the outcome at any arbitrary point during the interval $[t_0, t_f]$. It is also worth mentioning that the proposed state and control functions are distinguishable and can be used in applications.

3.4 Metaheuristic Algorithm

In Computer Science and Mathematical Optimization, Metaheuristic is a high-level process. Alternatively, a Metaheuristic can be a Heuristic designed to find, generate, or select a heuristic that provides a good enough optimization solution under conditions of limited computing power. From a very large collection, Metaheuristic selects a subset that can be studied in a different way. Metaheuristic does not have too many assumptions to solve the problem, so it is used for many different problems.

Compared with other optimization algorithms, Metaheuristic does not guarantee that a globally optimal solution is found. However, Metaheuristic approaches the optimization problem in a useful way. Many small simulations applied some form of random optimization yield small solutions. These solutions are dependent on sets of random variables because of the stochastic optimization of the simulation. A large collection of solutions are created. The large collection of solutions gives Metaheuristic an advantage. Metaheuristic does not need too much computational effort to find the optimal solution from that collection.

Many Metaheuristic Method has been published with claims of novelness and practical utility. One of the powerful algorithms that is applied in this study is Self-Organizing Migrating Algorithm with Narrowing Search Space Strategy (iSOMA) is applied in this study.

iSOMA is defined as a algorithm of optimizing a problem by repeated iterations in order to improve the experimental solution. The experimental solution will be tied to a metric to assess whether it tends to be optimal. The iSOMA generates a set of candidates, also known as particles. These particles move in a search space generated by the algorithm. A mathematical formula based on position and velocity evaluates the position of particles. The motion of the particles is affected by the position of the particle whose position is the best and the better position in the search space. One of the remaining particles moves in the direction of the best position it found. The particle with the best position will be updated when the other particles find a better positions. This process is repeated many times to direct the whole swarm to a better position. The final solution is found when the whole swarm is in the best position in the search space. However, there are also cases where the final solution is not found. See iSOMA algorithm at [22, 23].

In this study, Artificial Neural Networks are applied Metaheuristic algorithms such as iSOMA to optimize the weights. The optimal Neural Network weights provide an approximate solution to the platoon control problem base on Hamiltonian trail solutions 3.6.

Chapter 4

Experimental Setup

4.1 Test Function

The Sigmoid function takes a real integer as an input and transforms it to a value between 0 and 1 (see Figure 4.1). The output is asymptotically to 0 if the input is a very tiny negative real number, and asymptotically to 1 if the input is a very big positive real number. The Sigmoid function is frequently utilized because its derivatives are quite attractive. The Sigmoid function also has the benefit of producing a smooth and continuous output.

In this section, we employ the Sigmoid function (3.4) as an activation function for Neural Networks.

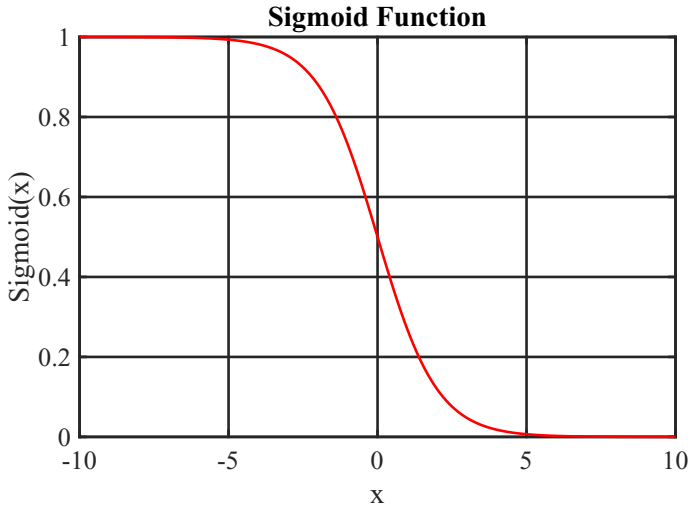

Figure 4.1: Graph of Sigmoid function.

4.2 Comparison Algorithms

The optimization of Artificial Neural Networks is the optimization of expressions (3.9) as well. As the E loss in expression (3.9), gets smaller, the more the Neural Network becomes efficient. For this we can make use of a typical Metaheuristic algorithm such as the iSOMA. To determine whether or not using iSOMA [22, 23] algorithm may be more effective in optimizing these Neural Networks, we intend to apply more existing popular algorithms to evaluate the effectiveness such as the Genetic Algorithm (GA) [24, 25], Differential Evolution (DE) [26, 27], Self-organizing migrating algorithm AllToOne (SOMA ATO) [28, 29], Self-organizing migrating algorithm team to team adaptive (SOMA T3A) [30, 31].

4.3 Parameter Settings

The simulation is based on $n = 3$ vehicles within a the platoon, and the reference vehicle/trajectory consists of three of the actual vehicles in the platoon. Within a finite horizon time under PF topology, $t_f = 5$ is considered. In this simulation, by applying the closed-form solutions given in Theorem 1 with the parameter values given in Table 4.1, the simulation results are illustrated under two different scenarios. Unless otherwise stated in Table 4.1, the positions and spacing policies for CAVs, as well as weighting parameters, are entirely generated at random.

Table 4.1: Simulation model's parameter values under the PF topology.

	Scenario 1		Scenario 2	
i	d_i	$x_i(0)$	d_i	$x_i(0)$
0	-	5.0937	-	4.3064
1	-0.1	4.6469	-0.2	3.2456
2	-0.2	3.8786	-0.1	0.8450
3	-0.2	2.4340	-0.3	0.2151

Considering each vehicle, it is necessary to build three Neural Networks, n_y, n_{lambda} and n_u corresponding to the values we are interested in recognizing as state, state control, and control. The input layer, output layer, and bios vector of each Neural Network have a total of five weights. Overall, we have 9 Neural Networks, which is the number of vehicles $n = 3$. Each of the Neural Networks has 15 weights. The total weight is $n \times 3 \times 3 \times 5 = 135$ weights.

Scenario 1. The weighting parameter values given on the left section of Table 4.1 are generated randomly in range [0.1,1]. In terms of distances between adjacent vehicles, the difference between them is not too great.

Scenario 2. Another experiment with the platoon formation control model's parameter values given on the right section of Table 4.1 is carried out. For this experiment, the range for generating the weighting parameter values is [0,1]. From the value of the initial position, the distance between

the 1^{st} and 2^{nd} vehicle is much larger than the distance between the 2^{nd} and 3^{rd} vehicle or the distance between the 1^{st} vehicle and the vehicle leading the platoon.

Regarding the parameter settings for the algorithms, the dimensions of the algorithms are the same. There are 15 neurons in a network, so dimensionality is defined as the sum of all the weights of all the Neural Networks for a vehicle. When the conditional GA algorithm reaches the limit after a certain amount of time, it stops. In our case, the GA algorithm is stopped after 400 seconds. The following parameters need to be installed for the DE, iSOMA, SOMA ATO, and SOMA T3A algorithms. Maximum function evaluations ($MaxFEs$) were calculated with a dimension as $MaxFEs = 10000 * dimensions$. The control parameter of iSOMA, SOMA ATO, SOMA T3A: $PopSize = 100, Njump = 10, n = 5, m = 10, k = 15, Step = 0.3$. There have been no differences between the control parameter values of the rest algorithms, as they are the same as in the original articles cited.

Chapter 5

Result and Discussion

Throughout this section, we perform a number of simulation experiments to demonstrate the effectiveness of the proposed models and to verify the closed-form solutions under the PF topology in Theorem 1. The differential input values implemented in the proposed platoon formation control is also simulated on the general topology to observe the behavior of the platoon formation. Due to the fact that for general topology case, there are no closed-form solutions (see [11]), the relative distance trajectories of all CAVs can be approximated by Neural Networks utilized in this thesis. Each scenario is built Neural Networks based on the experimental setup outlined in Section 4. Optimization algorithms are applied to optimize the error function value for each vehicle in the platoon. The error function value of the whole platoon is the error function values of all vehicles in the platoon. Each optimization algorithm is run 15 times for each scenario to eliminate unpredictability. One of 15 executions of the algorithms is seen in the graphs below.

Scenario 1. The time histories of the sum of the relative distances trajectories with the CAVs spacing policies are shown in Figure 5.1. On a time history model, we see that each of the time histories approaches zero, so each relative distance satisfies $\lim_{t\to\infty} y_i(t) + d_i = 0$ (note that $d_i < 0$). We are able to observe, from Figure 5.2, that the associated control input to the formation control signal tends to zero by the end of the terminal time. Since their distance at the beginning of the convergence process is not too far away from each other, the convergence speed of the CAVs does not differ much in this scenario.

In Figure 5.3, we can observe the error function value ($Bestfitness$) of the platoon when applying the GA. In the first 30 generations of the algorithm, the error function value of the Neural Networks has a very high value but decreases very quickly from 4500 to 200. From generation 30^{th} to generation 45^{th}, the amplitude of change of error value is not big. After generation 45^{th} until the end of the algorithm, the error function value approaches 0.

The value of the error function while using the DE approach (Figure 5.4) is initially lower than when using the GA algorithm, about 50. The amplitude of $Bestfitness$ falling is seldom consistent,

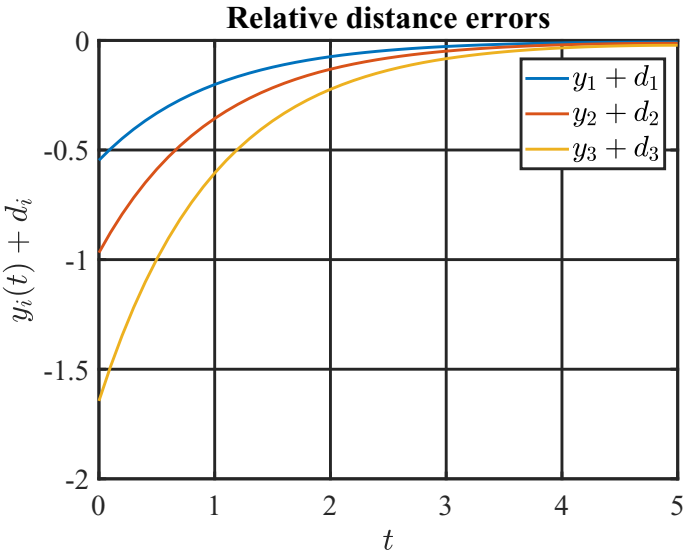

Figure 5.1: Time histories of relative distance errors under the PF topology. The parameter values are set according to Scenario 1 column of Table 4.1.

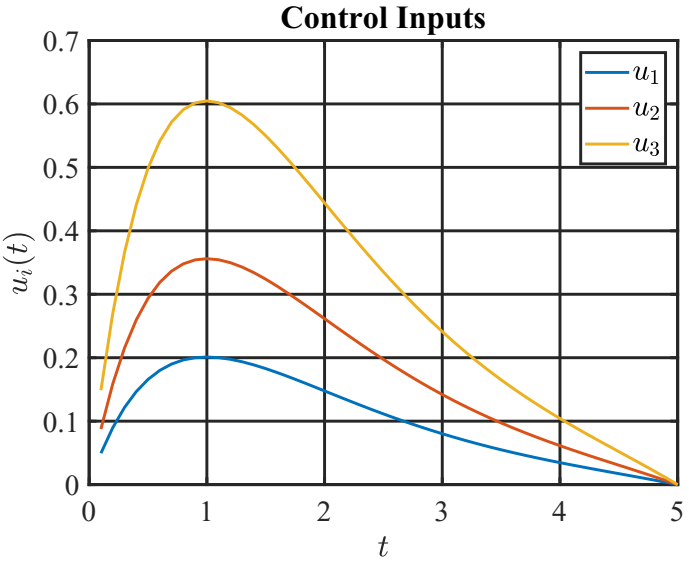

Figure 5.2: Time histories of control input under the PF topology. The parameter values are set according to Scenario 1 column of Table 4.1.

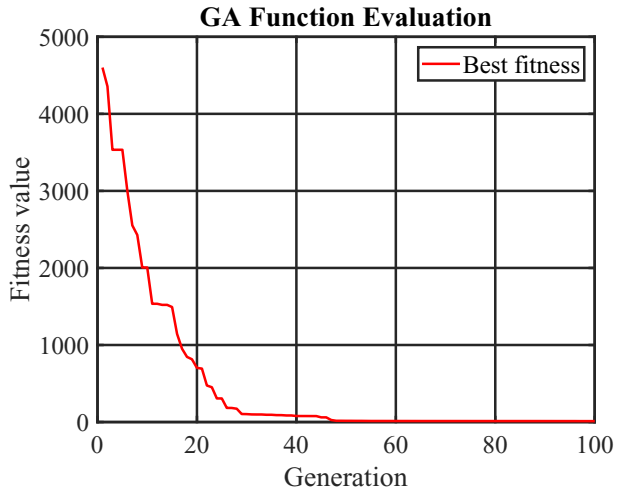

Figure 5.3: Error for estimating E in (3.9) according to Scenario 1 with GA.

but the trend is always downward. In the first six generations, the value of $Best fitness$ dropped by 90%, then dropped steadily as it approached 0, finally ending in the 23^{rd} generation.

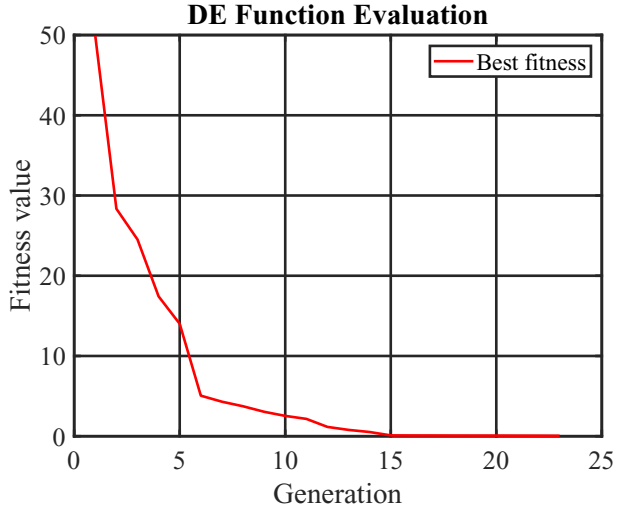

Figure 5.4: Error for estimating E in (3.9) according to Scenario 1 with DE.

The SOMA ATO in Figure 5.5, on the other hand, includes three phase. Unlike the first phase of the DE algorithm, the error function's value reduces from 72 to 15 (79%) throughout the first eight generations. During the 8^{th} to 20^{th} generation, the value of the error function continues to decline by 93%, although at a slower rate. The value of the error function approaches close 0 and finishes at the 42^{th} generation.

SOMA T3A is an advancement over SOMA ATO. The function evaluation is depicted in Figure 5.6. This strategy appears to have worked effectively for the first 10 generations. The optimal

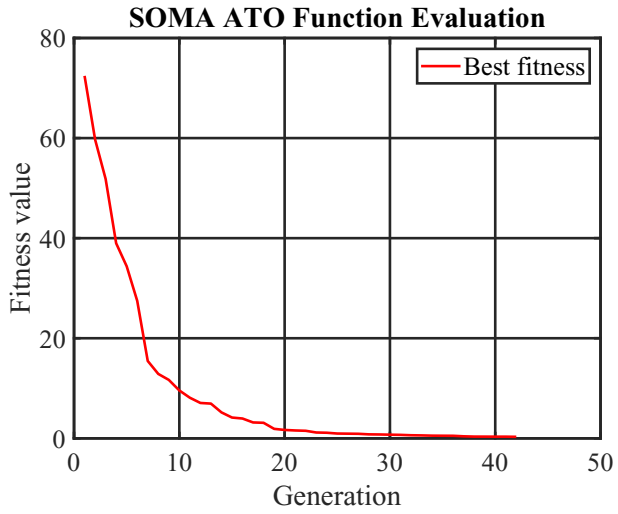

Figure 5.5: Error for estimating E in (3.9) according to Scenario 1 with SOMA ATO.

amplitude of the $Best fitness$ value (89) is rather large. After the $10th$ generation, the optimal speed begins to drop, and by the $30th$ generation, it has saturated.

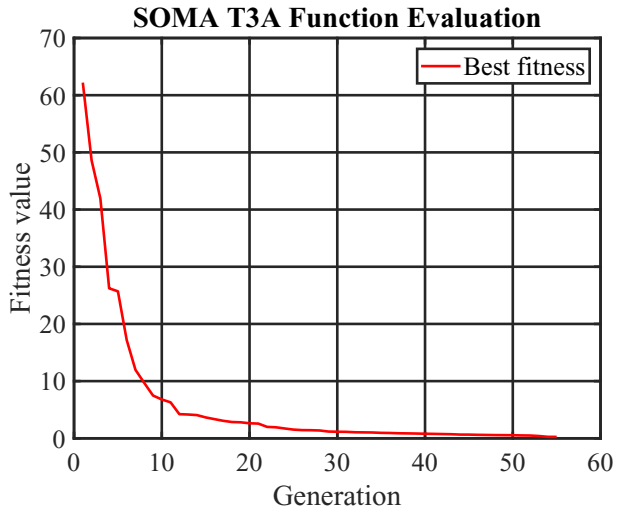

Figure 5.6: Error for estimating E in (3.9) according to Scenario 1 with SOMA T3A.

When evaluating the first 20 generations, the iSOMA algorithm is significantly more powerful than the SOMA T3A. The iSOMA method only decreases the value of $Best fitness$ by roughly 94% when compared to the starting value. The best fitness value was decreased until the $40th$ generation, when it saturated. The iSOMA algorithm generated more than 1100 generations. To make comparisons with different algorithms simpler, Figure 5.7 only shows the first 150 generations.

Scenario 2. The relative distance trajectories of CAVs 1 and CAVs 3 converge quicker, as seen in Figure 5.1. This occurs because CAV 2 initial distance is greater. Figure 5.9 shows that similar

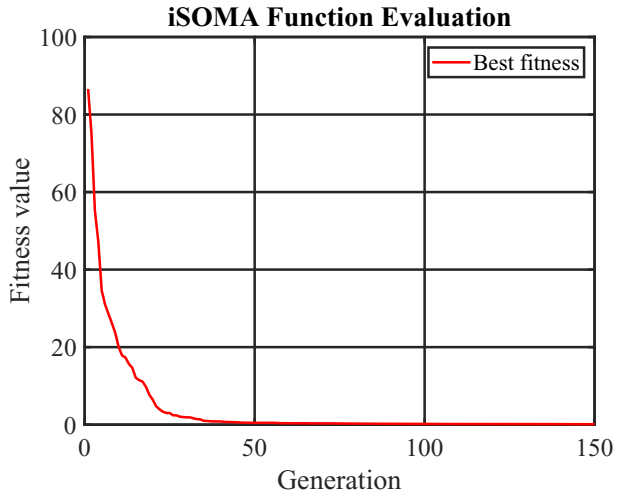

Figure 5.7: Error for estimating E in (3.9) according to Scenario 1 with iSOMA.

observations can be made about their control inputs.

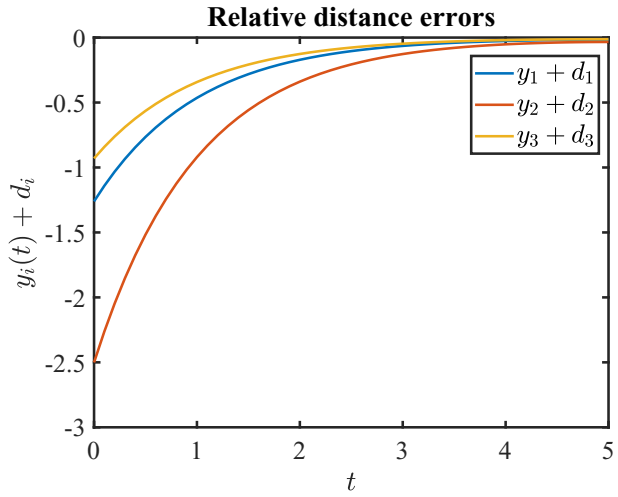

Figure 5.8: Time histories of relative distance errors under the PF topology. The parameter values are set according to Scenario 2 column of Table 4.1.

On Scenario 2, Figure 5.10 illustrates the values of E in (3.9) through generations using Genetic Algorithms (GA). The $Best fitness$ values tend to decline over generations. The value of $Best fitness$ reduces dramatically throughout the first five generations, from 6330 to 1879. The $Best fitness$ value is found in the saturated optimal state. It is deteriorating at a glacial rate.

Unlike the Genetic algorithm, when using the Differential evolution (Figure 5.11), The algorithm ends up with a very small number of generations, 13 generations. The value of fitness has dropped by more than 97% over 13 generations.

When the SOMA ATO algorithm is applied to Scenario 2 in Figure 5.12, the algorithm begins to work well with very high value of error function, approximate 137. From the initial to the 20^{th}

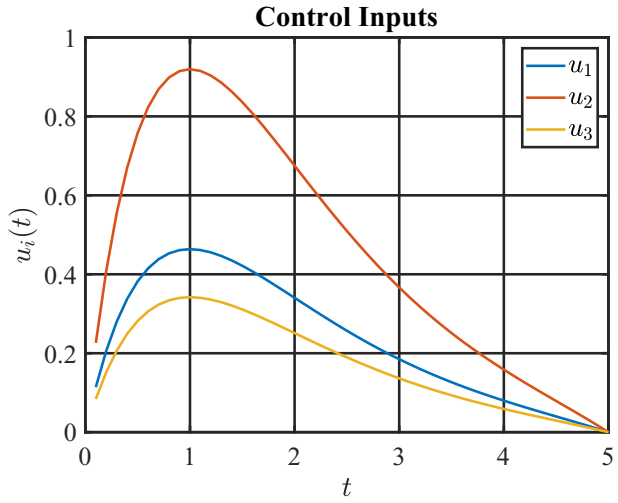

Figure 5.9: Time histories of control input under the PF topology. The parameter values are set according to Scenario 2 column of Table 4.1.

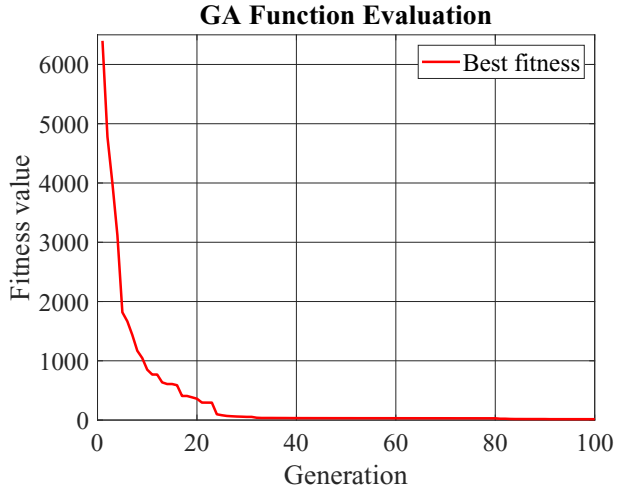

Figure 5.10: Error for estimating E in (3.9) according to Scenario 2 with GA.

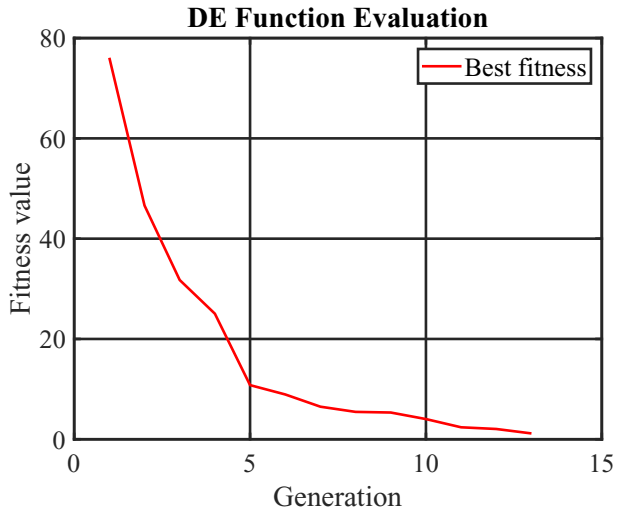

Figure 5.11: Error for estimating E in (3.9) according to Scenario 2 with DE.

generation, the most significant decreasing rate is observed (98%).

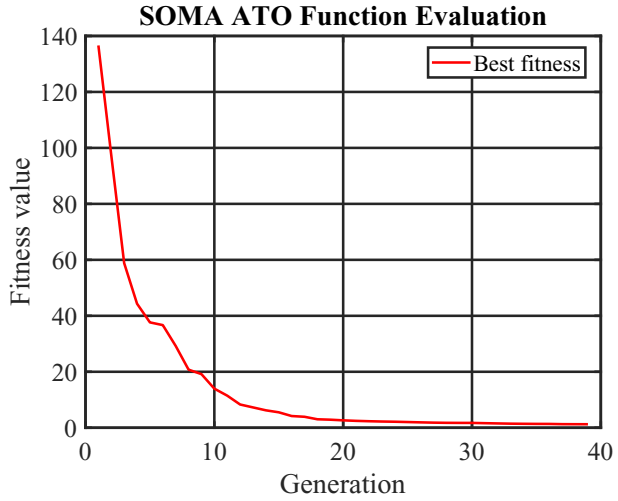

Figure 5.12: Error for estimating E in (3.9) according to Scenario 2 with SOMA ATO.

Scenario 2's SOMA T3A algorithm eliminates data at a faster pace than Scenario 1. This drop rate is estimated to be 97% during the first 20 generation.

The iSOMA algorithm retains the same decreasing rate, as seen in Figure 5.14. The value of Best fitness dropped dramatically from the first to the 30^{th} generation (97%). The algorithm begins nearing the optimal solution around the 30^{th} generation and continues to minimize the value of Best fitness until the method is completed around 1200 generations. After reviewing the algorithms to evaluate the optimal efficiency of the algorithm, and Table 5.1 shows the average value of the algorithm's execution time along with the minimum E value found by the algorithm, the GA algorithm and the algorithm DE is not effective in some cases. The value of error function

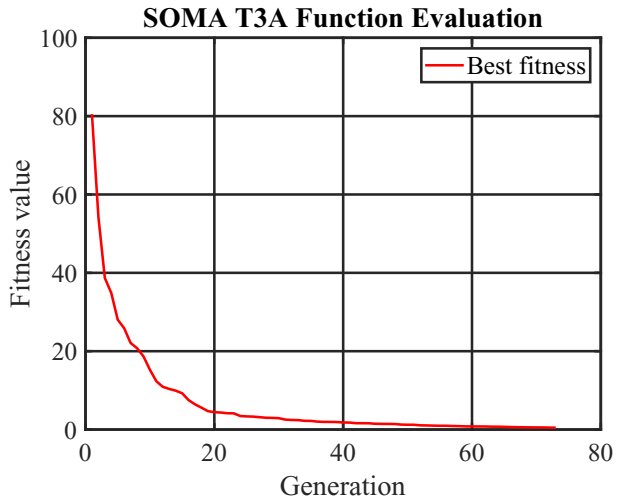

Figure 5.13: Error for estimating E in (3.9) according to Scenario 2 with SOMA T3A.

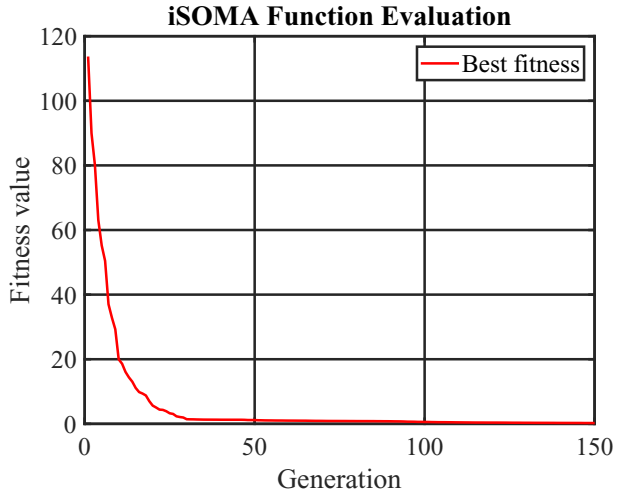

Figure 5.14: Error for estimating E in (3.9) according to Scenario 2 with iSOMA.

Table 5.1: The average value of simulation time and error for Scenario 1 and 2 after executing 15 times.

	GA		DE		SOMA ATO		SOMA T3A		iSOMA	
Scenario	Time	Error	Time	Error	Time	Error	Time	Error	Time	Error
1	186	1.520	92	0.757	**29**	0.443	87	0.386	89	**0.013**
2	186	2.463	90	2.250	**36**	1.321	81	0.698	86	**0.013**

of GA and DE greater than 0,1 is not good. The SOMA ATO algorithm outperforms the DE and GA algorithms. The SOMA ATO algorithm not only employs a smallest time interval than the other algorithm, but also it produces a error that is approximately $65-70\%$ smaller than with GA and DE. The iSOMA algorithm is the one that produces the best results. Despite taking around twice as long as the SOMA T3A algorithm, it produces an astonishingly optimum outcome for the Eq.3.9. When compared to other algorithms, the generated outputs are at least ten times smaller and might be hundreds of times smaller. Figure 5.15, 5.16, 5.17, 5.18 depicts the approximate solution provided by iSOMA for relative distance errors and control input of scenario 1 and 2, respectively. It is impossible to deny that this approximate solution is excellent. The PF topology approximations have relatively modest deviations from the answer.

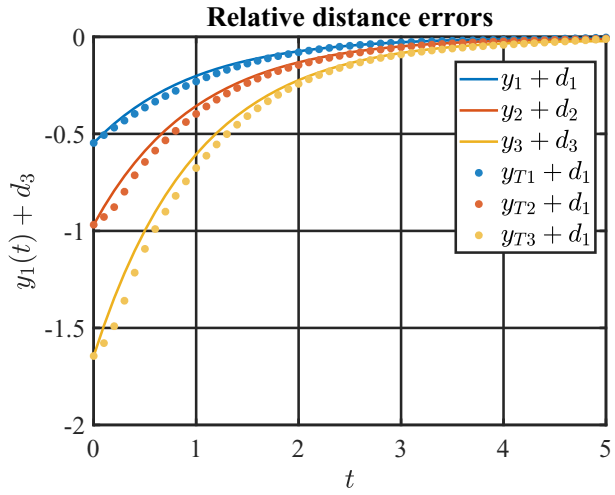

Figure 5.15: Time histories of relative distance errors under the PF topology and approximation solution using Neural Network with iSOMA. The parameter values are set according to Scenario 1 column of Table 4.1.

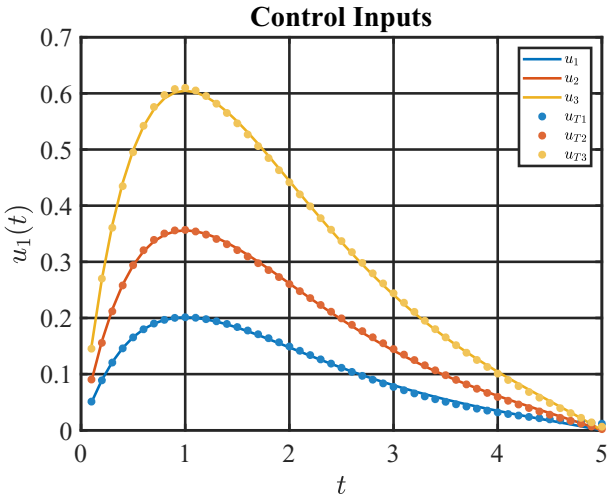

Figure 5.16: Time histories of control input under the PF topology and approximation solution using Neural Network with iSOMA. The parameter values are set according to Scenario 1 column of Table 4.1.

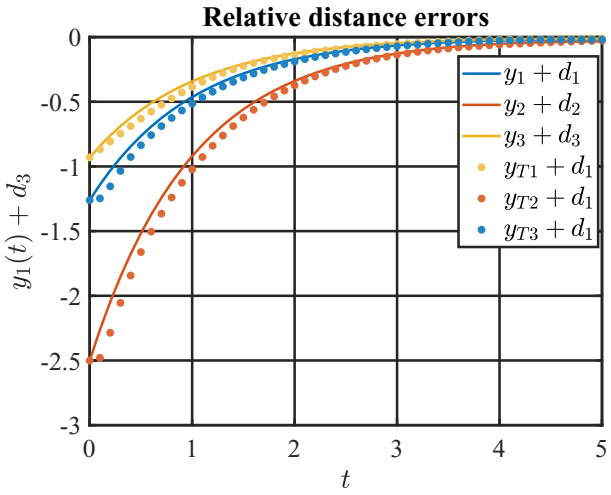

Figure 5.17: Time histories of relative distance errors under the PF topology and approximation solution using Neural Network with iSOMA. The parameter values are set according to Scenario 2 column of Table 4.1.

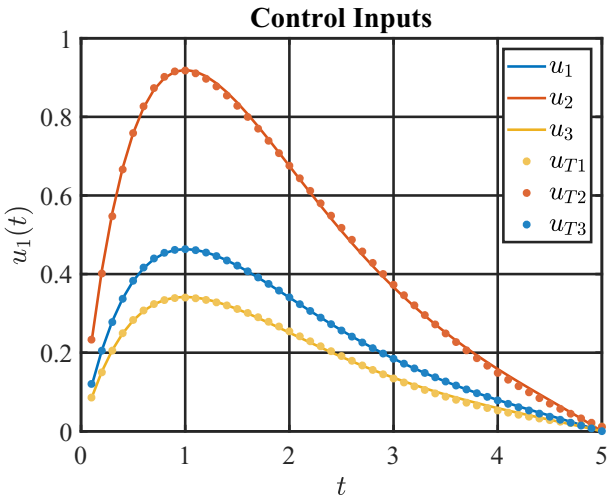

Figure 5.18: Time histories of control input under the PF topology and approximation solution using Neural Network with iSOMA. The parameter values are set according to Scenario 2 column of Table 4.1.

Chapter 6

Conclusion

In this thesis, Neural Network-based function approximation is utilized to find approximate solutions to the platoon control problem. To solve this problem, the Neural Network is applied to approximate 3 functions namely, the individual relative distancing policies trajectories y_i, costate λ_i, and control actions u_i. Each function is built 3 Neural networks to find the approximate solution. Essentially, the solution proposes is a distinguishable function that consists of three components respectively: state, co-state, and control. Heuristic algorithms such as GA, DE, SOMA ATO, SOMA T3A and iSOMA are used to apply the function approximation to optimize the Neural Network weights .The iSOMA algorithm gives the best results when simulating the problem as a platoon control problem. On the other hand, when applied to practical problems such as automatic control of a platoon, there are plenty of unforeseen conditions that may arise while traveling on road. Including these dynamic constraints will make the control problem very difficult to solve. Here, the approximation ability of Neural Networks as well as deep learning methods in general can be used to cope with highly intricate control problems.

Bibliography

1. LI, Qingquan; CHEN, Long; LI, Ming; SHAW, Shih-Lung; NÜCHTER, Andreas. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. IEEE Transactions on Vehicular Technology. 2013, vol. 63, no. 2, pp. 540–555.

2. PEK, Christian; ALTHOFF, Matthias. Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization.
In: 2018 21st International Conference onIntelligent Transportation Systems (ITSC). 2018, pp. 1447–1454.

3. BOJARSKI, Mariusz; DEL TESTA, Davide; DWORAKOWSKI, Daniel; FIRNER, Bernhard; FLEPP, Beat; GOYAL, Prasoon; JACKEL, Lawrence D; MONFORT, Mathew; MULLER, Urs; ZHANG, Jiakai, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316. 2016.

4. WANG, Zhaodong; CHEN, Xiqun; OUYANG, Yanfeng; LI, Meng. Emission mitigation via longitudinal control of intelligent vehicles in a congested platoon.
Computer-Aided Civil and Infrastructure Engineering. 2015, vol. 30, no. 6, pp. 490–506.

5. LARSSON, Erik; SENNTON, Gustav; LARSON, Jeffrey. The vehicle platooning problem: Computational complexity and heuristics. Transportation Research Part C: Emerging Technologies. 2015, vol. 60, pp. 258–277.

6. KIM, Namwook; CHA, Sukwon; PENG, Huei. Optimal control of hybrid electric vehicles based on Pontryagin's minimum principle. IEEE Transactions on control systems technology. 2010, vol. 19, no. 5, pp. 1279–1287.

7. ZHANG, Huaguang; LUO, Yanhong; LIU, Derong. Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints.
IEEE Transactions on Neural Networks. 2009, vol. 20, no. 9, pp. 1490–1503.

8. WANG, Ziran; WU, Guoyuan; BARTH, Matthew J. Developing a distributed consensus-based cooperative adaptive cruise control system for heterogeneous vehicles with predecessor following topology. Journal of Advanced Transportation. 2017, vol. 2017.

9. MYERSON, Roger B. Nash equilibrium and the history of economic theory. Journal of Economic Literature. 1999, vol. 37, no. 3, pp. 1067–1082.

10. YILDIZ, Aykut; JOND, Hossein B. Vehicle Swarm Platooning as Differential Game. In: 2021 20th International Conference on Advanced Robotics (ICAR). 2021, pp. 885–890. Available from DOI: 10.1109/ICAR53236.2021.9659431.

11. JOND, Hossein B.; YILDIZ, Aykut. Connected and Automated Vehicle Platoon Formation Control via Differential Games. arXiv, 2022. Available from DOI: 10.48550/ARXIV.2202.02602.

12. SANZ-SERNA, Jesus-Maria; CALVO, Mari-Paz. Numerical hamiltonian problems. Courier Dover Publications, 2018.

13. LI, Yongfu; CHEN, Bangjie; ZHAO, Hang; PEETA, Srinivas; HU, Simon; WANG, Yibing; ZHENG, Zuduo. A Car-Following Model for Connected and Automated Vehicles With Heterogeneous Time Delays Under Fixed and Switching Communication Topologies. IEEE Transactions on Intelligent Transportation Systems. 2021.

14. CHOI, Byung Chan; KWON, Jaerock; NAM, Haewoon. Image Prediction for Lane Following Assist using Convolutional Neural Network-based U-Net. In: 2022 Inter. Conference on Artificial Intelligence in Information and Communication (ICAIIC). 2022, pp. 078–081.

15. KUHLMAN, Kristopher L. Review of inverse Laplace transform algorithms for Laplace-space numerical approaches. Numerical Algorithms. 2013, vol. 63, no. 2, pp. 339–355.

16. ALBAWI, Saad; MOHAMMED, Tareq Abed; AL-ZAWI, Saad. Understanding of a convolutional neural network. In: 2017 international conference on engineering and technology (ICET). 2017, pp. 1–6.

17. SHERSTINSKY, Alex. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena. 2020, vol. 404, p. 132306.

18. KIRK, Donald E. Optimal control theory: an introduction. Courier Corporation, 2004.

19. EFFATI, Sohrab; PAKDAMAN, Morteza. Optimal control problem via neural networks. Neural Computing and Applications. 2013, vol. 23, no. 7, pp. 2093–2100.

20. DENNIS JR, John E; SCHNABEL, Robert B. Numerical methods for unconstrained optimization and nonlinear equations. SIAM, 1996.

21. WANG, Dongshu; TAN, Dapei; LIU, Lei. Particle swarm optimization algorithm: an overview. Soft Computing. 2018, vol. 22, no. 2, pp. 387–408.

22. YU, Yang; HANDA, Shiro; SASAMORI, Fumihito; TAKYU, Osamu. Improved soft-output M-algorithm for differential encoded LDPC coded systems with multiple-symbol differential detection. In: 2012 IEEE 23rd I.S. on Personal, Indoor and Mobile Radio Communications-(PIMRC). 2012, pp. 1985–1991.

23. DIEP, Quoc Bao; TRUONG, Thanh Cong; DAS, Swagatam; ZELINKA, Ivan. Self-Organizing Migrating Algorithm with narrowing search space strategy for robot path planning. Applied Soft Computing. 2022, vol. 116, p. 108270.

24. DASGUPTA, Kousik; MANDAL, Brototi; DUTTA, Paramartha; MANDAL, Jyotsna Kumar; DAM, Santanu. A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technology. 2013, vol. 10, pp. 340–347.

25. MOZAFAR, Mostafa Rezaei; MORADI, Mohammad H; AMINI, M Hadi. A simultaneous approach for optimal allocation of renewable energy sources and electric vehicle charging stations in smart grids based on improved GA-PSO algorithm. Sustainable cities and society. 2017, vol. 32, pp. 627–637.

26. PRICE, Kenneth V. Differential evolution. In: Handbook of optimization. Springer, 2013, pp. 187–214.

27. DAS, Swagatam; SUGANTHAN, Ponnuthurai Nagaratnam. Differential evolution: A survey of the state-of-the-art. IEEE transactions on evolutionary computation. 2010, vol. 15, no. 1, pp. 4–31.

28. ONWUBOLU, Godfrey C; BABU, BV. New optimization techniques in engineering. Springer, 2013.

29. DAVENDRA, Donald; ZELINKA, Ivan, et al. Self-organizing migrating algorithm. New optimization techniques in engineering. 2016.

30. DIEP, Quoc Bao; ZELINKA, Ivan; DAS, Swagatam. Self-organizing migrating algorithm pareto. In: Mendel. 2019, vol. 25, pp. 111–120. No. 1.

31. DIEP, Quoc Bao. Self-organizing migrating algorithm Team To Team adaptive–SOMA T3A. In: 2019 IEEE Congress on Evolutionary Computation (CEC). 2019, pp. 1182–1187.