MASTER'S DEGREE PROGRAMME

AUTOMOTIVE MECHATRONICS AND MANAGEMENT

AI-based Energy Management Strategies for P2 Plug-in Hybrid Electric Vehicles

SUBMITTED AS A MASTER THESIS

to obtain the academic degree of

Master of Science in Engineering (MSc)

by

Amr Mousa Mohamed Mousa

November 2021

Thesis supervisor

FH-Prof. DI Mag. Dr. techn. Thomas Schlechter MA

DEDICATION

To my beloved parents for their devotion, prayers, love, and unconditional support. I'm tremendously grateful for your unwavering faith in me throughout the whole journey... It is because of you; my life is more worthwhile.

ACKNOWLEDGMENTS

Foremost, I'd like to express my sincere gratitude to my supervisors Prof. Thomas Schlechter and Prof. Stefan Winkler. Your continuous support during my thesis research, your pertinence, enthusiasm, motivation, and immense knowledge were my guidance all the time of research and writing this thesis.

Moreover, I'd like to extend my thanks and gratitude to the AVL DSP team: DI. Patrick Teufelberger, Ing. Huan Chen, Dr. Evgeny Korsunsky and Ing. Bhargav Adabala. Your collaboration and insightful feedback pushed me to sharpen my thinking and brought this work to a higher level.

In addition, I would like to thank my parents for their wise counsel and inspirational drive. I'm glad you are always there for me. To my siblings and my extended family, I'm very grateful for your love and your spiritual support throughout my life.

Finally, I could not have completed this dissertation without the support of my friends, especially Ahmed Hashem, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Thank you all!

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Wels Campus

SWORN DECLARATION

I hereby declare that I prepared this work independently and without help from third parties, that I did not use sources other than the ones referenced and that I have indicated passages taken from those sources.

This thesis was not previously submitted in identical or similar form to any other examination board, nor was it published.

This printed thesis is identical to the electronic version submitted.

.....

Amr Mousa

Wels, November 2021

ABSTRACT

Plug-in Hybrid Electric Vehicles (PHEVs) offer a promising solution for the increasing CO2 emission problem. However, the improved economy of PHEVs strongly depends on the control strategy that decides on the power distribution between the Internal Combustion Engine (ICE) and the electric battery. Traditional rule-based control strategies are no more practical considering the increasing and more complex control objectives introduced by the emerging technologies such as automated driving and connected vehicles. In this study, an advanced Energy Management Strategy (EMS) based on Deterministic Dynamic Programming (DDP) and Reinforcement Learning (RL) is developed.

DDP solves a finite-horizon optimization problem given the driving cycle a priori to obtain a global optimal vehicle power distribution that contributes mostly to the fuel economy improvements. DDP results are used to benchmark the subsequent RL-developed algorithms' performance. In the newly proposed control strategy, an adaptive online learning RL agent is introduced into the existing Hybrid Control Unit (HCU) architecture solving the EMS for near-optimal solutions.

The objective is to minimize the vehicle's expected total fuel consumption with a proper battery depletion rate besides penalizing the frequent engine on/off switching. Several RL-based algorithms have been experimented with using a vehicle model simulation. As a result, an Extended Deep Q-Network (E-DQN) agent is proposed by the thesis, trained on one cycle, and deployed on two other cycles to evaluate the performance. The thesis findings showed that E-DQN outperformed the rule-based strategy achieving up to 10.46% improvement in fuel economy closer to the DP performance alongside providing adequate compliance with the vehicle drivability and driver comfort objectives.

Keywords: Plug-in Hybrid Electric Vehicles, Energy Management Strategy, Deterministic Dynamic Programming, Reinforcement Learning, Machine learning, Deep Neural Networks, Model-based RL.

KURZFASSUNG

Plug-in-Hybrid-Elektrofahrzeuge (PHEV) bieten eine vielversprechende Lösung für das Problem der zunehmenden CO_2-Emissionen. Die verbesserte Wirtschaftlichkeit von PHEVs hängt jedoch stark von der Regelungsstrategie ab, die über die Leistungsverteilung zwischen dem Verbrennungsmotor und der elektrischen Batterie entscheidet. Herkömmliche regelbasierte Steuerungsstrategien sind angesichts der zunehmenden und komplexeren Steuerungsziele, die durch neue Technologien wie automatisiertes Fahren und vernetzte Fahrzeuge eingeführt werden, nicht mehr praktikabel. In dieser Studie wird eine fortschrittliche Energiemanagement-Strategie (EMS) entwickelt, die auf Deterministischer Dynamischer Programmierung (DDP) und Reinforcement Learning (RL) basiert.

DDP löst ein Optimierungsproblem mit endlichem Horizont, bei dem der Fahrzyklus a priori gegeben ist, um eine globale optimale Leistungsverteilung des Fahrzeugs zu erhalten, die hauptsächlich zur Verbesserung des Kraftstoffverbrauchs beiträgt. Die DDP-Ergebnisse werden als Benchmark für die Bewertung der Leistung der anschließend entwickelten RL-Algorithmen verwendet. In der neu vorgeschlagenen Kontrollstrategie wird ein adaptiv online lernender RL-Agent in die bestehende Hybrid-Steuergerät-Architektur integriert. Die mittels DDP ermittelte (optimale) Leistungsverteilung, kann mit dem RL-Agenten nahezu erreicht werden.

Ziel ist es, den erwarteten Gesamtkraftstoffverbrauch des Fahrzeugs mit einer angemessenen Entladungsrate der Batterie zu minimieren und gleichzeitig das häufige Ein- und Ausschalten des Motors zu bestrafen. Mehrere RL-basierte Algorithmen wurden anhand einer Fahrzeugmodellsimulation erprobt. Als Ergebnis wird in dieser Arbeit ein Erweiterter Deep Q-Network (E-DQN) Agent vorgeschlagen, in einem Zyklus trainiert und in zwei weiteren Zyklen eingesetzt, um die Leistung zu bewerten. Die Ergebnisse der Arbeit zeigen, dass E-DQN die regelbasierte Strategie übertrifft und eine Verbesserung des Kraftstoffverbrauchs bis zu 10,46 % gegenüber der DP-Leistung erzielt und gleichzeitig eine angemessene Einhaltung der Fahrbarkeits- und Fahrerkomfortziele des Fahrzeugs gewährleistete.

Schlüsselwörter: Plug-in-Hybrid-Elektrofahrzeuge, Energiemanagement-Strategie, Reinforcement Learning, Maschinelles Lernen, Deep Neural Networks, Modellbasierte RL.

CONTENTS

1 INTRODUCTION

1.1 PROBLEM DEFINITION

By the end of the 19th century, a major technological invention emerged which dramatically changed societies in many countries later in the 20th century. This invention was the vehicle that therefore established the automotive industry. Originally, the automobile was electrically powered, but with the discovery of fossil fuel resources around the world, the internal combustion engine vehicle quickly dominated the vehicles' powertrain architectures.

The automotive industry has been one of the most important and influential sectors since the end of the Second World War, and its worldwide emerging has been spectacular. The global annual car production exceeded 92.78 million units by the end of 2019 [1], and the fleet is estimated to have exceeded 1.32 billion by 2016 in comparison with 0.67 billion in 1996 [2]. The automobile revolutionized transportation, development of the economy, cultural exchanges and has led to massive development of new infrastructures such as roads, highways, and car parking, etc.

Despite the amelioration to the user's mobility and comfort, the automobile has been the subject of significant criticism, especially for its environmental impacts such as the use of non-renewable energies, atmosphere, and noise pollution. The combustion engine operation releases harmful air pollutants such as Carbon Dioxide (CO_2), Carbon Monoxide (CO) and Nitrogen Oxides (NO_x) released into the atmosphere regularly. An EU passenger car releases on average 140 g/km of CO_2 in comparison to 190 g/km of CO_2 in higher polluting fleets such as in the USA as reported in 2010 and shown in Figure 1 [3]. The resultant gases, known as greenhouse gases, heat the climate considerably over years causing global warming. Scientists warn of possible consequences such as rising sea levels, melting glaciers, extreme rain, and heatwaves [4].

For the sake of alleviating global warming and emission problems, governmental legislation is defined to set emission levels for automotive manufacturers. Figure 1 shows a comparison of the global CO_2 regulation standard for passenger cars normalized to the

New European Driving Cycle (NEDC) in gCO_2/km. EU passenger cars are restricted to 95 g/km for 2020 in comparison to 93 g/km for 2025 in the US standard, 117 g/km and 105 g/km by 2020 for China and Japan respectively [5].

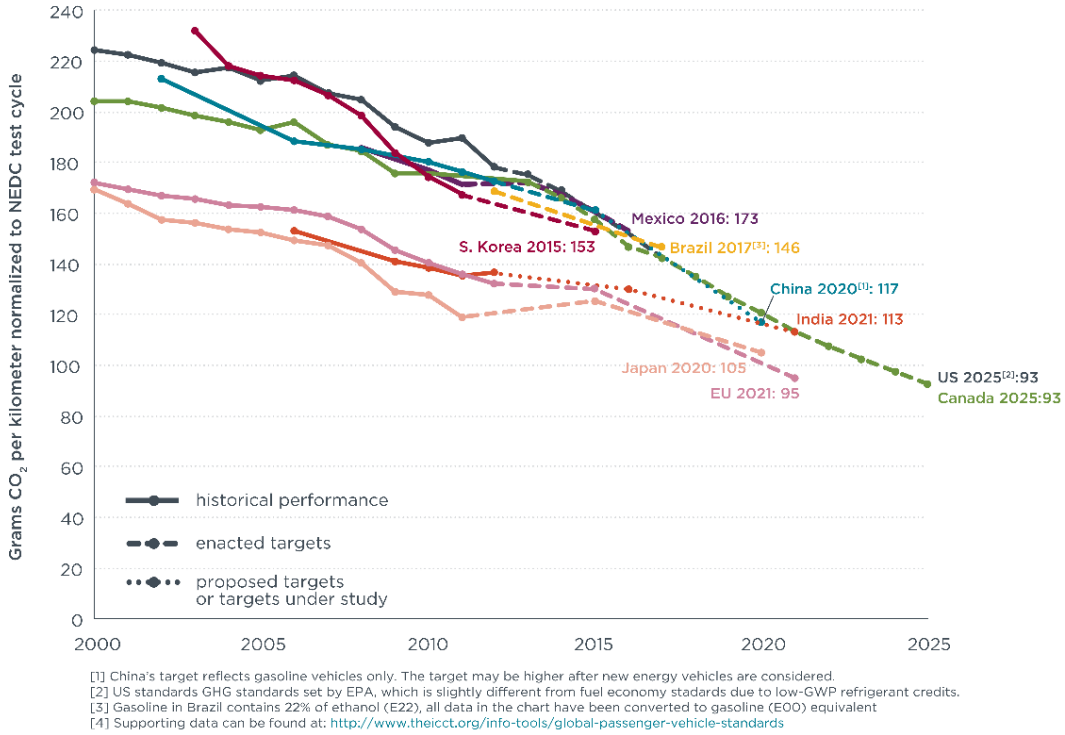


Figure 1 - Passenger car CO_2 emission standard and fuel consumption, normalized to New European Driving Cycle (NEDC) [3]

The automotive industry exerts tremendous efforts to keep up with the emissions standard side by side with balancing customer acceptance and profits simultaneously. Several OEMs invested in improving the ICE technology to minimize fuel consumption and emissions while others develop new vehicles depending on alternative sources of energy such as organic fuels, solar energy, electric battery, and hybrid electric vehicles.

1.2 MOTIVATION

New vehicle concepts were developed such as Electric Vehicle (EV) that is defined as a vehicle propelled by a motor operating exclusively on electric power. On the contrary, Hybrid Electric Vehicle (HEV) combines two or more energy sources, typically fuel and electric energy as considered in this thesis. EVs are promoted by the automotive manufacturers as one of the cleanest and most ecological transport solutions where it could

indeed be an alternative to such urgent pollution problems. EVs have numerous advantages such as unprecedented motor performance and driving experience in addition to minimal operating emissions, noise pollution, and maintenance cost.

Nevertheless, the EV market is not growing as fast as expected due to several drawbacks such as the relatively high cost of the vehicle, and the non-zero carbon footprint associated with battery production and end-of-life vehicle waste management. Customers' acceptance level is highly affected by the EVs limited range and the painfully long charging time, which ranges from 30-40 minutes with DC Fast Chargers (Level 3 charging) for 80% state of charge, up to 60 hours for a full charge of Nissan Leaf from 120 V home socket (Level 1 charging) [6-7].

In the context of the transition towards a sustainable and more ecological energy system, vehicles hybridization is widely considered a promising technique to overcome the drawbacks of EVs with nowadays technology level. Accordingly, HEVs are the subject of significant scientific research and industrial development.

The hybrid vehicles concept goes back to the end of the 19th century when the first hybrid vehicle "Lohner-Porsche Mixte Hybrid " was developed by Ferdinand Porsche in 1898 [8]. Several automotive companies invested in the development of hybrid vehicles till the F3DM PHEV-60 hatchback was released in 2008 by the Chinese manufacturer BYD Auto as the first Plug-In Hybrid Electric Vehicle (PHEV) to be sold in the market [9]. PHEVs, which are the focus of the thesis, are full hybrids equipped with a large capacity battery that can be recharged from the network. They can thus ensure 100% electric propulsion over at least 45 km in All-Electric Range (AER). Compared to full-hybrids, their lithium-ion battery stores 6 to 14 kWh in addition to a charger on board.

Plug-in hybrids offer a significant improvement in CO_2 emissions showing less than 50 g/km (less than 2.2 l/100 km) in the Worldwide Harmonized Light Vehicles Test Procedure (WLTP). This market is mainly occupied currently by high-end brands such as BMW, Audi, or Porsche. The German market alone has 64% of the registered new electric passenger vehicles as PHEVs [10]. The growth of annual registrations of plug-in electric passenger vehicles in Europe between 2011 and 2020 is shown in Figure 2. It

demonstrates that PHEVs contribute to 45% of Europe's electrified passenger vehicles in 2020 in comparison to just 3.3% in 2011.

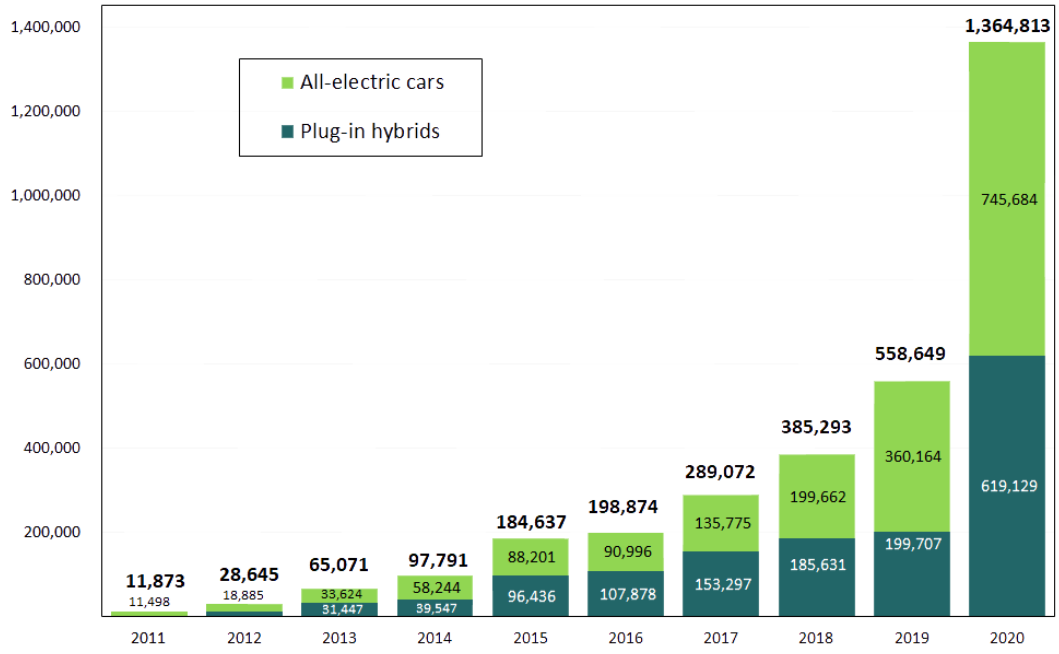


Figure 2 - Growth of annual registrations of plug-in hybrid electric passenger cars in Europe between 2011 and 2020 [11]

In the case of hybrid vehicles, the power distribution is more complex than conventional engine vehicles, in which the power required is supplied only by the engine, because of the necessity of managing two distinct energy sources, fuel, and electrical energy. The battery performs an energy buffer function to increase the overall efficiency of the system through two mechanisms. The first mechanism depends on the controllability of the ICE operating point because the output power of the engine is no longer imposed by the power supplied to the wheels. At any time, it is possible to make the engine operate at an efficient load point where the difference between the power supplied, and the power consumed by the traction is to be supplied by the battery. Since the operation of the electric machine is reversible, it is also possible to recover the kinetic energy of the vehicle during the braking phases and to store it in the battery for later use.

Correct coordination of engine power and electrical power usage reduces energy conversion and transmission losses from the energy source to the wheels, and therefore fuel

consumption and associated emissions are reduced. The key to success is the quality of the on-board energy management strategy which controls the contribution of each source in response to the driver's request through the accelerator pedal. The objective of this strategy shall be to control energy flows in such a way that fuel consumption and emissions are minimized.

Under the assumption of complete prior knowledge of the driver's demand along the trip (driving cycle), the best energy flow distribution between both sources can be determined by applying mathematical theories, typically the Pontryagin Minimum Principle (PMP) and DP [12-13]. These methods are called "offline" while they cannot be used on the onboard system in practice because the reality does not allow such full knowledge of the driving cycle a priori.

Therefore, a real-time energy management strategy is needed that extends the system capabilities to deal with stochastic scenarios and immediately makes the power distribution decision. This type of strategy is called “online”. Online strategies get sub-optimal results compared to optimal results from offline strategies.

Designing an optimal online strategy for real-world driving applications remains a challenge and an open question in research for years. Several real-time strategies have been developed since the 90s emphasizing the complexity of classical control techniques to optimally solve this problem because of the uncertainty and the system dynamics complexity.

1.3 THESIS OBJECTIVE AND CONTRIBUTION

The thesis research is based on P2-PHEV configuration with the existing hybrid control units implemented by the industrial partner, AVL List GmbH. The objective is to investigate and apply advanced online onboard energy management strategy using modern techniques such as machine learning and artificial intelligence. The developed strategy shall overcome the limitations of the offline strategies and provide improvement to the vehicle fuel consumption and emissions level.

The contribution of this thesis is the development of such advanced EMS based on reinforcement learning that enables HEV's real-time control to meet the desired optimization objectives. First, a vehicle model is built, and a global optimization technique is used to derive the optimal EMS for the available driving cycles as input. Then, reinforcement learning-based EMS is introduced to enable real-time control applying the Q-learning algorithm. Several techniques such as Deep Neural Networks (DNNs) and model-based reinforcement learning are presented and experimented with. The proposed methodology combined some of these techniques into an Extended-Deep Q-Network (E-DQN) agent. E-DQN is tested and verified in a model-in-the-loop environment and recommendations for subsequent steps till the vehicle-in-the-loop validation are provided.

1.4 THESIS OUTLINE

The thesis is composed of six chapters: **Chapter 1** describes the thesis background and motivation including an introduction to the EVs, HEVs, and challenges associated with each technology. Finally, the thesis ultimate objectives and contribution are clearly defined. **Chapter 2** discusses the hybrid vehicle technicalities, classification, and different powertrain concepts. Additionally, control-oriented vehicle model implementation and validation are demonstrated as part of the environment model used throughout the energy management strategy development. **Chapter 3** outlines the state-of-the-art energy management strategies in hybrid electric and plug-in hybrid electric vehicles.

Chapter 4 clarifies Global OPtimization techniques (GOP) that are used as a benchmark and criterion for the developed strategies' performance. Moreover, the implementation of the two algorithms, tabular Q-Learning and Deep Q-Network (DQN), while adopting the latter for the rest of the thesis, is discussed. Several improvement techniques for DQN are illustrated and subsequently, several experiments on both algorithms with the corresponding modifications are conducted throughout **Chapter 5** to further increase learning efficiency and improve strategy performance. Simulation analysis and results are outlined using diverse driving cycles for the evaluation of the proposed strategy as well. **Chapter 6** includes a summary and concluding remarks as well as recommendations for further improvements based on the thesis findings for future research.

2 TECHNICAL FRAMEWORK

This chapter explains hybrid vehicles' technicalities, terminologies, and different powertrain concepts in order to introduce the architecture used later in the EMS development process. HCU's software architecture and hybrid modes utilized by the industrial partner AVL List GmbH are discussed to give a deep understanding of the plant to be controlled in the subsequent chapters. Afterward, a control-oriented vehicle model, used later for developing the EMS, is implemented and validated with a high-fidelity model provided by the AVL DSP team.

2.1 HYBRID VEHICLES CLASSIFICATION

2.1.1 HYBRIDIZATION LEVELS

Automotive manufacturers and suppliers often use classification for the hybrid electric vehicles according to the Hybridization Factor (HF), defined as the ratio between the power of the electric machine and the power of the ICE. This type of classification distinguishes between four technological levels/concepts: micro hybrid, mild hybrid, full hybrid, and plug-in hybrid level. The reader is referred to reference [14] for more details.

The plug-in hybrid level has a traditional EMS that favors the use of electric propulsion up to the minimum state of charge of the battery, then switching to ICE propulsion mode. In comparison to the other non-rechargeable hybrid levels (micro, mild and full hybrid), the battery of the plug-in hybrid has much higher energy storage capacity which provides an extended AER. When the battery is nearly empty, it can re-operate as a non-rechargeable hybrid electric vehicle.

2.1.2 POWERTRAIN ARCHITECTURES

The architecture of a hybrid vehicle's powertrain is more complex in comparison to the traditional vehicles. It can be classified into three different powertrain architectures: series, parallel, and power-split (series/parallel). The latter architecture falls within the

thesis scope and the reader is referred to reference [15] for more details regarding the other architectures.

The power-split architecture, shown in Figure 3, makes it possible to achieve a synergy based on the advantages of the series and parallel architectures. The ICE operates at optimum efficiency and can provide propulsion solely, avoiding the accumulation of energy conversion losses. The power-split architecture is more complex with three energy systems with several electrical and mechanical couplings in between, which underlines the need for an adequate energy management strategy to control such complex transmissions efficiently.

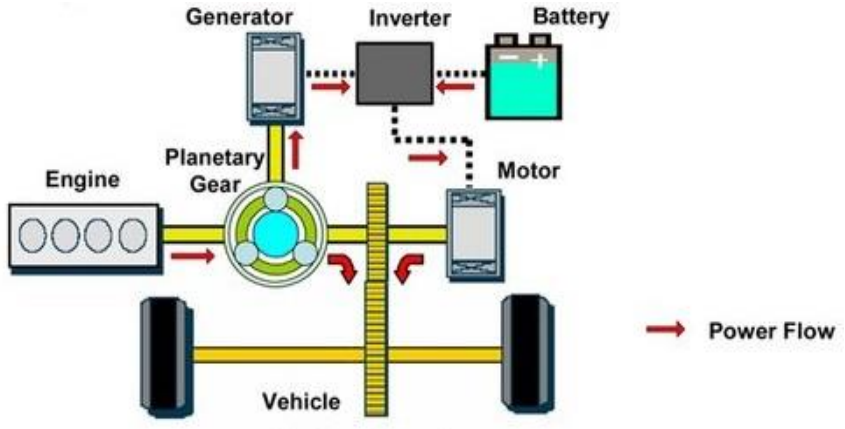


Figure 3 - Hybrid vehicle power-split (parallel/series) configuration [15].

2.1.3 POWERTRAIN CONFIGURATIONS

Classifying HEV topologies based on the location of the EM in the powertrain system is another common classification used by OEMs. As illustrated in Figure 4, HEV configurations range from P0 to P4 topologies.

P0 HEVs feature a Belt Driven Starter/Generator (BSG) that is directly coupled to the ICE while **P1** locates the EM on the crankshaft, known as Integrated Starter/Generator (ISG). **P2**, the configuration used throughout the thesis, locates the EM on the gearbox input, after the clutch. A separation clutch (K0) is an essential clutch in all P0-P4 powertrains. However, a start-up clutch (K1) is an additional clutch only used for P2 HEVs. The P2 configuration offers higher efficiency without ICE drag torque losses by

disconnecting it from the EM which makes pure electric drive possible. **P3** has an EM at the gearbox output and **P4** at the driving axle, connected to wheels all the time [16- 18].

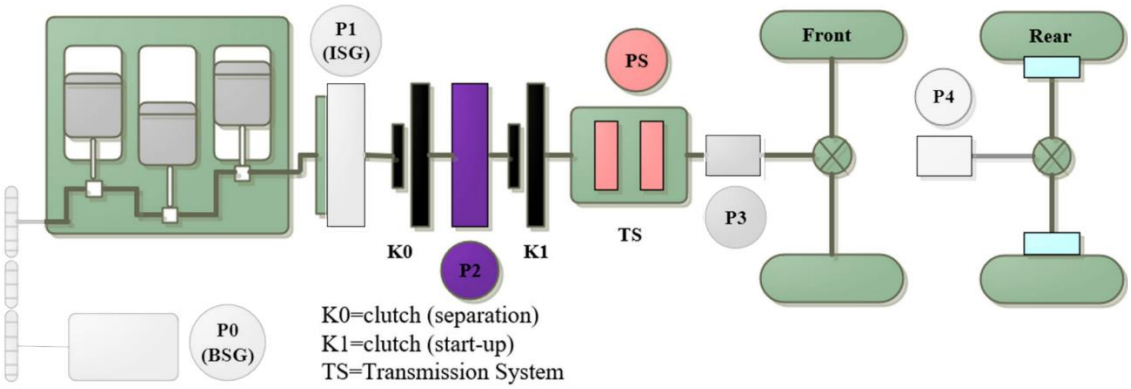


Figure 4 – The configurations of HEVs based on the EM location [17].

2.2 HYBRID CONTROL UNITS

Fuel improvement in HEVs ranges from 10% in mild hybrids up to more than 30% for highly hybridized vehicles [19]. Providing the HCU with sophisticated strategies is needed to realize such potential. As a vital component in the powertrain, HCU works coherently with several vehicle subsystems such as Human-Machine Interface (HMI) with the driver's demands, Transmission Control Unit (TCU), Engine Control Unit (ECU), Battery Management System (BMS), etc. The aforesaid systems send status signals, several limitations, and requests to the HCU as shown in Figure 5. All inputs combined with the driving situation are processed within the HCU and the target "optimum" settings of drive train components ("optimum" hybrid operation mode) are selected.

PHEVs are characterized by having batteries with relatively large capacities. They share the major characteristics with the HCUs of the traditional HEVs but with an additional control layer to provide all-electric driving within the AER. HCU with Energy Storage System (ESS) provides two State of Charge (SoC) change-modes in the long-distance range: Charge-Depletion (CD) and Change-Sustaining (CS). CD mode can be classified as Electric Drive (EltlDrv) mode or blended mode. SoC depletes faster in the EltlDrv mode in comparison to the blended mode because the latter starts ICE frequently in between. Battery operation modes in PHEVs are explained as follows:

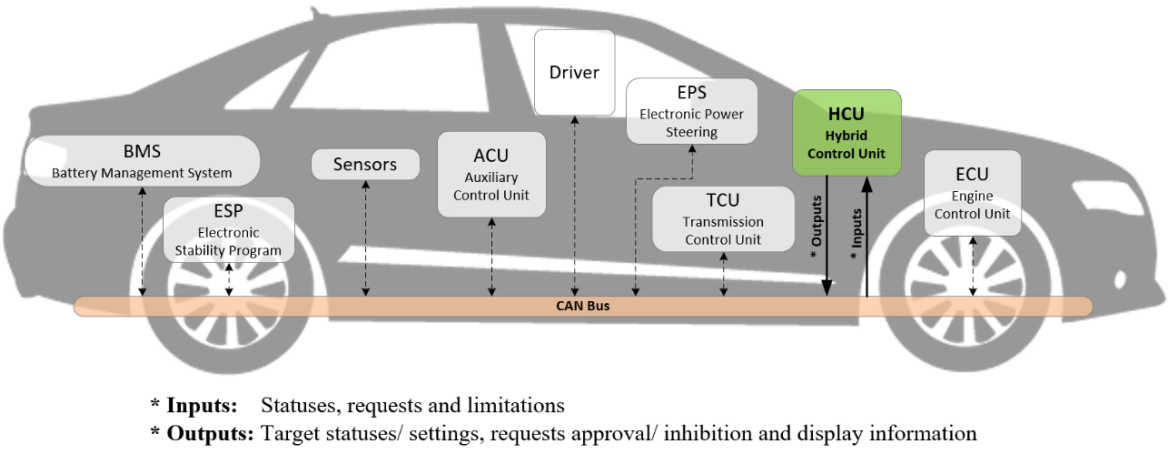


Figure 5 - Various input signals from vehicle components and output signals utilized by HCU, source: HCU software architecture documents, AVL List GmbH.

- **EltlDRV mode within the AER:** Thanks to the PHEVs' large size batteries and EM, the vehicle can be propelled electrically for a specified distance, the AER. This mode is trip distance-dependent and is allowed to be used within the AER where SoC decreases the fastest. The ICE is allowed to start in this mode range if the driver power demand is more than the maximum battery and EM-provided power.
- **CDCS mode:** This mode is the most common strategy in PHEVs when the trip distance is larger than the AER. The vehicle drives in the CD mode (EltlDRV mode) and then shifts to the CS mode when the SoC reaches a certain pre-defined value. During CS mode, vehicle traction power is provided by the ICE or both the ICE and the EM. Consequently, SoC variations in the CS mode are always limited and sustained within a certain narrow range in the long term. The CS mode is used until an external charging process is initiated.
- **Controlled blended mode**: This mode is based on depleting SoC throughout the whole trip by starting the ICE frequently in between to reduce the SoC depletion rate in a controlled fashion.

It is preferred for PHEVs to run out of charge after a driving trip, where there is supposed to be an external charging source to offer a battery charge. In this case, more electricity is utilized, which means more fuel is saved [20].

P2-HCU has a model to generate the quasi-stationary requests for ICE, EM, and drivetrain configuration which are calculated in parallel for each hybrid mode as shown in Figure 6 and as follows:

- **Conventional Drive (ConvDrv):** ICE is the only propulsion source and supplying the low voltage auxiliaries.
- **Additive Boost (AddBoost):** the driver demanded traction torque exceeds the maximum ICE torque under the current engine speed where the EM works as a torque reserve without downshifting the ICE. This mode can be enabled within the AER, where the PEHV initially drives electrically.
- **Substitute Boost (SubBoost):** the demanded torque is relatively high, so part of the torque is going to be offered by the EM. The ICE can work on an optimal efficient operation point without upshifting the load point. This mode does not exist in the actual P2-HCU anymore. Therefore, it is not considered in this work.
- **Optimum Generation (OptmGentn):** The ICE load point is increased to a better fuel-economy location under the same speed. The EM works as a generator for charging the High Voltage (HV) battery with the leftover power from the ICE.
- **Minimum Generation (MinGentn):** a mode designed for battery health by keeping the SoC above a lower threshold. The ICE offers more power than the driver's demand, thus EM works as a generator to charge the HV battery and prevents too deep depletion.
- **Idle Generation (ldleGentn):** the vehicle is standing still and SoC drops below the lower SoC threshold. The ICE runs at the idle speed and charges the battery with the EM in the generation mode and no power is transferred to the wheels.
- **Electric Drive (EltlDrv):** The EM is the only source of propulsion. This mode is the most commonly used in PHEV. The battery offers propulsion energy and

powers the low voltage auxiliaries while the ICE is shut down. Electric creep for low velocities range is included in this mode as well.

- **Recuperation (Recup):** a regenerative breaking mode that uses the generator negative torque to slow down the vehicle. Most of the recuperation energy can be utilized to charge the battery due to the large battery capacity and efficient EM generation mode in P2-PHEV.
- **Stop/Standstill (St/Sndtl):** A vehicle's stationary mode where the vehicle is not moving, and the ICE is turned off. The main functionality is to stop the ICE and enable the EM speed control to activate the transmission oil pump that is used to control the clutches of the Dual Clutch Transmission (DCT).

The P2-HCU hybrid modes are summarized in Table 1 with the different drivetrain component statuses in each mode. The generation modes, OptmGentn, MinGentn, and ldleGentn, use the EM as a generator to charge the HV battery. However, MinGent and IdleGent are only selected when the SoC is below a certain threshold. Therefore, the ICE is not guaranteed to run on the optimal load point and accordingly, they are considered emergent generation modes. On the contrary, OptmGentn is selected based on the energy management strategy, and ICE is operated in the optimal zone.

The boost modes, AddBoost and SubBoost, offer cooperation between the ICE and the EM to meet the torque demand. The difference is that under AddBoost, the ICE works on maximum torque setpoints, while it works in optimal torque setpoints under SubBoost decided by the dynamic control as shown in Figure 6. The P2-PHEV's most frequently selected modes are ConvDrv, EltlDrv, and OptmGentn. Accordingly, the proper and optimal sequence selection between such modes can guarantee optimal fuel economy behavior and minimizes emission levels.

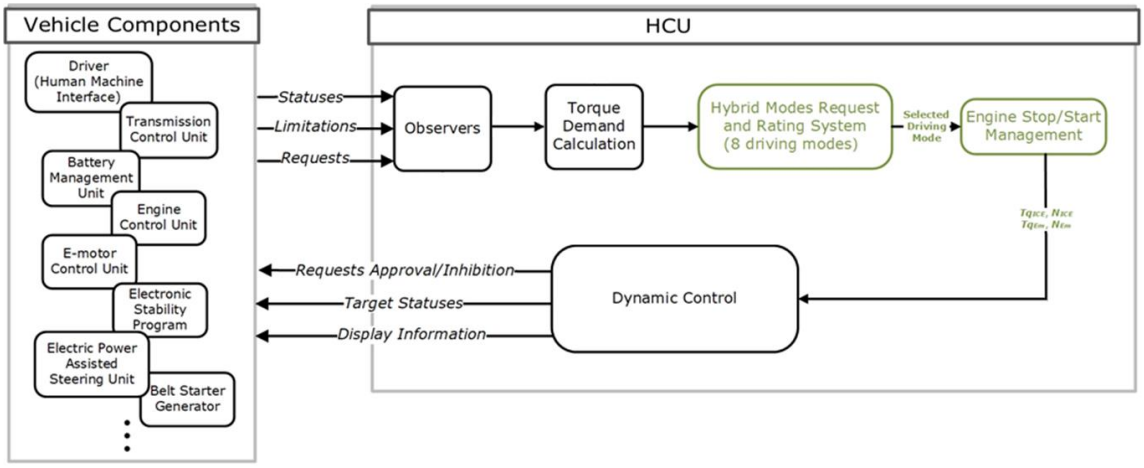


Figure 6 - Hybrid mode requests function in P2-HCU, source: AVL DSP team.

Table 1 - P2-HCU Hybrid modes summary.

	Mode Name	Acr	**Torque status**	**ICE** On RPM_{opt}	On RPM_{Norm}	Off	**EM** Motor	Gen	**HV Battery** Charge	Discharge	**Clutch** Closed	Open
0	**Conventional Drive**	**ConvDrv**	**$T_{req} < T_{ICE\_max}$**	-	✔	-	-	-	-	-	✔	-
1	Additive Boost	AddBoost	T_{req} >>	-	✔	-	✔	-	-	✔	✔	-
2	Substitute Boost	SubBoost	$T_{req} > T_{ICE}/T_{EM}$	✔	-	-	✔	-	-	✔	✔	-
3	**Optimum Generation**	**OptmGentn**	**$T_{req} < T_{ICE}$**	-	✔	-	-	✔	✔	-	✔	-
4	Minimum Generation	MinGentn	$T_{req} < T_{ICE}$	-	✔	-	-	✔	-	-	✔	-
5	Idle Generation	ldleGentn	$T_{req} = 0$	✔	-	-	-	✔	✔	-	-	✔
6	**Electrical Drive**	**EltlDrv**	**$T_{req} < T_{EM}$**	-	-	✔	✔	-	-	✔	-	✔
7	Recuperation	Recup	$T_{req} < 0$	-		✔	-	✔	✔	-	✔	
8	Stop/Standstill	St/Sndtl	$T_{req} = 0$	-	-	✔	-	-	-	✔	-	✔

2.3 VEHICLE MODEL

Developing the hybrid energy management strategy requires having a reliable vehicle dynamics model that represents the plant to be controlled. The AVL DSP department provided a high-fidelity plant model that considers vehicle dynamic characteristics including modeling the vehicle behavior during transient driving such as gear shifting. Such a model is not suitable for agent development and offline training because it brings a huge computational burden and needs complicated numerical solving solutions.

Therefore, a simplified quasi-static vehicle model for P2 PHEV is considered to be sufficient enough to maintain the vehicle physical causality and provide a plant model to develop the agent quickly and efficiently. A later study will be conducted to compare the accuracy of the developed model in energy consumption estimation with the high-fidelity plant model.

2.3.1 MODEL PARAMETERS

The vehicle powertrain component parameters of the investigated P2 PHEV are supplied by the AVL DSP department and listed in Table 2 for the most important parameters. The powertrain architecture is shown in Figure 7 as well.

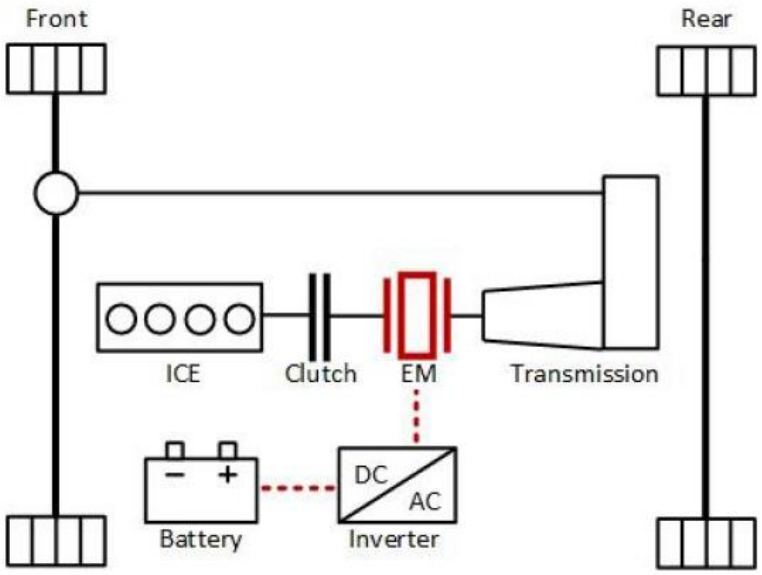


Figure 7 - P2 PHEV powertrain architecture used in the thesis [21].

In P2-PHEV, there are several energy-consuming devices in the vehicle called auxiliary systems which are for vehicle safety or comfort considerations, e.g. lights and infotainment devices. Indeed, the auxiliary power value changes under various driving conditions [22]. Therefore, the auxiliary power consumption is assumed as a constant 500W which is a default setting for other existing P2-HCU simulations.

Table 2 - Component parameters of the P2 PHEV model, source: AVL DSP department.

Component	Parameter	Value
Vehicle	Total mass	2000 kg
	Frontal area	2.35 m^2
ICE	Type	1.2L TGDI Gasoline Engine
	Maximum power	102 kW @ 5500 rpm
EM	Type	Permanent Magnet Synchronous Motor
	Maximum power	94 kW
Battery	Capacity	14.7 kWh
	Nominal voltage	350 V
	Maximum charge/discharge current	450 A
	Useable SoC range	20% - 95%
Transmission	Type	7-speed dual-clutch with gear ratio [16.799 9.444 6.323 4.718 3.498 2.776 2.386]
Misc.	Electrical auxiliary load	500 W

2.3.2 LONGITUDINAL VEHICLE MODEL

The longitudinal vehicle model depends on the dynamics of the vehicle in order to generate forward motion. It is developed to calculate the power and torque demanded by the driver given the operation condition of the vehicle. The free-body diagram of the vehicle is shown in Figure 8.

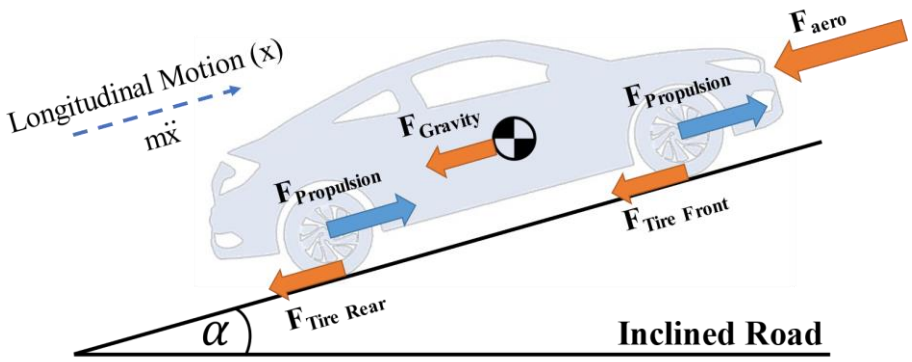


Figure 8 - Vehicle free-body diagram.

Considering the vehicle moves on a road with inclination θ, the power demand P_d depends on the drivetrain internal power loss $P_{int\ loss}$, and external forces F_{ext} as given by equation (2.1):

$$P_d = P_{int\ loss} + (F_{ext} \cdot V) \tag{2.1a}$$

$$= (T_{loss} \cdot \omega) + \left(F_{aerodynamics} + F_{tire} + F_{gravity} + F_{inertia}\right) \cdot V \tag{2.1b}$$

$$= (T_{loss} \cdot \omega) + \left(\frac{1}{2}\rho A C_d V^2 + mg\cos(\theta)C_r + mg\sin(\theta) + ma\right) \cdot V \tag{2.1c}$$

ω is crankshaft rotational speed, ρ is the air density, A is the frontal area, C_d is the aerodynamic drag coefficient, $C_r\ is$ rolling resistance coefficients, m is vehicle mass, θ is the inclination angle, and V is the current vehicle speed. The drivetrain T_{loss} results from the internal mechanical friction losses which depend on the rotational speed, the torque, and the gear selected.

Gear selection and shifting maps are used as a lookup table for the vehicle model simulation. Figure 9 shows the upshift and downshift maps which are considered only for the hybrid mode among others such as Sport and E-drive modes for model simplicity.

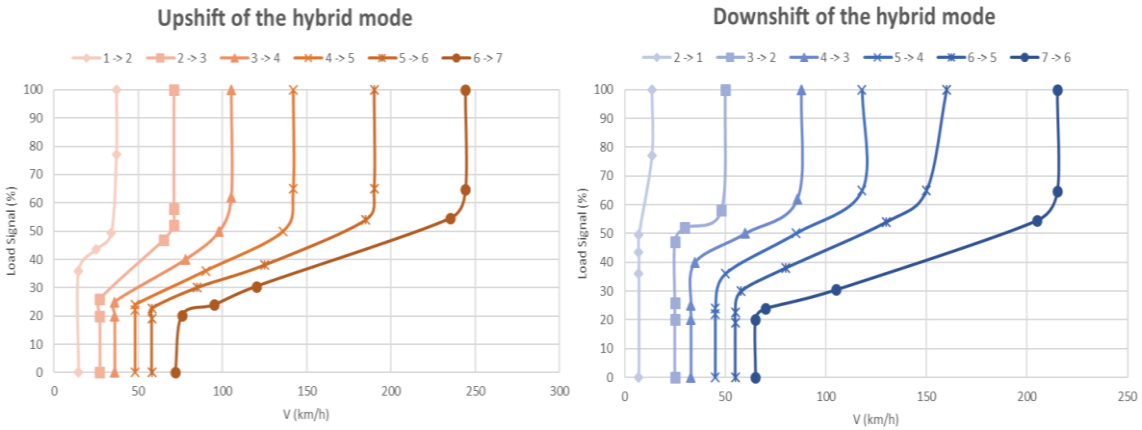


Figure 9 - The gear shift map for the 7-speed dual-clutch transmission considering only the hybrid mode, source: AVL DSP team.

2.3.3 DRIVETRAIN COMPONENT MODELS

Vehicle components are developed based on mathematical models and empirical performance maps. The ICE has a quasi-static fuel consumption model where the engine transients are neglected due to being much faster than the vehicle dynamics. The fuel consumption depends on the engine rotational speed ω_{ICE} and engine torque T_{ICE} as described by equation (2.2) which is plotted as the ICE Brake-Specific Fuel Consumption (BSFC) map in Figure 10.

$$m_{fuel\,ICE} = f(\omega_{ICE}, T_{ICE}) \tag{2.2}$$

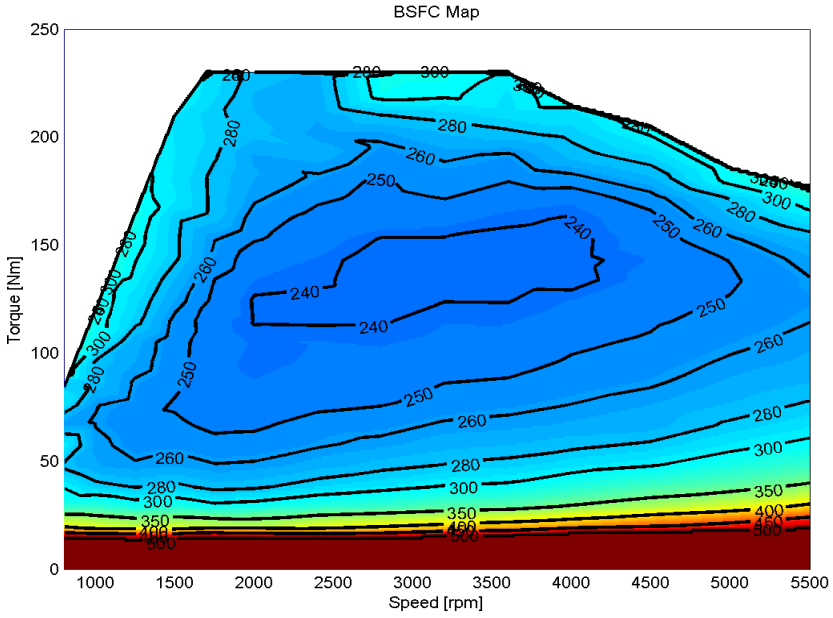


Figure 10 - ICE BSFC map (g/kWh), source: AVL DSP team.

The Electric Machine (EM) model calculates the motor efficiency η_{EM} as a function of the motor rotational speed ω_{EM} and the torque T_{EM} governed by equation (2.3). The motor transient dynamics are assumed to be small, hence neglected. The EM efficiency map for both modes, the eMotor in the positive torque region and the inverter in the negative torque region, is shown in Figure 11.

$$\eta_{EM} = f(\omega_{EM}, T_{EM}) \tag{2.3}$$

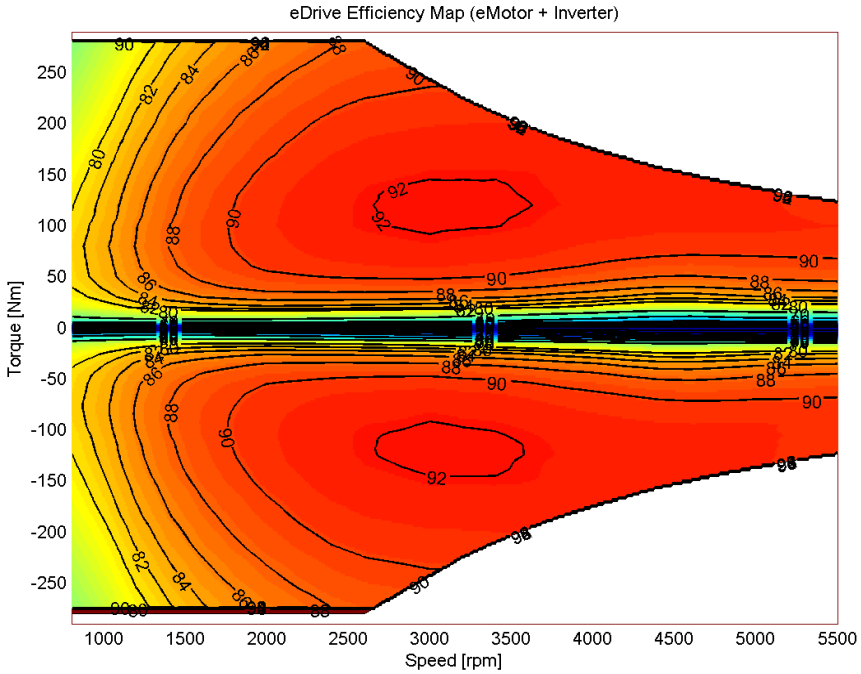


Figure 11 - EM efficiency maps, source: AVL DSP team.

The battery is modeled based on the Thevenin model which is famously applied in hybrid electric vehicles [23]. The Thevenin-based equivalent electric circuit configuration is shown in Figure 12.

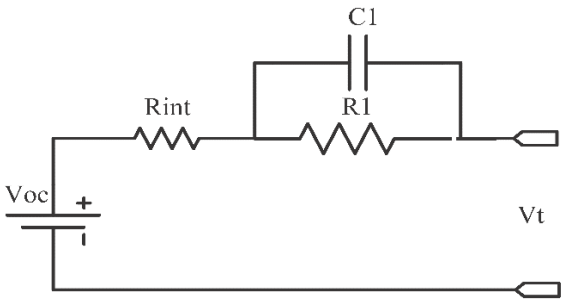


Figure 12 - Battery equivalent electric circuit based on Thevenin's model [23].

The change in the battery's state of charge (SoC) is modeled with equation (2.4).

$$\dot{SOC} = -\frac{1}{Q_{bat}} . \frac{V_{oc} - \sqrt{V_{oc}^2 - 4\, P_{bat}\, R_{bat}}}{2\, R_{bat}} \tag{2.4}$$

The equation parameters are open-circuit voltage V_{oc}, battery internal resistance R_{bat}, battery terminals consumed power P_{bat}, and battery capacitance Q_{bat}. Battery pre-determined maps are used to estimate the open-circuit voltage and internal resistance only at 25 °C for model simplicity as shown in Figure 13. The minimum and maximum battery SoC thresholds are set to be [20%, 95%] for the sake of battery health.

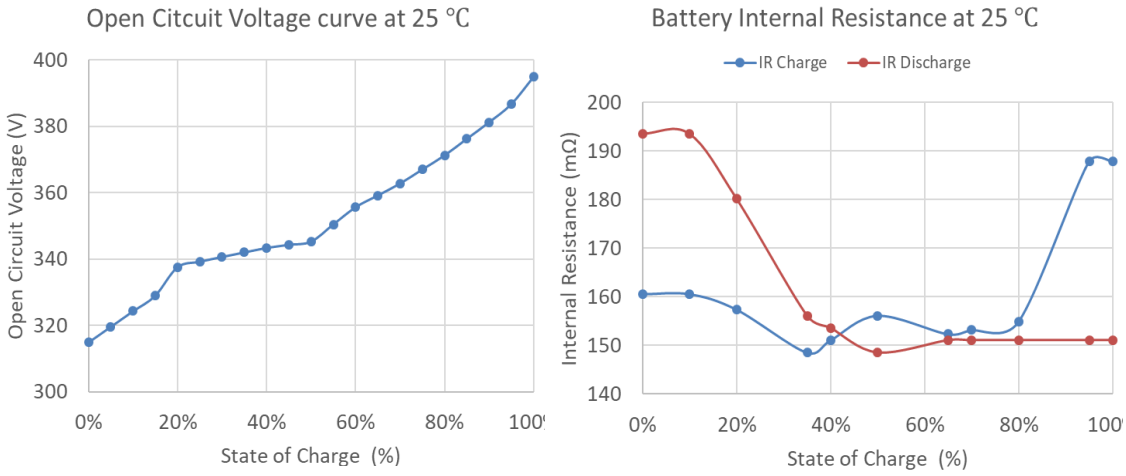


Figure 13 - Powertrain battery characteristics at 25 °C, source: AVL DSP team.

3 STATE OF THE ART EMSS

This chapter focuses on state-of-the-art strategies for solving the energy management problem of the hybrid powertrain. It distinguishes, on one hand, the offline strategies and on the other hand the online strategies, applicable to the real-time control of the P2-PHEVs. It gives the reader an overview of various recent studies related to the EMS problem for the aforementioned vehicle architecture and the thesis problem statement is revisited in section 3.3 taking into consideration the literature review reflections.

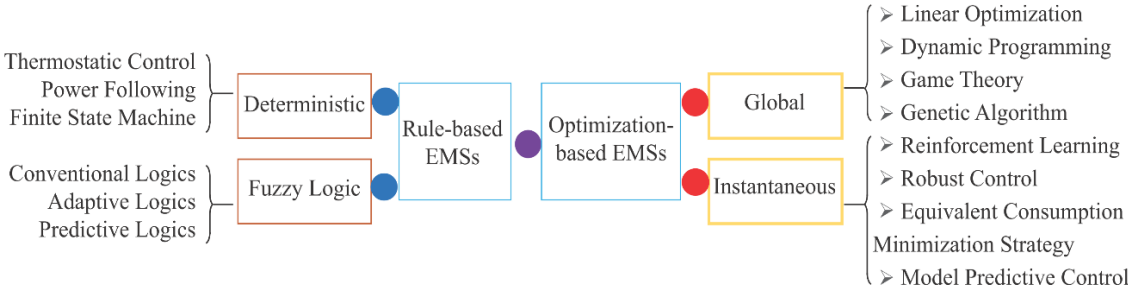


Figure 14 - Classification of the energy management strategies for HEVs [24].

Studies on the design and control of hybrid vehicles began in the 1970s. Many prototypes were made although, the development of energy management strategies was not keeping up with the boom in the hybrid vehicles market. The energy management of the hybrid vehicle has been the subject of many studies in the field of research in recent years. Different strategies have been proposed, analyzed, and compared with each other, mostly by simulation. Few studies have been tested on a Hardware-In-the-Loop (HIL) test bench and few others have been implemented on vehicle prototypes. Figure 14 demonstrates the different classes of energy management strategies and more details are followed in sections 3.1 and 3.2.

3.1 RULES-BASED EMSS

Rules-based strategies, also called heuristic-based strategies, allow rapid implementation without the need to thoroughly study the energy flows in the hybrid system. In general, there is no objective function to minimize, and the control outputs are determined

according to deterministic rules using look-up tables or simple logic. The vehicle can shift between various driving modes according to the current powertrain components state such as SoC, vehicle speed, temperature, and components limitation [20]. In addition to the deterministic rules, fuzzy logic rule-based strategies have more operational freedom. The problem solution is easily understood by human experts for further adjustments and improvements. Moreover, the control law is robust regarding measurement noise and components variability with real-time adaption [25-26].

Rules-based methods are still most used due to their simplicity, low computational power demand, real-time capabilities, and robustness. Their performance depends on the setting of the parameters (often called parameters calibration) which is usually done empirically and then validated by experiment. Obtaining the optimal value of such parameters becomes possible in simulation with the help of optimization algorithms such as genetic algorithms [27]. However, these values strongly depend on the system characteristics. The generalization of a certain strategy to other architectures, or even to other driving scenarios in the same vehicle, is impractical.

The PHEV system is a multi-domain, nonlinear and time-varying system. Accordingly, the calibration of the parameters obtained does not mean the obtained results are optimal. A heuristic approach does not guarantee optimality, but it nevertheless provides avenues for improving the vehicle's energy efficiency.

3.2 OPTIMIZATION-BASED EMSS

Rules-based strategies are simple, often robust, and quick to implement. However, the pursuit of performance optimization and generalization to several architectures encourages researchers to develop new optimization methods derived from mathematical theories, mainly depending on optimal control theory.

The energy management problem can be formulated as an optimal control problem of balancing power distribution of two different energy sources with desired control objectives, physical constraints, and limitations (Figure 15). Control objectives, one or multiple, can be optimizing fuel consumption, emissions, battery State of Health (SoH), SoC, or vehicle operational costs [28].

The state variables, for example, SoC, driver torque demand, velocity, etc., are provided to the powertrain dynamics model in advance where fuel consumption and SoC change (energy sources power output) are calculated. Thus, a high-fidelity model of the powertrain components is required that considers the calculations' non-linearity such as engine BSFC, motor/generator efficiency maps, and battery SoC relation with open-circuit voltage and internal resistance.

The system's physical constraints are the component limitations such as engine speed, torque, and battery charge/discharge rate. Moreover, other constraints such as satisfying power demand, drivability, Noise, Vibration, and Harshness (NVH) might be imposed as additional system requirements.

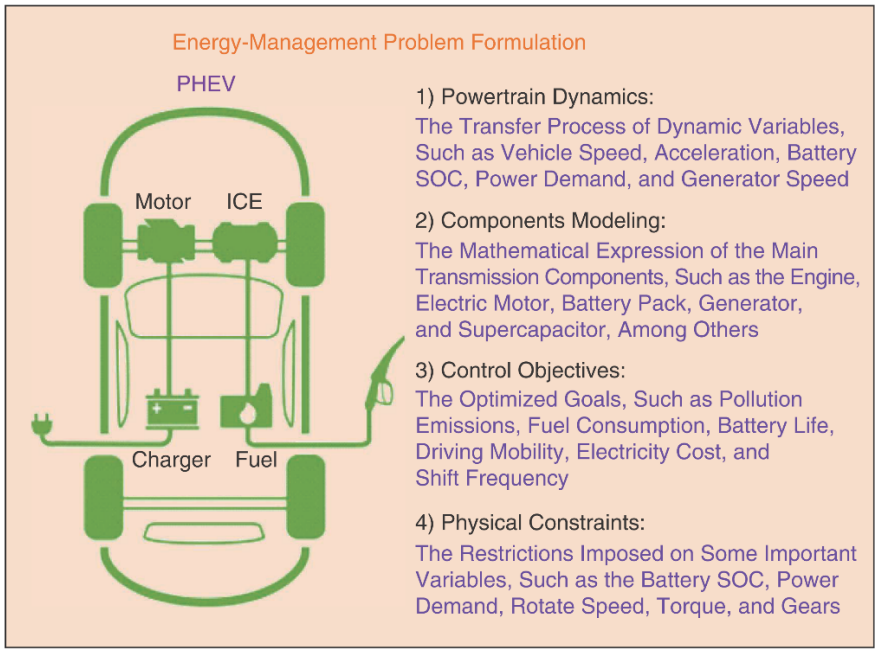


Figure 15 - The formulation of the energy-management problem for PHEVs [28].

3.2.1 GLOBAL OPTIMIZATION-BASED EMSS

3.2.1.1 DYNAMIC PROGRAMMING

One of the most frequently used techniques to solve the energy management problem for a globally optimal solution is Dynamic Programming (DP). It is considered the most efficient general approach for sequential optimization of discrete dynamical systems with an additive cost function under uncertainty [29].

From a mathematical point of view, DP also deals with the stochastic aspects of the problem and it is a proven efficient technique for various problems such as the shortest path problem. It is well suited to solve the problem of optimizing the energy management strategy of the hybrid vehicles, in particular when compliance with conditional constraints is required, as presented by previous studies [13, 15-16, 20, 30-31].

DP has a non-causal property which means it requires a finite horizon with a defined velocity and torque profile in the future in order to obtain global optimality that will be valid only for that specific driving cycle. Therefore, DP is impractical to be implemented as a real-time controller, but the results can be providing theoretical benchmarks for other implementable control algorithms.

3.2.1.2 STOCHASTIC DYNAMIC PROGRAMMING

Another derived algorithm from DP is Stochastic Dynamic Programming (SDP). It follows a well-defined mathematical procedure where generalization to other systems is possible without the need of re-determining calibration parameters. The driving cycle prediction (stochastic variables) can be achieved either by a normal distribution of the vehicle speed, which is more adapted to reality or by the Markov chain considering both the speed and the traction power.

Energy management control algorithms based on SDP are developed by several studies recently. Wang et al. proposed an SDP heuristic approach based on the rules extracted from the results obtained offline by DDP to achieve a near-optimal solution in real-time [32]. Another study formulated the optimization objective as a tradeoff between fuel savings and electrical powertrain stress indicated by incorporating the square of battery charge (C-rate) [33].

3.2.2 INSTANTANEOUS OPTIMIZATION-BASED EMSS

3.2.2.1 EQUIVALENT CONSUMPTION MINIMIZATION STRATEGY (ECMS)

The Equivalent Consumption Minimization Strategy (ECMS) has been proposed by Paganelli in 1999 [34]. The idea of this approach is to convert battery energy to fuel

consumption by multiplying an equivalent factor λ. The usage of such a conversion is assumed because the consumed electric energy in PHEVs is compensated with fuel in the future to charge the battery. The Hamiltonian instantaneous cost function of ECMS, based on the calculus of variations or Pontryagin's Minimum Principle [35], is formulated in equation (3.1). $\lambda(t)$ is the equivalent factor which defines the equivalence between fuel consumption and electricity, while $P_{Bat}(t)$ is the instantaneous battery charge/discharge electric power.

$$\min_{t} f(t) = m_{equiv.\ fuel}(t) = m_{fuel\ ICE}\ (t) + \ \lambda(t)\ \cdot P_{Bat}(t) \qquad (3.1)$$
Subject to: component constraints.

The ECMS approach depends on tuning the $\lambda(t)$ parameter meticulously because its change contributes to battery charge/discharge behavior each time step which directly affects the total fuel consumption. Although the ECMS is direct and intuitive to understand, the major problem with this approach is results sensitivity to the value of λ, which leads to unstable behavior of the strategy. A bad initialization of λ might lead to a significant increase in consumption over the cycle covered in real-time while simply assuming it to be a constant value does not lead to outstanding fuel economy behavior. Based on that, the Adaptive Equivalent Consumption Minimization Strategy (A-ECMS) is introduced with the idea of estimating in real-time the equivalent cost λ making it possible to reduce fuel consumption while ensuring that the state of charge remains within acceptable limits [35-36].

3.2.2.2 MODEL PREDICTIVE CONTROL

Model Predictive Control (MPC) is a commonly used optimal control strategy to solve constrained multi-input multi-output problems. In comparison to other EMSs, the MPC is more far-sighted than the ECMS which helps to gain better sub-optimal results. Additionally, it is simpler in the computational burdensome and does not require predictive information as GOP techniques such as the DP.

The MPC solves a finite-horizon optimization problem for each time instance within a finite receding prediction window. Hence, a control-oriented vehicle model is to be

implemented inside the MPC that is used to optimize the control variable $u(t)$ within the moving horizon window (t+1, t+2, ..., t+ η_{PH}) as shown in Figure 16. The optimization results are η_{PH} control variables $u(t+1)$, $u(t+2)$, …, $u(t+\eta_{PH})$ for the next η_{PH} time segments. However, only the first output control variable $u(t+1)$ is applied and this process is repeated by moving the prediction horizon window one by one step forward.

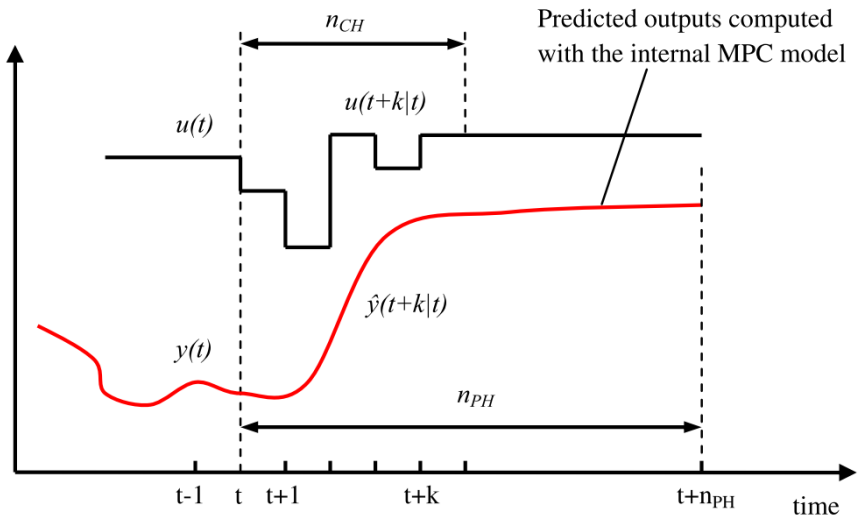


Figure 16 - MPC strategy with prediction and control horizon η_{PH} and η_{CH} respectively [37].

MPC is considered a model-based control strategy that can optimize multiple objectives simultaneously. Accordingly, it has numerous applications in real-world problems. The HEVs control problem, on the contrary, is a nonlinear and constrained problem that requires an MPC based on nonlinear dynamic models and nonlinear solvers to be used. A non-linear MPC is developed in [38] with a promising calculation accuracy, but the high computational resources required make it impractical for real-time implementation on HCUs.

Several studies developed more simplified methods to trade-off between accuracy and real-time capability. Linearizing and discretizing the model is one approach so that the cost function can be reformulated as a quadratic function which can be solved by Quadratic Programming (QP). Guo et al. implemented an MPC-based EMS, with a novel velocity prediction method adopting QP, that achieved improvement in fuel economy with

2.87 ms computational time [39]. Although such simplification has advantages in computational time, further fuel economy improvement is desirable.

Explicit MPC (eMPC) is another practical real-time implementable algorithm. It computes a set of function evaluations offline (optimal policy), that is stored in a state-dependent lookup table. The eMPC does not require an optimization solver in real-time which significantly improves computational time and meets the memory and the computational power of the automotive hardware. A near-optimal eMPC was developed for a power-split Toyota Prius PHEV. The performance is acceptable for limited states, inputs, and constraints. A simplified control-oriented model was implemented although such power-split HEV is much more sophisticated [40].

The aforementioned linear and explicit MPCs fail to achieve satisfying improvement in the fuel economy of HEVs. Consequently, several studies discussed solving a nonlinear model-based MPC. The Continuation/Generalized Minimum RESidual method (C/GMRES) is used in [17] with a 2D dynamics model to solve the MPC receding optimization problem for HEVs while a particle swarm optimization-based nonlinear MPC strategy (PSO-based MPC) is proposed in [41] saving 10% in the fuel consumption in comparison to the CDCS strategy.

In conclusion, the MPC-based energy management strategies offer a feasible solution to real-time HCUs, although a trade-off between accuracy and real-time capability is to be considered. Tuning prediction horizon length and discrete-time sample highly affects the real-time performance of the MPC besides the selected vehicle model fidelity level, the optimization solver, and vehicle hardware capabilities as well [21].

3.2.2.3 REINFORCEMENT LEARNING

Machine learning is a subfield of Artificial Intelligence (AI) that is growing recently by providing state-of-the-art solutions to various problems in many research fields. Reinforcement Learning is an advanced mathematical formulation to control problems utilizing machine learning techniques and algorithms. RL-based energy management approaches in HEVs/ PHEVs took immense attention of several researchers due to their

ability to learn control policies through interaction without being explicitly programmed on a certain strategy.

The reinforcement learning framework embraces an agent that learns how to select the optimal control actions by interacting with the environment that feeds it back with the immediate reward and the updated state. The agent objective is to maximize the cumulative episodic reward given by the environment as illustrated in Figure 17.

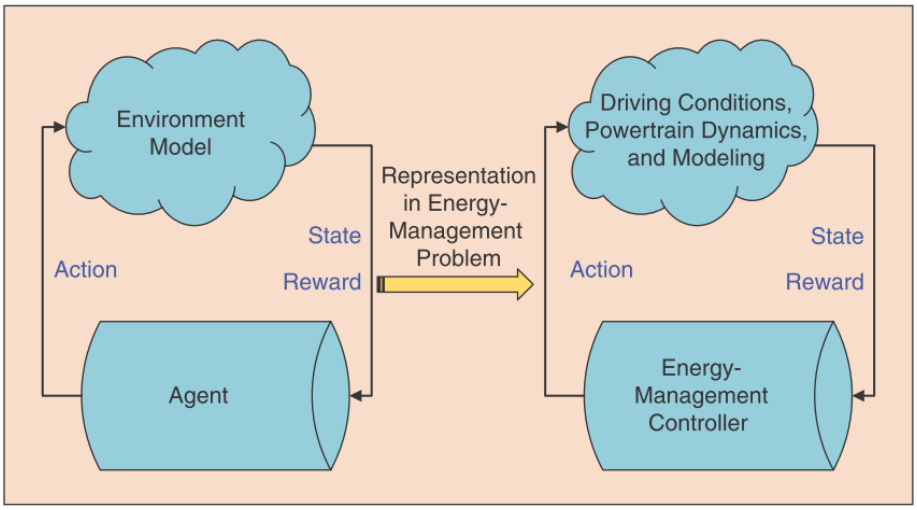


Figure 17 - The RL framework and its representation in the energy-management problem [28].

In HEVs, the agent "energy management controller" uses a trial and error search process to investigate which action leads to the maximum cumulative return in the future. The environment "HEV" serves as a plant in the normal control process by executing the agent control decision and returning the next state and the immediate reward. The environment is represented by the driving conditions, powertrain components modeling, and dynamics. The agent is trained "offline" with a limited number of driving cycles to build a control policy that shall be generalized to other never-seen driving cycles in the real world. When the agent is deployed in the vehicle, it keeps learning from its actions' return and improves the control policy by "online" learning. Further details about RL algorithms implementation and techniques are covered in section 4.2.

Xiaosong, et.al classified RL methods in energy management into two categories, simplex, and hybrid algorithms. The former indicates using a simple tabular algorithm to

derive the EMS control policy such as Q-learning, SARSA, and Dyna while the latter uses a commixture of other information or algorithms such as trip information, predictive algorithms, and deep learning using neural networks [28]. Both simplex and hybrid algorithms include model-free and model-based RL agents. Model-free algorithms, such as Q-learning and SARSA, do not depend on Transition Probability Matrices (TPM), which represents the probability of getting a certain next state given the current state and action, to take the action. On the contrary, model-based RL algorithms, such as DynaQ, do not only learn a control policy through interacting with the environment, but also learn the environment dynamics and use such knowledge to improve the overall learning process, hence the agent performance.

3.2.2.4 RL SIMPLEX ALGORITHMS

Simplex algorithms derive the EMS decision using a simple tabular algorithm of the RL framework. Several researchers began seeking to achieve breakthroughs in the hybrid vehicles EMSs. Considering PHEVs, Qi et al. proposed using a Q-learning algorithm to deplete the battery SOC optimally throughout the cycle. The resulting control policy is to be combined with the charge-sustaining strategy to balance the optimality and real-time performance. The authors claim their strategy can achieve 8% and 12% fuel saving with and without considering charging opportunities respectively [42].

In 2014, Yue et al. utilized a TD(λ)- learning-based model-free algorithm as an EMS for a hybrid vehicle with a supercapacitor and battery. The action space is continuous which controls the current flow into and from both energy reservoirs [43]. The same algorithm was used by Lin et al. to derive an advanced strategy for a thermal HEV using ADvanced VehIcle SimulatOR (ADVISOR). The resulting control strategy was compared with the rule-based strategy using different driving cycles with analyzing both the convergence and the complexity of such an approach [44].

The researchers' contributions in RL were not only limited to innovative algorithms but also novel reward functions. In 2014, Liu et al. defined a new reward signal based on the driver power demand, the battery SOC, and the remaining travel distance obtained from GPS data. The authors proposed using the remaining travel distance as an additional state

to the agent because it is highly correlated to the future energy consumption which the agent tries to learn and optimize continuously. The tabular TD(0) algorithm was utilized to train the estimated Q-table. The results confirmed the agent's self-improving capability, and the performance was very close to the optimal results generated by the DP algorithm [45]. One year later in 2015, Lin et al. proposed a nested RL-based framework to minimize the HEVs operation cost. The agent has an inner loop that focuses on minimizing the fuel cost and an outer loop that works on battery replacement cost minimization. Experimental results showed an operation cost reduction of up to 48% [46].

Other researchers focused on another branch of RL called Inverse Reinforcement Learning (IRL) which labors on learning the agent objectives and reward function from a human expert control policy executed to the environment as illustrated in Figure 18.

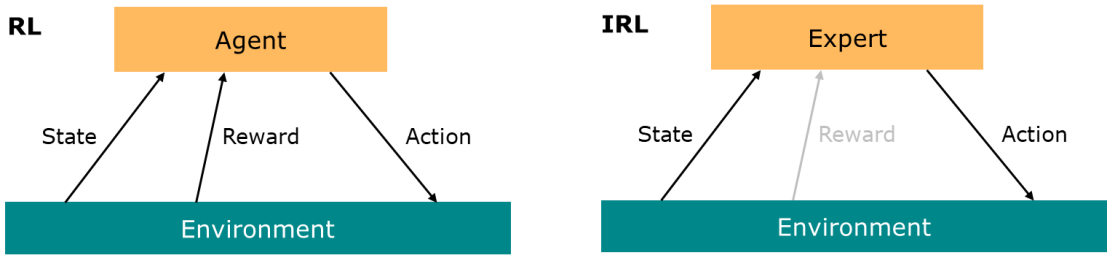


Figure 18 - Reinforcement learning and Inverse Reinforcement learning; source: CS885 Lecture 7, Prof. Pascal Poupart, University of Waterloo.

Vogal et al. used IRL to predict driver behavior by utilizing a probabilistic driving route prediction system and accordingly, the engine and motor power-split ratio is calculated. Results showed an improvement of 1.22% in fuel consumption while the authors believe their approach has high potential to increase vehicle power efficiency without modifying hardware or changing the driver behavior [47].

The model-based Dyna RL algorithm was investigated by Liu et al. in 2015 for a hybrid electric tracked vehicle. The control performance of the Dyna and Q-learning algorithms was compared including fuel cost savings and computational load. Q-learning showed a faster computational time with 43% of the Dyna algorithm but with 1.7% higher fuel

consumption. The Dyna-based fuel consumption was lower than the SDP and very close to the DP results [48].

In summary, simplex algorithms such as Q-learning, Dyna, and TD(λ) showed great potential for online realization in EMS for hybrid and plug-in hybrid electric vehicles. However, such tabular algorithms are not capable of handling stochastic driving situations including driving behavior and road environments. Furthermore, the discretized Q-tables are not a very efficient methodology to represent the control policy because the fine discretization to state-space will lead to exceeding the hardware resources available in HEVs.

3.2.2.5 RL HYBRID ALGORITHMS

The field of artificial intelligence witnessed an immense boost after the rapid development of deep learning and neural networks in recent years. The energy management strategies for HEV/PHEV gained an increasing intelligence with hybrid RL algorithms commonly known as Deep Reinforcement Learning (DRL).

DRL showed great capabilities in designing adaptive energy-management strategies based on historical driving data as reported by Qi et al. in 2017. The agent controls the power-split ratio for a PHEV without depending on any predictions or predefined rules. The experimental results showed that the proposed model saved fuel consumption up to 16.3% in comparison to the conventional binary control strategy [49]. Additionally, Hu et al. developed a DRL-based EMS with MATLAB and ADVISOR simulator previously introduced in [44]. They compared their online learning algorithm with the rule-based strategy which proved to be effective. They showed how such a model can ensure optimality and real-time applicability when trained with sufficient data [50].

Adaptability to different powertrain models and driving situations was discussed in detail by [51] in addition to different drivers' behavior [52], which is proven to be a promising fuel efficiency improvement technique. Actor-critic is another novel technique used in DRL algorithms to represent the agent learning process with continuous action space. Such an algorithm is leveraged by He et al. to overcome the discretization challenge in tabular Q-learning algorithms and tackle the curse of the state variable dimensionality

problem. Stochastic gradient descent is used to train the neural network "Q-value function" and results showed achieving 89% of the fuel economy of DP with unknown driving conditions [53].

The horizon of DRL-based algorithms is not only limited to controlling the engine power or the power-split ratio, but also driving modes and charging/discharging strategies. Recently in 2020, Wang et al. proposed a DRL-based agent for automatic mode-switching of a multimode PHEV, Chevrolet Volt 2016. RL agent is trained offline with the Future Automotive Systems Technology Simulator (FASTSim) to select either charge depletion mode or charge sustaining mode [54]. The authors claim two benefits of such an action space selection; 1) it guarantees the vehicle will run on a highly optimized control action flow and the RL agent will not be required to handle highly dynamical changes and requests inside the HCU. 2) the RL agent will not replace the current HCU control algorithm however, it over-rides the EMS mode selection strategy so as to be engaged or disengaged from the control decision process whenever required. Experimental results showed an improvement in fuel savings from 5.5% up to 6.4% depending on the initial battery SoC.

Future communication technologies in the vehicle industry, incorporating the Internet of Things (IoT) and smart cities, such as Vehicle to X (V2X) including Vehicle to Vehicle (V2V), Vehicle to Grid (V2G), and Vehicle to Cloud (V2C), have leveraged the intelligence potential of DRL approaches in hybrid vehicles EMSs. Hoang et al. broadened the DRL action space to control charging and discharging strategies for PHEVs. The proposed approach depends on the information available from the advanced V2G and V2C communication (Figure 19).

The energy cost problem was formulated by an MDP, and the RL agent decides the online charging and cyber insurance strategies. The energy-management strategy considers the price information and the charging stations load to make an optimal decision such as when to charge/discharge in an online fashion. What is noticeable in this approach is that the EMS problem is extended from a single to multiple vehicles in one connected environment. Results are verified through simulations and showed that cyber insurance is able to maximize the revenue of PHEV users and deal with cyber risks efficiently [55].

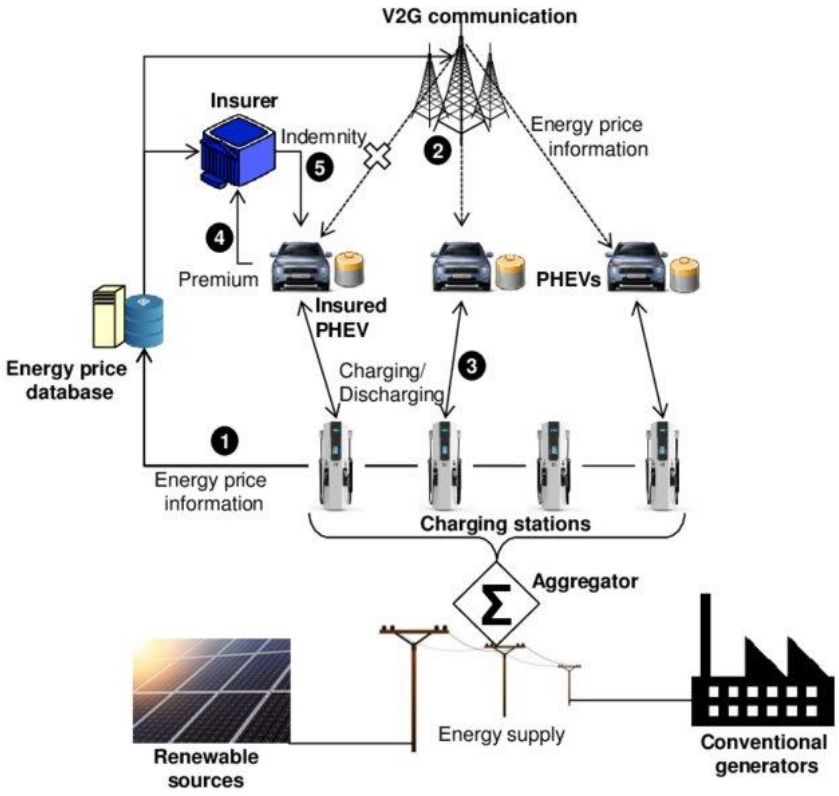


Figure 19 - Cyber insurance for PEVs charging/discharging processes [55].

More advanced hybridization in RL algorithms considering Connected and Automated Vehicles (CAVs) was introduced by Zhu e al. in 2021. CAVs with multiple power sources exhibit promising fuel savings through designing the optimal speed and power depletion by the virtue of connectivity look-ahead information and mapping features. This complex problem was formulated as a Partially Observable Markov Decision Process (POMDP) solved by the DRL actor-critic algorithm. Compared to a baseline controller, the proposed agent was able to save 17.4% more fuel by modulating vehicle velocity throughout the trip route and performing an energy-efficient approach/departure at signalized intersections [56].

Ultimately, migrating the rule-based energy management strategies for the majority of the HEVs and PHEVs in the market is a necessity in the era of big data and connectivity. DRL-based energy-management strategies showed superior performance in the scope of diverse problems which designates to where the future research and development for automotive OEMs should be directed. Several challenges for DRL hybrid algorithms shall

be taken into consideration, such as the onboard control unit computational power and storage resources. The former challenge is the need for an extra computer to be installed in the vehicle to process the data and train the control policy in parallel with the vehicle operation. The latter is collecting the data and storage since DRL needs a large amount of data to be able to generalize and adapt the derived control strategy for different driving situations. Network communication and Intelligent Transportation Systems (ITSs) will empower the DRL-based energy-management strategies to be applied in real-time in the future while overcoming the current stumbling obstacles and challenges.

3.3 FURTHER PROBLEM STATEMENT

The ultimate research objective as proposed in section 1.3 is to improve PHEV's overall fuel economy behavior by implementing an advanced energy management strategy that controls the power distribution decision. The developed method should be compatible with the existing P2-HCU. The trip distance is longer than the AER, which is the precondition of this thesis where optimization control strategies are advantageous.

The energy management strategies adopted in this thesis can be classified into offline and online strategies. Prior knowledge of the whole trip is a requirement for the offline GOP techniques such as DDP which invalidates the real-time applicability into the HCU. However, the performance results from the GOP are considered as benchmarks for the online simulation results, which reveal the potential of online strategies. DDP is used in this research to better understand the optimal behavior in each driving condition and benchmark the online controller performance.

The driving mode decision, selected by the P2-HCU, is a discrete optimization problem and accordingly, this research creates an optimization-based control strategy for the P2-HCU. RL models are proposed, which were shown to be promising advanced optimal-close methods for the HEV’s control problem. In the meantime, vehicle drivability and driver comfort objectives were considered in the online RL agent instead of being sacrificed too much for better fuel economy. Real-time capabilities, drive-train constraints, hardware resources utilization, and performance are discussed in Chapter 4 and compared experimentally in Chapter 5.

This thesis aims to provide a framework for further research efforts utilizing RL in HEV's control problem. To sum up, the thesis includes the subsequent key steps:

- DDP-based solver implementation, offline global optimization calculation, and analysis in Section 4.1.
- Tabular RL Q-learning algorithms design and implementation in Section 4.2.1
- Deep Q-learning techniques further investigation and integration into the existing P2-HCU control scheme in Section 4.2.2.
- Simulation and results analysis in Chapter 5.
- Concluding remarks, prospects and recommendations are in Chapter 6.

4 METHODOLOGY

This chapter deals with the underlying methods used for the rest of the thesis. The chapter includes two main sections. Section 4.1 describes the DDP approach revealing the benchmarking used to evaluate the subsequent developed RL-based EMSs. Section 4.2 demonstrates Reinforcement Learning algorithms and their improvement techniques while experimental details and results are presented in chapter 5.

4.1 DYNAMIC PROGRAMMING

Section 3.2.1 revealed that GOP is a promising solution to the HEV control problem due to its ability to guarantee optimality. It requires the knowledge of the future driving cycle ahead to compute the sequence of optimal control inputs applied to the HEV in order to meet the optimization objective. However, GOP techniques are impractical in the context of HEVs due to several reasons:

a. The non-causal characteristic, because accurate future driving information is not realistically available even with predictive information of the driver behavior and navigation.

b. The mathematical problem formulation is stochastic, not deterministic, which requires building a transition probability matrix for the future vehicle state prediction.

c. Several model simplifications and assumptions are implemented to avoid the huge computational burdens associated with the optimization problem solutions.

Accordingly, a Deterministic Dynamic Programming-based solver (DDP-based) is developed which can deliver a near-optimal solution that is close to the best theoretical power distribution. The following section discusses the design of the DDP control strategy for P2-PHEV which is developed to run offline within MATLAB to offer a benchmark for the online control strategies to be developed.

4.1.1 DDP-BASED SOLVER FORMULATION

DP is a numerical technique that applies to any kind of problem that can be solved at sequenced stages (divide-and-conquer approach). DP major components are recursion which is used to solve the subproblems recursively, and memoization that is used to store the computed values in a table for caching purposes.[13]

DP approaches extend the divide-and-conquer approach with two techniques, top-down (memoization) and bottom-up (tabulation). Both techniques cache and reuse the subproblems' solutions which help in improving the program performance dramatically.

- Top-down (memoization): it's a similar approach to recursion as it looks for the value in the table first which acts as a cache before computing solutions. If the problem solution is found, then the result will be taken from the cache. Otherwise, the problem will be solved, and the results are likely to be stored in the table for future use.
- Bottom-up (tabulation): the opposite approach of the top-down which avoids recursion. In this approach, the problem is solved in a bottom-up manner by solving all the related subproblems first. It is typically done by filling up all entries in the table. Based on the results in the table, the solution to the original problem is computed.

The PHEV problem can be expressed mathematically in a discrete-time space in equation (4.1) as follows:

$$s_{k+1} = f(s_k, a_k), \qquad k = 0,1, \dots n-1 \tag{4.1}$$

$$min \quad J = J(s_n) + \sum_{k=0}^{n-1} J(s_k, a_k) \tag{4.2}$$

s_{k+1} is the system state at time k that ranges from 0 to the driving cycle duration n. a_k is the control variable and $f(s_k, a_k)$ represents the system dynamics [29]. The DP problem is discretized into n sub-optimization steps, and a sequence of control inputs, following the optimal policy $\pi = \{a_0^*, a_1^*, \dots a_{n-1}^*\}$, is found based on the principle of optimality.

Created by Bellman, the principle of optimality states that an optimal policy has the property that, independent of the initial state and decision, the remaining decisions must form an optimal policy with regards to the state resulting from the first decision [57]. The Bellman principle contributed to the popularization of DP and its recent transformation into a systematic tool in different optimal control problems.

For the P2-PHEV quasi-static system, SoC is the state variable and driving mode is the control variable. The discrete-time space PHEV system and the overall control optimization problem are reformulated in equation (4.3) and defined to minimize the cost function formulated in equation (4.4). Optimal control policy, which is the sequence of driving modes selection during the cycle, needs to be found under various constraints for each component as described in equations (4.5) to (4.9).

$$SoC_{k+1} = f(SoC_k, a_k), \qquad k = 0,1, \dots n-1 \tag{4.3}$$

$$min_{u(k)} \quad m_{fuel} = \left[\dot{m}_{fuel}(SoC_n) + \sum_{k=0}^{n-1} \dot{m}_{fuel}(SoC_k, a_k)\right] \Delta t \tag{4.4}$$

Subject to

$$a_k \in \{a_1, a_2, \dots a_{k_{max}}\} \tag{4.5}$$

$$SoC_{min_k} \leq SoC_k < SoC_{max_k}, \qquad SoC_0 = SoC_{init} \tag{4.6}$$

$$T_{ICE_{min_k}} \leq T_{ICE_k} < T_{ICE_{max_k}} \tag{4.7}$$

$$T_{EM_{min_k}} \leq T_{EM_k} < T_{EM_{max_k}} \tag{4.8}$$

$$I_{Bat_{min_k}} \leq I_{Bat_k} < I_{Bat_{max_k}} \tag{4.9}$$

A backward DDP-based solver is developed in Algorithm 1 assuming a deterministic environment which means that stochastic uncertainty is not considered [30].

Algorithm 1: Deterministic Dynamic Programming for P2-PHEV

Step 1 (Initialization)
  Define initial and final state variables.
  Initialize k discrete steps number and state grid.
Step 2 (Backward Recursion):
  for $k = n - 1 \rightarrow 0$
    Calculate cost $J_k(s_k) = \min_{u_k}[L(s_k, a_k) + J^*_{k+1}(s_{k+1})]$
    $\underset{a_k^*}{\operatorname{argmin}} J_k(s_k)$
    Memoize min cost into optimal cost-to-go function $J^*_{k+1}(s_{k+1})$
  end for;
Step 3 (Forward simulation):
  for $k = 0 \rightarrow n - 1$
    Apply a_k^* from optimal policy $\pi^* = \{a_0^*, a_1^*, a_2^* \dots, a_{n-1}^*\}$
    Calculate the accumulated cost $J(s, a^*) \leftarrow \sum_{k=0}^{n-1} J_k(s_k, a_k^*)$
  end for;

4.1.2 DRIVING MODES SEGMENTATION

In HEV optimization problems, the driving modes can be divided into two operating segments: free segment and fixed segment. Fixed modes are selected in all the discrete-time steps where the SoC change is explicit and purely determined by the driver or the diving cycle. However, free modes are selected in the time steps where HCU must make the power distribution decision [58]. Additionally, the emergent modes category contains the modes applied in the emergent situations where the SoC is very close to the lowest boundary. This segmentation helps to shrink the computation burden of the DP since there is no need to optimize the control variable for the fixed segments.

The driving modes categories are explained in more detail as follows:

- Fixed segments (AddBoost, Recup, and Stop/Standstill):

The AddBoost mode is activated when the demanded torque is beyond the ICE maximum torque under certain speeds. The ICE maximum torque is defined as a component limitation in the PHEV system which cannot be overridden by the HCU decision. Recup mode is selected always when the demanded torque is negative. The calibration parameters in P2-PHEV defined the SoC range for recuperation to be between 94% and 21%. The battery overcharge phenomenon can only happen when a very deep steep downhill driving situation is encountered when the battery is full, which is not the case in all the driving cycles used in this thesis. Stop/Standstill is considered a fixed mode as it is activated if the demanded torque and velocity are both zero.

- Free segments (ConvDrv, EltlDrv, OptmGentn, and SubBoost):

  Free segments include the time steps where deciding the energy distribution is crucial for accomplishing the optimization target. The ConvDrv and EltlDrv modes stand for drawing the demanded energy from a single energy source, either the electricity or the fuel. OptmGentn and SubBoost are not enabled all the time in reality due to other constraints and objectives such as stability and NVH. For the P2 PHEV, the OptmGentn mode is more limited since the ICE is always connected to the driveline.

- Emergent Modes (ldleGentn and MinGentn):

  These category modes are selected when the battery SoC is near the minimum threshold, 20% for the sake of battery health. However, the ICE under both modes is not usually working on optimal load points.

- Excluded modes (SubBoost):

  Substitute Boost (SubBoost) is not allowed in P2 PHEV and is replaced by ConvDrv when SoC falls below the lower threshold or EltlDrv if it exceeds the upper threshold.

To find the global minimum fuel consumption, or in other words, to solve the GOP's control problem, DDP needs to search for the optimal modes' selection only in the time segments where free modes are available. In this work, the time segments where the

demanded torque is negative or equals zero in addition to both emergent modes and Sub-Boost mode are not considered as a selection option to the agent.

4.2 REINFORCEMENT LEARNING

RL is a subset of machine learning used for solving control problems. This method allows the agent to automatically determine the ideal behavior by reinforcing or inhibiting patterns of behavior to maximize the reward, which is the environment feedback. The RL differs from supervised learning, a commonly known technique in machine learning, in that the latter uses the training data to explicitly correct undesired behaviors. However, RL applies sequential decisions and learns through a delayed environment feedback (reward/penalty) by giving appropriate compensation for the learning outcomes which makes it more suitable for applications in real-time such as the HEV's EMS.

As illustrated in the RL concept in Figure 20, the system starts at state s_t and the agent decides on a certain action a_t that is executed in the environment. The next state s_{t+1} (often called s' in several studies) and the next reward r_{t+1} are fed by the environment back to the agent where the learning process is done accordingly.

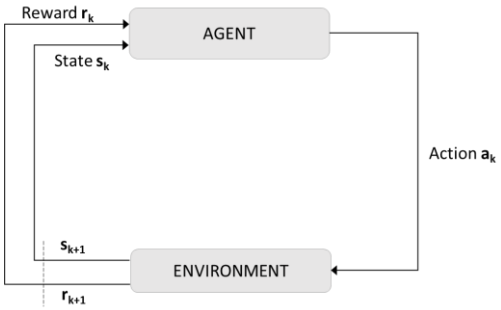


Figure 20 - Reinforcement learning concept

4.2.1 TABULAR Q-LEARNING BASED EMS

Among popular RL algorithms, Q-learning, developed by Werbos and Watkins in 1989 [59- 60], is a widely used method to learn optimal control policies online. It depends on the Temporal Difference (TD) technique to estimate future rewards giving the current state in a bootstrapping fashion [61].

The HEV cost function in an infinite horizon is described in equation (4.10) and expressed recursively using the Bellman equation as equations (4.11) and (4.12). γ is the discount factor which determines either the agent cares more about the immediate or the distant rewards in the future when γ approaches 0 or 1 respectively.

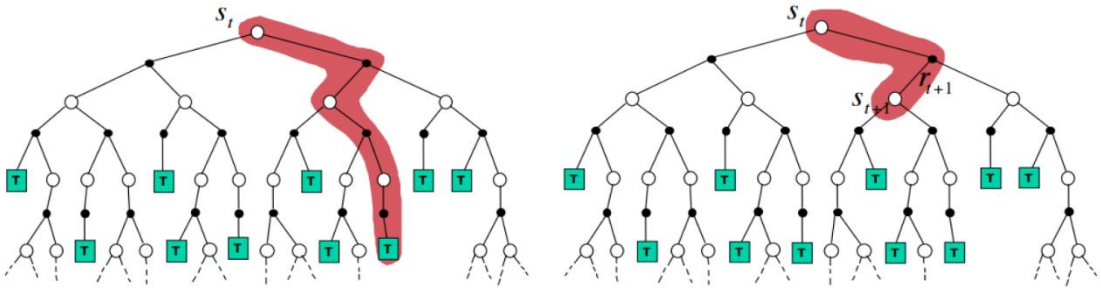
$$J_\pi(s_k) = \sum_{i=k}^{\infty} \gamma^{i-k} L(s_i, a_i) \tag{4.10}$$

$$J_\pi(s_k) = L(s_k, a_k) + \gamma \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} L(s_i, a_i) \tag{4.11}$$

$$J_\pi(s_k) = L(s_k, \pi(s_k)) + \gamma J_\pi(s_{k+1}) \tag{4.12}$$

Equation (4.12) satisfies the Bellman optimality condition when the right-hand side equals the left-hand side or at least the difference between them is minimized. The temporal difference error is defined as the difference between them as equation (4.13) states.

$$e_k = L(s_k, \pi(s_k)) + \gamma J_\pi(s_{k+1}) - J_\pi(s_k) \tag{4.13}$$



a) Monto Carlo *b) Temporal Difference*

Figure 21 - Monte Carlo and Temporal Difference learning backup diagrams [62].

TD learning estimates cumulative future reward based on the current reward and next state estimate while Monto Carlo learning (MC) randomly samples the environment for the entire episode till the terminal state as shown in Figure 21. TD learning shows

improved real-time performance in comparison to MC learning although it comes at the expense of the sample efficiency and the solution suboptimality.

The HEV control problem cost $J_\pi(s_k)$ can be updated using the equation (4.13) as follows:

$$J_\pi(s_k) \leftarrow J_\pi(s_k) + \alpha\,[L(s_k, \pi(s_k)) + \gamma\, J_\pi(s_{k+1}) - J_\pi(s_k)] \tag{4.14}$$

The learning rate α, which controls the update process, ranges from 0 to 1 where 0 means no learning happens and 1 means that the update fully uses the most recent knowledge. The Q-learning approach defines the Q-value function as a state-action value function following equation (4.15) [62].

$$Q_\pi(s_k, a_k) = L(s_k, a_k) + \gamma\, J_\pi(s_{k+1}) \tag{4.15}$$

$Q_\pi(s_k, a_k)$ and $L(s_k, a_k)$ are the function and the environment immediate reward of the state s_k and the action a_k respectively. Using the TD learning, the Q-function can be updated in equation (4.16) in the same way as equation (4.14).

$$\mathrm{Q}(s_k, a_k) \leftarrow \mathrm{Q}(s_k, a_k) + \alpha\,[r_k + \gamma \max_a \mathrm{Q}(s_{k+1}, a) - \mathrm{Q}(s_k, a_k)] \tag{4.16}$$

Equation (4.16) presents the essence of the Q-learning algorithm whose convergence for a finite MDP has been proven based on stochastic approximation methods by Watkins [59].

- **Exploration/ exploitation strategy**

For an RL agent, to learn a correct state-action value (Q-value) for all possible state-action pairs and find the optimal policy that maximizes the total cumulative reward, it shall try different actions in different states which is referred to as exploring the environment “exploration”. On the other hand, if the agent is just exploring, the episodic cumulative reward is never maximized. Accordingly, the agent shall make use of the policy learned, which is called "exploitation", to exploit the available knowledge to maximize the rewards received.

The trade-off between exploration and exploitation is one of the biggest challenges in RL problems. The ε-greedy policy is a widely used technique in several studies [24, 28, 50]. The strategy defines an exploration rate ε that is set initially to one. This exploration rate is the probability that the agent will explore the environment rather than exploit it. With ε = 1, the agent selects a random action sampled uniformly each time step. As the agent learns more about the environment, ε shall degrade at a pre-specified rate so that the probability of exploration becomes less, and greedy actions are selected more frequently with 1- ε probability.

- **Problem space definition**

The RL agent for P2-PHEV has a continuous state space defined by $s_k = [SoC_k, T_{tot_k}, V_k, D_{rem_k}, E_{on_k}]$ where SoC_k is the current battery state of charge, T_{tot_k} is the driver torque demand, $V_k$is the vehicle velocity, $D_{rem_k}$is the trip remaining distance and E_{on_k} is the engine on/off state at time step k. D_{rem_k} and E_{on_k} is included in the state space for a later use in the RL reward function definition in section 4.2.4.3.

The action space is discrete where the control variable, the driving mode selection, $a_k \in \{0,1,3,6,7,8\}$ because this driving mode set consists of the allowed modes to be selected by the agent. Refer to Table 1 and section 2.2 for more details.

Analogous to equations (4.3) to (4.9), P2-PHEV's discrete-time space control optimization problem, reward function $r(s_k, a_k)$ and system constraints as described in equations (4.17) to (4.24).

$$[SoC_{k+1}, T_{tot_{k+1}}, V_{k+1}, D_{rem_{k+1}}, E_{on_{k+1}}] = f\big([SoC_k, T_{tot_k}, V_k, D_{rem_k}, E_{on_k}], a_k\big), \quad k = 0,1,\dots n-1 \tag{4.17}$$

$$min \quad J_\pi(s_0) = \lim_{n\to\infty} E\{\sum_{k=0}^{n-1} \gamma^k \, r(s_k, a_k)\} \tag{4.18}$$

$$where \quad r(s_k, a_k) = \dot{m}_{fuel}(s_k, a_k) \tag{4.19}$$

Subject to

$$a_k \in \{0,1,3,6,7,8\} \tag{4.20}$$

$$SoC_{min_k} \leq SoC_k < SoC_{max_k}, \qquad SoC_0 = SoC_{init} \tag{4.21}$$

$$T_{ICE_{min_k}} \leq T_{ICE_k} < T_{ICE_{max_k}} \tag{4.22}$$

$$T_{EM_{min_k}} \leq T_{EM_k} < T_{EM_{max_k}} \tag{4.23}$$

$$I_{Bat_{min_k}} \leq I_{Bat_k} < I_{Bat_{max_k}} \tag{4.24}$$

- **Q-function representation**

The tabular Q-Learning based EMS algorithm updates the Q-value for each state-action pair using equation (4.16) until the Q-function converges to the optimal Q-function (Q^*). This approach is called the iteration of values. Q-function is represented in a table called Q-table whose rows represent the states, and the columns represent the actions as illustrated in Figure 22. Thus, the table dimensions are the number of states multiplied by the number of actions.

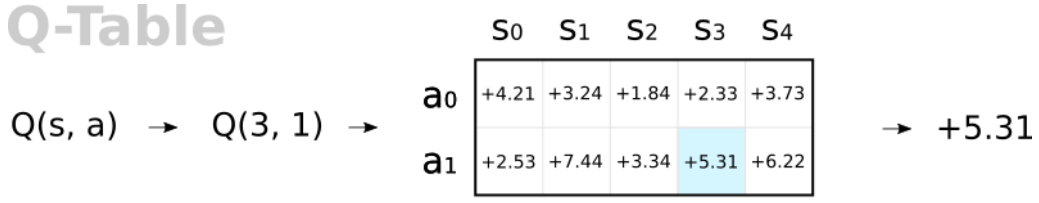


Figure 22 - Q-function tabular representation [63].

Since the agent knows nothing about the environment or the expected returns for the state-action pairs, all Q-values in the table are first initialized to zeros. During training, the agent performs several episodes where the Q-values produced for the experienced state-action pairs are updated and stored in the Q-table. Therefore, continuous state space is not possible as the Q-table number of rows of would be infinite, and accordingly, state-space discretization is a necessity.

According to the exploration strategy, the agent selects the greedy action based on the highest Q-value for the current state, that is why Q-learning is called a value-based RL

method. The P2-PHEV's EMS algorithm based on a model-free Q-learning RL agent is represented in Algorithm 2.

Algorithm 2: Model-free Q-learning algorithm for P2-PHEV

Set values for learning rate α, discount factor γ, epsilon ε, epsilon decay $d\varepsilon$
Initialize $\mathrm{Q}(s, a)$ to zeros
Repeat **for** episode = 1: number of episodes
    for k = 1: number of steps per episode
        With probability ε, select a random action a_k
        Otherwise, select $a_k = argmax\, Q(s, a)$
        Execute action a_k and observe reward r_k and state s_{k+1}
        Update Q

$$\mathrm{Q}(\mathrm{s}, a) \leftarrow \mathrm{Q}(\mathrm{s}, a) + \alpha\,[r + \gamma \max_{a} \mathrm{Q}(\mathrm{s}', a) - \mathrm{Q}(\mathrm{s}, a)]$$

        $\mathrm{s} \leftarrow \mathrm{s}',\ \varepsilon \leftarrow \varepsilon \cdot d\varepsilon$
        end for;
    end for;

4.2.2 DEEP Q-LEARNING BASED EMS

Deep Q-learning, often called Deep Q-Network (DQN), is an advanced version of Q-Learning that allows the agent to utilize Deep Learning (DL) capabilities to achieve improved performance in complex control problems. DQN was first proposed by DeepMind in a paper published by Nature in 2015 solving challenging problems of classic Atari 2600 games [64]. Subsequently, several researchers investigated using DQN because the DL could empower the RL to directly deal with high-dimensional continuous state spaces which imply an infinite number of states.

The iterative process of calculating and updating Q-values for each state-action pair in a large state space becomes inefficient and perhaps impractical due to the limited computational resources and real-time performance requirements. The advantage of DQN is that the agent control policy can be a continuous function (neural network defined by several

hyperparameters) that can be generalized to new unvisited states or previously visited states with uncertainty.

4.2.2.1 NEURAL NETWORK FUNCTION REPRESENTATION

The DNN representation is used to estimate and update the Q-values of each state-action pair in a given environment, and in turn by agent training, the network approaches the optimal Q-function. Figure 23 shows how a neural network can represent the same Q-values as the tabular Q-learning method shown previously in Figure 22.

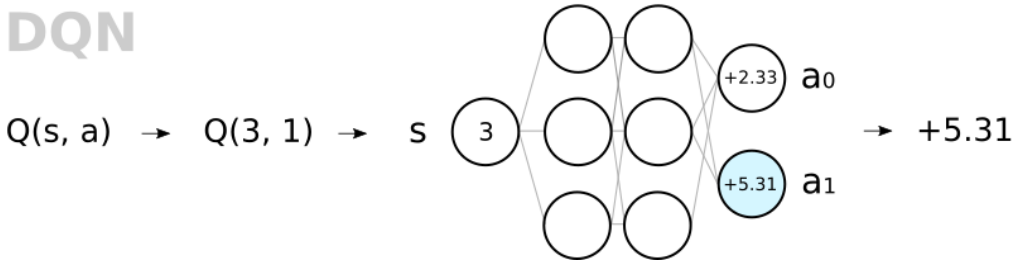


Figure 23 - Q-function neural network representation [63].

The DQN has an Artificial Neural Network (ANN) with a state input layer, n-hidden layers with m-neurons in each layer, and a linear regression output layer. The input layer includes state values $s_k = [SoC_k, T_{tot_k}, V_k, D_{rem_k}, E_{on_k}]$ and the output layer estimates the Q-value of each action in the agent action space. Such a network that represents the agent control policy is called a "Policy Network" as demonstrated in Figure 24. The EM RL algorithm based on a model-free Q-network agent is represented in Algorithm 3. The RL training parameters and Neural Network (NN) hyperparameters are discussed in chapter 5 in the context of the experiments conducted using the proposed algorithm.

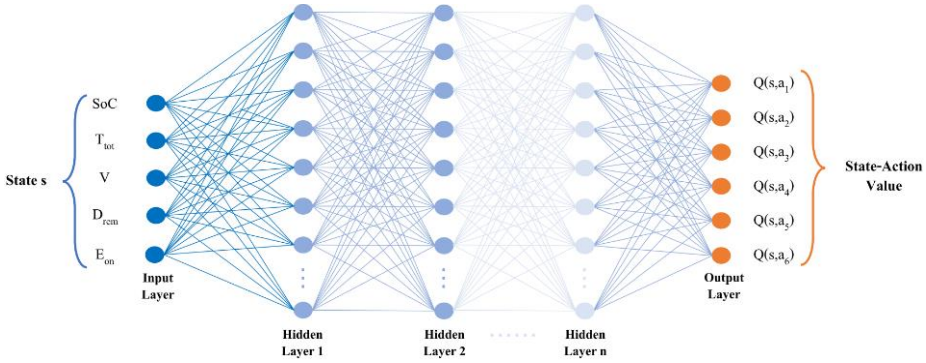


Figure 24 - DQN policy network

Algorithm 3: Model-free DQN algorithm for P2-PHEV

1: Set values for RL training parameters and NN hyperparameters
2: Initialize Q network with random weights θ and memory D with capacity N
3: **for** episode = 1: number of episodes
4: Reset environment with s_0
5: **for** k = 1: number of steps per episode
6: With probability ε, select a random action a_k
7: Otherwise, select $a_k = argmax\, Q(s_k, a)$
8: Execute action a_k and observe reward r_k and state s_{k+1}
9: Store transition (s_k, a_k, r_k, s_{k+1}) in memory D
10: $s \leftarrow s',\ \varepsilon \leftarrow \varepsilon \cdot d\varepsilon$
11: Sample random minibatch of (s_j, a_j, r_j, s_{j+1}) from memory D:
12: Set $y_j = \begin{cases} r_j & \text{if } s_j \text{ is terminal} \\ r_j + \gamma \max\limits_a Q(s_{j+1}, a;\ \theta) & \text{otherwise} \end{cases}$
13: Perform gradient descent step on $\left(y_j - Q(s_j, a_j\ ;\ \theta)\right)^2$
14: **end for;**
15: **end for;**

- **DNN-learning based on experience replay**

The agent experience is defined as the 4D tuple (state, action, reward, next state) = (s_k, a_k, r_k, s_{k+1}). Replay Memory is a set of experiences of size N that uses the standard First In First Out (FIFO) buffer to ensure that the network is always trained on all new incoming experiences. If network learning is only done from the successive experiences directly from the environment, experiences would be strongly correlated, and the learning would be therefore inefficient. Accordingly, sampling random experience tuples from the Replay Memory resolves such a correlation problem.

DNN training begins at step 11 in Algorithm 3 by retrieving a random sample batch (s_j, a_j, r_j, s_{j+1}) from the replay memory. The state batch s_j is fed as input to the policy network for the forward propagation. The network then produces an estimated Q-value for each possible action from the given input state. The optimal Q-values (often called

target Q-values) are calculated at step 12 in Algorithm 3 which is the same as the Q-learning update equation (4.16). The loss function is then defined as the error between the optimal Q-value and the Q-value of the action stored in the replay memory as formulated in equation (4.25).

$$Loss = (r_j + \gamma \max_a Q(s_{j+1}, a;\ \theta)) \quad - Q(s_j, a_j\ ;\ \theta) \tag{4.25}$$

A DNN optimization method, such as batch gradient descent, is then used to update the network weights to minimize the loss. Minimizing the loss leads the output Q-values of the policy network for each state-action pair to approximate the target Q-values given by the Bellman equation. The reader is referred to reference [65] for more information about the DNN forward/backward propagation and training processes.

Batch Gradient Descent (BGD) is a basic algorithm used to drive the convergence of the neural network by minimizing the cost function formulated in equation (4.25). BGD is a relatively fast optimizer and intuitive to grasp the mathematics behind it. However, it can be easily stuck at a local minimum, and convergence is not always guaranteed with relatively large learning rates. Other optimizers such as Stochastic Gradient Descent (SGD) [66], Adaptive Gradients (AdaGrad) [67], and Root Mean Squared Propagation (RMSprop) [68] are used in several studies. Nevertheless, the Adaptive Moment Estimation (ADAM) optimizer is widely used in DRL problems because it combines the AdaGrad and RMSProp algorithms' best properties to efficiently handle sparse gradients on noisy data [69].

4.2.2.2 ADDITIONAL DQN TYPES

- **Double DQN**

Seven months after proposing DQN by DeepMind in 2015, Hasselt et al. introduced the Double DQN (DDQN) to reduce the observed Q-values overestimation bias. It is done by decoupling the action selection from the target Q-value estimation leading to a more stable training and an improved policy [70]. Using the same policy network in traditional DQN, the estimated Q-values will update, and the target Q-values will also be discounted

since they are calculated using the same weights. Subsequentially, the Q-values are updated within each iteration to get closer to the target Q-values which also move in the same direction, and this creates a problem that the policy network tries to catch up with itself.

Hasselt et al proposed a separate target network that is a clone of the policy network with the same architecture and the same weights. However, the target network weights are frozen and only periodically or slowly get updated to match the policy Q-network weights. In this way, the agent training can proceed in a much more stable manner. Algorithm 4 shows the implementation of the DDQN approach for the P2-PHEV problem.

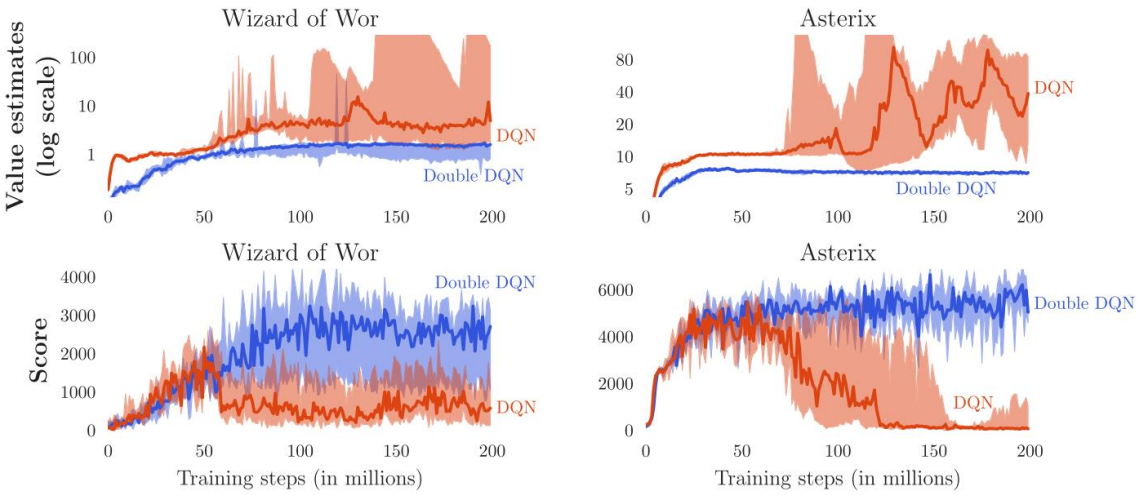


Figure 25 – Hasselt et al. results for DDQN agent playing Atari games and showing more stable training performance in comparison to traditional DQN [70].

Algorithm 4: Model-free Double DQN algorithm for P2-PHEV

Set values for RL training parameters and NN hyperparameters
Initialize replay memory D with capacity N
Initialize policy network Q with random weights θ
Initialize target network $\widehat{\mathrm{Q}}$ with random weights θ^{-}
for episode = 1: number of episodes
    Reset environment with s_0
    for k = 1: number of steps per episode
        With probability ε, select a random action a_k

9: Otherwise, select $a_k = argmax\, Q(s, a)$

10: Execute action a_k and observe reward r_k and state s_{k+1}

11: Store transition (s_k, a_k, r_k, s_{k+1}) in memory D

12: $s \leftarrow s',\ \varepsilon \leftarrow \varepsilon \cdot d\varepsilon$

13: Sample random minibatch of (s_j, a_j, r_j, s_{j+1}) from memory D:

14: Set $y_j = \begin{cases} r_j & \text{if } s_j \text{ is terminal} \\ r_j + \gamma \max_a \hat{Q}(s_{j+1}, a;\ \theta^-) & \text{otherwise} \end{cases}$

15: Perform gradient descent step on $\left(y_j - Q(s_j, a_j\ ;\ \theta)\right)^2$ with respect to the policy network parameters θ

16: Every C steps, set $\hat{Q} = Q$

17: **end for;**

18: **end for;**

- **DQN critic representation**

Although Mnih et.al used the DQN architecture similar to Figure 24, they referred to another representation, called critic representation, to parameterize the Q-function using neural networks [64].

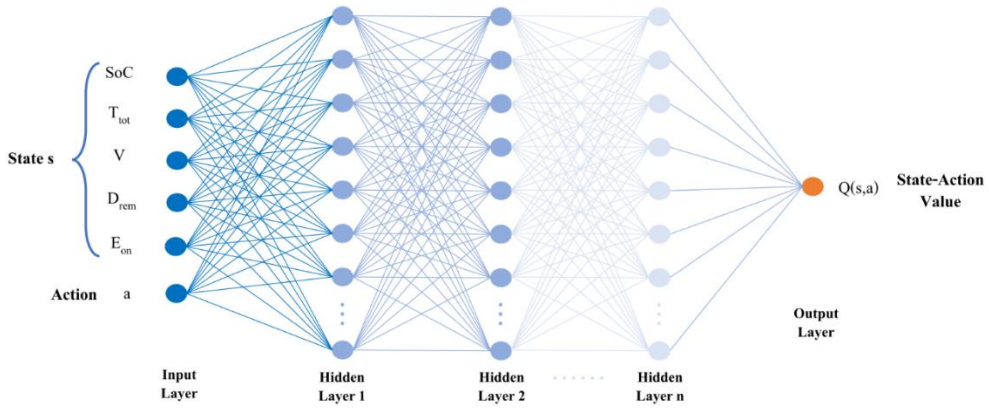


Figure 26 - DQN critic representation.

Critic representation, as shown in Figure 26, feeds the state-action pair as an input to the neural network and retrieves the state-action value Q(s, a) as an output. This representation output differs from the commonly used representation where the output is the Q-value for all actions in the discrete action space. Critic representation has the drawback

of being required to perform a separate forward propagation to calculate the Q-value for each available action to select an action according to the policy which results in more computational cost. Such a drawback is not critical in the existing HEV problem domain because the maximum free modes the agent can select from each time step are three.

However, the main advantage of the critic representation is the efficient backward propagation. According to step 14 in Algorithm 4, the target value y_j is calculated only for the executed action while the other five action values are backpropagated with the same value. Consequently, the network is adjusting only one of the six output values, hence the network learning is not very efficient. On the other hand, the critic representation has one output that is updated completely by the target value and the network is utilizing the available data in the best way. This architecture is widely used in the MATLAB reinforcement learning toolbox and the examples provided by Mathworks for discrete action space environments with Q-learning, DQN, and SARSA algorithms [71].

4.2.3 DRL IMPROVEMENT TECHNIQUES

4.2.3.1 NEURAL NETWORK ARCHITECTURE OPTIMIZATION

Incorporating deep learning models into the RL world witnessed intensive studies on improving the agent performance. Various studies confirmed that the agent performance is affected by changing the neural network type, architecture, hidden layers, number of neurons in each layer, activation functions, and layer inputs normalization/standardization [72, 73].

Several scholars tackling similar HEV control problems using DQN agents used different architectures. Most of them used the Multi-Layer Perceptrons (MLP) while few considered Recurrent Neural Networks (RNN). Zhaoxuan et al. utilized an MLP neural network with four hidden layers each having 128 neurons followed by a Rectified Linear Unit (ReLU) layer [74]. Chengzhao et al. constructed the NN out of two hidden layers with 350 neurons followed by a ReLU layer each [75]. Other architectures are considered such as the pyramid architecture incorporating three hidden layers with [200 100 50] neurons

respectively by Renzong et al. [76] and the inverse-pyramid architecture with three hidden layers with [20 50 100] neurons, respectively [50].

Given the diversity of the available NN structures, a separate optimization study is conducted to select the optimal NN architecture fitting our problem domain, cycles, and reward function the best. The DQN agent ran for a few episodes to collect experiences with a full exploration behavior and the data was recorded for further processing. The step rewards were accumulated by the end of each episode to represent the expected cumulative return which will represent the model's desired output. The input to the model will be the same as the DQN agent input state and action vector.

Several NN architectures are defined to represent the aforementioned diverse designs, trained using a portion of the available data, and tested with the rest to calculate the generalization error for each design as a Root Mean Square Error (RMSE). The architecture which gives the minimum RMSE is selected as the function approximator for the subsequent development steps.

4.2.3.2 N-STEPS BOOTSTRAPPING

The bellman equation for estimating the Q(s, a) of each state-action pair was first introduced by Sutton in 1988 [61] as follows:

$$Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, a_{t+1}) \tag{4.26}$$

Equation (4.26) represents the 1-step TD which is often called TD(0). However, the equation is recursive which means $Q(s_{t+1}, a_{t+1})$ can be replaced by its estimate from s_{t+1} assuming a_{t+1} is chosen optimally or near optimally regarding the agent current policy as expressed in equation (4.27).

$$Q(s_t, a_t) = r_t + \gamma\, r_{t+1} + \gamma^2 \max_a Q(s_{t+2}, a_{t+2}) \tag{4.27}$$

Accordingly, 2-step TD up to ∞-step TD can be achieved as illustrated in Figure 27 and this technique is called n-steps bootstrapping. Mathematically, n-steps bootstrapping will help the cumulative rewards to quickly propagate back to the previous states and the Q-

function to converge faster. The ∞-step TD, in this case, will represent the Monto Carlo learning as shown previously in Figure 21.

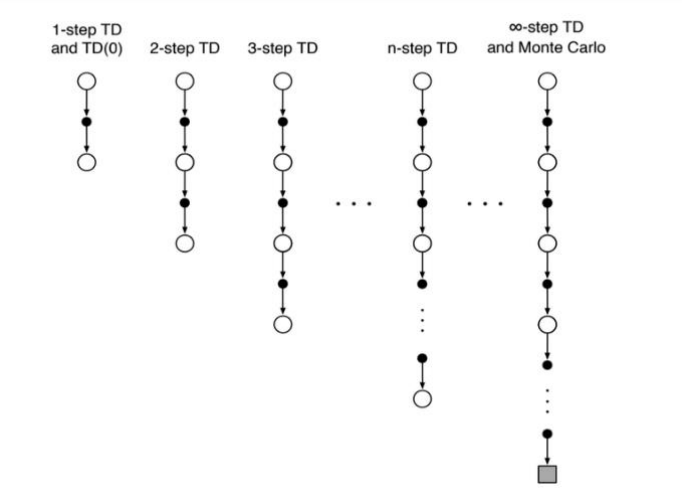


Figure 27 - N-steps bootstrapping return, Sutton [62].

Although n-steps bootstrapping can improve the Q-function approximation convergence, it should be used wisely because unrolling equation (4.26) too often will cause the DQN to be an on-policy algorithm. DQN is considered an off-policy algorithm because the experiences available in the buffer don't depend on the agent's policy. However, unrolling equation (4.26) to equation (4.27) included selecting the action a_{t+1} optimally according to the agent policy, hence the algorithm is getting closer to an on-policy behavior by increasing the n-value further.

This technique has been studied systematically by Fedus et al. recently in 2020 revealing that the agent's performance will rely heavily on the proper selection of the n-value. They gave suggestions based on empirical results to select the n-value to be small, e.g., 3 or 4 [77].

4.2.3.3 PRIORITIZED EXPERIENCE REPLAY

The experience replay technique, introduced in section 4.2.2.1, is used to improve the agent's learning capabilities by remembering and reusing past experiences. The experiences are sampled uniformly from the memory buffer into mini-batches for agent training regardless of their importance and significance. Although random sampling breaks the

correlation between experience samples, task-relevant and important transitions are missed often which results in poor data utilization decreasing the learning efficiency and speed.

Schaul et al. proposed a different sampling technique that samples more frequently the transitions with high expected learning outcomes [78]. The TD error is calculated using equation (4.28) during appending the transition into the memory buffer to be used later to evaluate the sample importance. Equation (4.29) expresses the importance probability of an experience sample where ε is a small number to avoid the zero division, and α determines the priority sampling degree where zero means random sampling and one means full prioritization.

$$TD = |Q(s_{t+1}, a_t) - Q(s_t, a_t)| \tag{4.28}$$

$$p_i = \frac{(TD_i + \varepsilon)^{\alpha}}{\sum_k^{memory\ size} (TD_k + \varepsilon)^{\alpha}} \tag{4.29}$$

The Q-value update in DQN depends mainly on the assumption of the consistency of both mini-batches and memory buffer sample distributions. However, prioritization will change the mini-batch sample distribution which brings an additional error source in the Q-value estimation and the agent policy. Accordingly, the authors proposed importance-sampling weight (IS weight) which is a factor to be multiplied by the sample priority to compensate for such error in equation (4.30) and is multiplied by the training loss as expressed in equation (4.31).

$$Importance = \left(\frac{1}{p_i} \cdot \frac{1}{memory\ size}\right)^{\beta} \tag{4.30}$$

$$J = \frac{1}{m} \sum \left(y_i - Q(s_i, a_i\,;\,\theta)\right)^2 \cdot Importance \tag{4.31}$$

β is the IS weight factor controlling how much prioritization is to be applied where $\beta \in (0 \leq \beta \leq 1)$. The authors argue that the training is highly unstable at the beginning, therefore it is recommended to start β with a small value of 0.4 up to 0.6 and to anneal it

gradually towards one where the importance sampling correction is more significant near the end of training.

4.2.3.4 TAU-SOFT UPDATE

The DDQN algorithm, introduced in section 4.2.2.2, solves the moving target problem by having a separate target network for the Q-value estimation which is updated frequently each C steps as demonstrated in algorithm 4. Lillicrap et al. used the 'τ-soft update' technique to make the target network weights slowly track the learning policy network instead of directly copying the weights as illustrated in equation (4.32) [79].

$$\theta^{-} = \tau\,\theta + (1-\tau)\,\theta^{-} \tag{4.32}$$

In algorithm 4, step 16 is simply replaced by equation (4.32). Choosing the τ value to be small, such as 0.001, constrains the target network change to be slow, hence greatly improving the learning stability. Although the Q-value propagation is delayed by the target network soft update, the authors claim that such minor change guarantees convergence to a robust solution by moving the unstable function approximation problem closer to a supervised learning problem.

4.2.4 HEVS SPECIAL CONSIDERATIONS

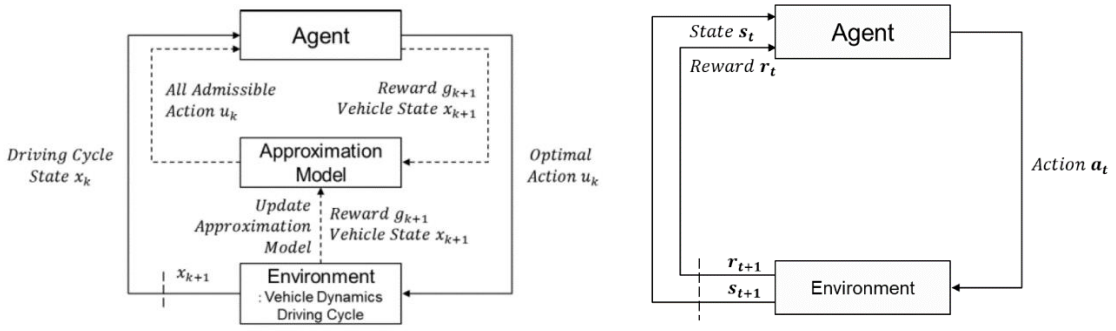
Although the topic of hybrid controls based on RL algorithms drew huge attention in recent years, most scientific publications considered replacing the energy management optimization part completely with the RL. The drawback of such an approach, however, is the difficulty to achieve a robust real-time control due to the huge computational resources needed. Moreover, system constraints, safety, diagnostics, and component protection topics, which constitute a large number of hybrid control modules, are not considered, which limits this approach's applicability to prototype and demonstration vehicles. The following section discusses the feasible methodologies to implement the developed RL agent into series-intended controls.

4.2.4.1 MODEL-BASED RL

Planning and reinforcement learning were merged into a research field called model-based RL. Such a new field can be defined as "any MDP approach that uses both a reversible model (known or learned) and learning of a value or policy to act on the environment" [80].

Several model-based RL approaches were considered by researchers such as sampling additional data (often called Dyna-style algorithms [62, 81]), multi-step approximate dynamic programming, backward trials, and value gradients. The first approach is considered by the thesis while the reader is referred to [80] for more information regarding the other approaches.

In 2016, Gu et.al with a team from Google proposed using sampling additional simulated data called imagination rollouts to improve the agent learning capabilities and lead to better policy [82]. Imagination rollouts are generated from a learned dynamics model to accelerate their model-free RL agent. They concluded that such a technique was extremely effective only when the learned dynamics model matches the true environment model perfectly. However, performance degrades dramatically with the imperfect learned model.



a) Model-based RL *b) Model-free RL*

Figure 28 - Model-based and model-free Reinforcement learning in HEVs [83].

In hybrid-electric vehicles, the powertrain model is deterministic which means the vehicle state (such as $SoC_t, \dot{m}_{fuel_t}, E_{on_t}$) moves deterministically to the next state (SoC_{t+1}, $\dot{m}_{fuel_{t+1}}, E_{on_{t+1}}$) in the driving environment by the control input. Lee et al. used such

system characteristics to develop a model-based Q-learning agent with two separate statistical models; one for the SoC prediction and the other for the fuel consumption estimation [83]. Both models are combined into an approximation model and used in a Dyna-Q algorithm to generate synthetic experiences and improve the agent policy with as shown in Figure 28a compared to Figure 28b.

Based on the findings of Gu et al. and Lee et al., two supervised learning NN models were developed using the same methodology, as explained in section 4.2.3, to generate the data and train several architectures for the minimum training and generalization errors. Both models are combined into a deterministic environment model to be used in generating synthetic experiences each time step and appending them into a simulated replay buffer as shown in Figure 29. Simulated experiences are bootstrapped for n-steps in the future to provide more on-policy behavior for the agent which showed some benefit in Q-function approximation in the early stages of the learning process as concluded by Gu et al. [82]. The detailed algorithm of the model-based DDQN agent with n-steps bootstrapping is presented in algorithm 5.

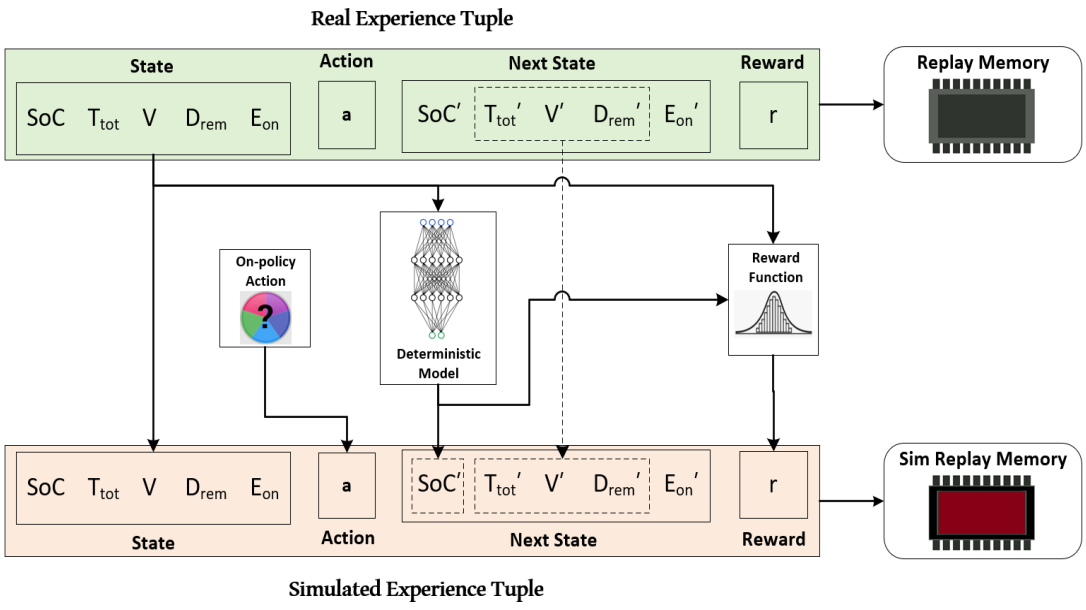


Figure 29 - Real and simulated experiences in model-based RL.

Algorithm 5: Model-based Double DQN algorithm with n-steps bootstrapping for P2-PHEV

Set values for RL training parameters and NNs hyperparameters
Initialize replay memory D, simulated replay memory S and model buffer M with capacity N
Initialize policy network Q with random weights θ
Initialize target network $\widehat{\mathrm{Q}}$ with random weights θ^{-}
Load pre-fitted dynamics model $\mathcal{M} \leftarrow \emptyset$
for episode = 1: number of episodes, **do**
    Reset environment with s_0
    for k = 1: number of steps per episode, **do**
        With probability ε, select a random action a_k^1
        Otherwise, select $a_k^1 = argmax\, Q(s, a)$
        Execute action a_k^1, observe reward $r_{k,1}$ and state s_{k+1}
        Simulate actions $a_k^{2,3}$, observe reward $r_k^{2,3}$ and state $s_{k+1}^{2,3}$
        Store transition $T_k^1 = (s_k, a_k^1, r_k, s_{k+1})$ in memories D, M and simulated transitions $T_k^2 = \left(s_k, a_k^2, r_k^2, s_{k+1}^2\right)$ and $T_k^3 = (s_k, a_k^3, r_k^3, s_{k+1}^3)$ in memory S
        $\mathrm{s} \leftarrow \mathrm{s}',\ \varepsilon \leftarrow \varepsilon \cdot d\varepsilon$
        Use last n transitions from D to bootstrap T_{k-n}^1 for n-steps, use $\mathcal{M}$ to bootstrap T_{k-n}^2 and T_{k-n}^3 with on-policy actions and replace all in D, S buffers
        Sample minibatch of m transitions from memories D and S
        Set $y_j = \begin{cases} r_j & \text{if } s_j \text{is terminal} \\ r_j + \gamma \max\limits_a \hat{Q}\left(s_{j+1}, a;\ \theta^{-}\right) & \text{otherwise} \end{cases}$
        Perform gradient descent step on $\left(y_j - Q\left(s_j, a_j\ ;\ \theta\right)\right)^2$ with respect to policy network parameters θ
        Update target network weights $\theta^{-} \leftarrow\ \tau\ \theta\ +\ (1 - \tau)\ \theta^{-}$
    end for;
    Sample minibatch of random transitions from memory M
    Perform gradient descent on $\left(y_j - \mathcal{M}\left(s_j, a_j; \emptyset\right)\right)^2$ on model parameters $\emptyset$
end for;

4.2.4.2 ACTION MASKING

Incorporating system constraints, safety, diagnostics, and protection into the HEV's RL-developed algorithm is achieved through the action masking technique. The idea is to define the conditions imposed by the system functionalities as a "mode enabler" vector which is considered by the RL agent to select the appropriate action each time step. In this way, only the modes that are allowed from the system point of view will be selected by the RL agent.

The conventional RL architecture is shown in Figure 30a where the agent freely selects the action according to its policy and the environment responds with the system state and reward. While Figure 30b illustrates the modification proposed to the RL architecture based on the action masking idea presented by Vinyals et al. [84]. The main part of HCU decides on the available modes for the RL to select from according to its rule-based control scheme and constructs the 'mode enabler' logical vector. This vector is to be multiplied by the action space modes to result in a vector of the only available actions at the current state. The RL agent selects the appropriate action from them according to its policy and this action is further processed into the HCU remaining control functionalities and sent to the environment to be executed.

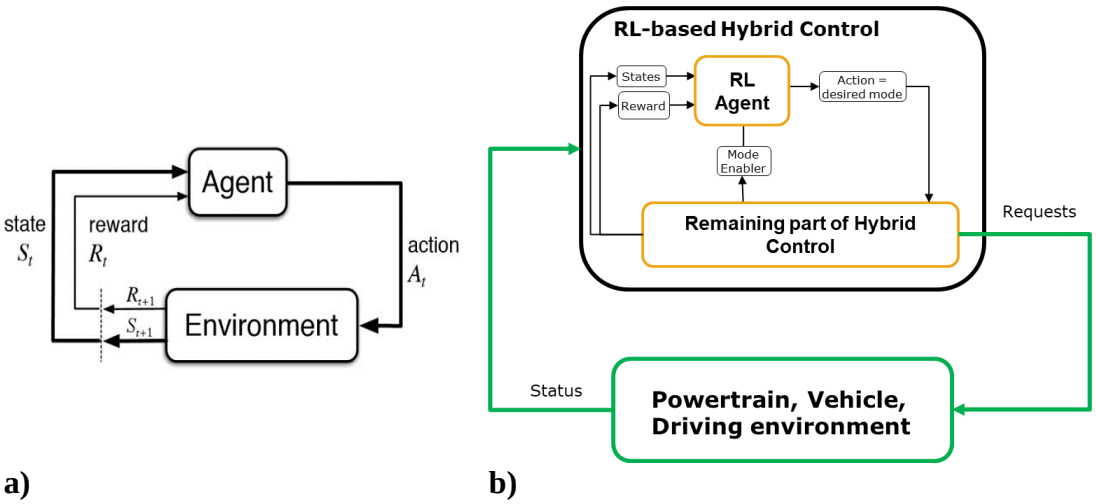


Figure 30 – a) The conventional and b) the proposed RL architectures, source: AVL DSP team.

4.2.4.3 REWARD FUNCTION

The reward signal represents the search objective of the RL problem where its proper selection directly affects the agent performance. At each training step, the agent selects an action to be executed in the environment and evaluated by returning a feedback scalar value called the reward. Accordingly, the agent can learn to incentivize the actions which led to higher rewards and inhibits the others led to lower rewards. The power of using RL in control systems is the ability to define multiple objectives to be maximized or minimized simultaneously in the reward function, hence its proper selection is a core task in the RL development cycle.

Various scholars researched the best battery depletion rate that brings better fuel economy to hybrid vehicles for a long time. They noticed that gradual battery SoC depletion, sustaining the battery charge over the entire trip, tends to provide a near-optimal performance for the EMS problem. Accordingly, space-domain indexed SoC reference is proposed to guide the SoC depletion rate gradually in the entire trip taking into account the prior knowledge of the trip distance [85–88]. The trip distance can be estimated from the navigation systems available nowadays in modern vehicles while Li et.al proposed a model called History Cumulative Trip Information (HCTI) to estimate its value with experimental proven accuracy [89]. The HCTI estimates the travel distance considering the historical record of the driver as a function of the day and time. The reward function defined utilizing the travel distance is shown in equation (4.33) where χ and φ are set to 40 and 36 respectively after careful tuning.

$$Reward = -tanh(\varphi \,.|SoC \; - \; SoC_{reference}| \; + \; \chi \,.\dot{m}_{fuel}) \tag{4.33}$$

The objectives considered in this thesis include driver comfort and vehicle drivability too. Hence, another term is added to equation (4.33) to inhibit the frequent engine start/stop switching to improve the vehicle efficiency with a smooth operational behavior as shown in equations (4.34) and (4.35).

$$Cost = tanh(\,\varphi \,.|SoC \; - \; SoC_{Optimal}| \; + \; \chi \,.\dot{m}_{fuel} + \; \psi \,.E_{switch}) \tag{4.34}$$

$$SoC_{Optimal} = SoC_{initial} \;.(1 - d/D) \; + \; d/D \;.SoC_{final} \tag{4.35}$$

χ and φ are using the same values and ψ is set to be 0.7 after tuning. E_{switch} is a binary value to indicate engine change state either starting or shutting down. d is the traveled distance while D is the total distance of the entire trip.

The negative sign is removed as the reward function is renamed to represent a positive cost function to be minimized by the agent. Two changes were done to compensate for such change, the first is to select the action that represents the minimum Q-value satisfying $a = argmin\, Q(s, a)$ instead of $a = argmax\, Q(s, a)$ in algorithms 3-5. The second change is to update the Q-value with $\gamma \min_a \hat{Q}(s_{j+1}, a;\, \theta^-)$ instead of $\gamma \max_a \hat{Q}(s_{j+1}, a;\, \theta^-)$ in the same algorithms too. The aforementioned changes bound the Q-network cumulative return estimates to be a positive value which showed improved performance incorporating the ReLU activation function in the hidden layers as will be shown and discussed in the results in Chapter 5.

5 RESULTS AND ANALYSIS

In this chapter, several experiments are performed to examine the proposed methodologies that were introduced previously in chapter 4. The driving cycles used through the whole chapter are introduced in section 5.1 while the simplified vehicle model validation is illustrated in section 5.2. RL agents are discussed in section 5.3 where separate designed experiments are conducted in sections 5.3.1- 5.3.4. The contribution of each proposed algorithmic module is tested and key conclusions are drawn. Based on such cognitions, an E-DQN is developed in section 5.3.5 and tested for generalization using two unseen cycles with different initial SoC levels.

5.1 BOUNDARY CONDITIONS

The driving cycles used in this chapter are NEDC, Highway Fuel Economy Driving Schedule (HWFET), Urban Dynamometer Driving Schedule (UDDS), and GRAZ cycle. HWFET is a highway cycle, UDDS is an urban cycle often called FTP-72 (FTP, Federal Test Procedure) while NEDC is usually used to assess the emission levels and fuel economy [90]. The three cycles are often used by scholars for research purposes.

However, the GRAZ cycle is a transient driving cycle defined by the AVL DSP team for internal use. It is a highly dynamic cycle due to several velocity and altitude fluctuations in each segment. GRAZ cycle is quite typical for long journey driving situations in Europe. It can be divided into four segments: city cycle, highway cycle, rural cycle, and another city cycle with around 53 km overall distance completed within 4445s (1.235 hours). The four-cycle velocity profiles are shown in Figure 31.

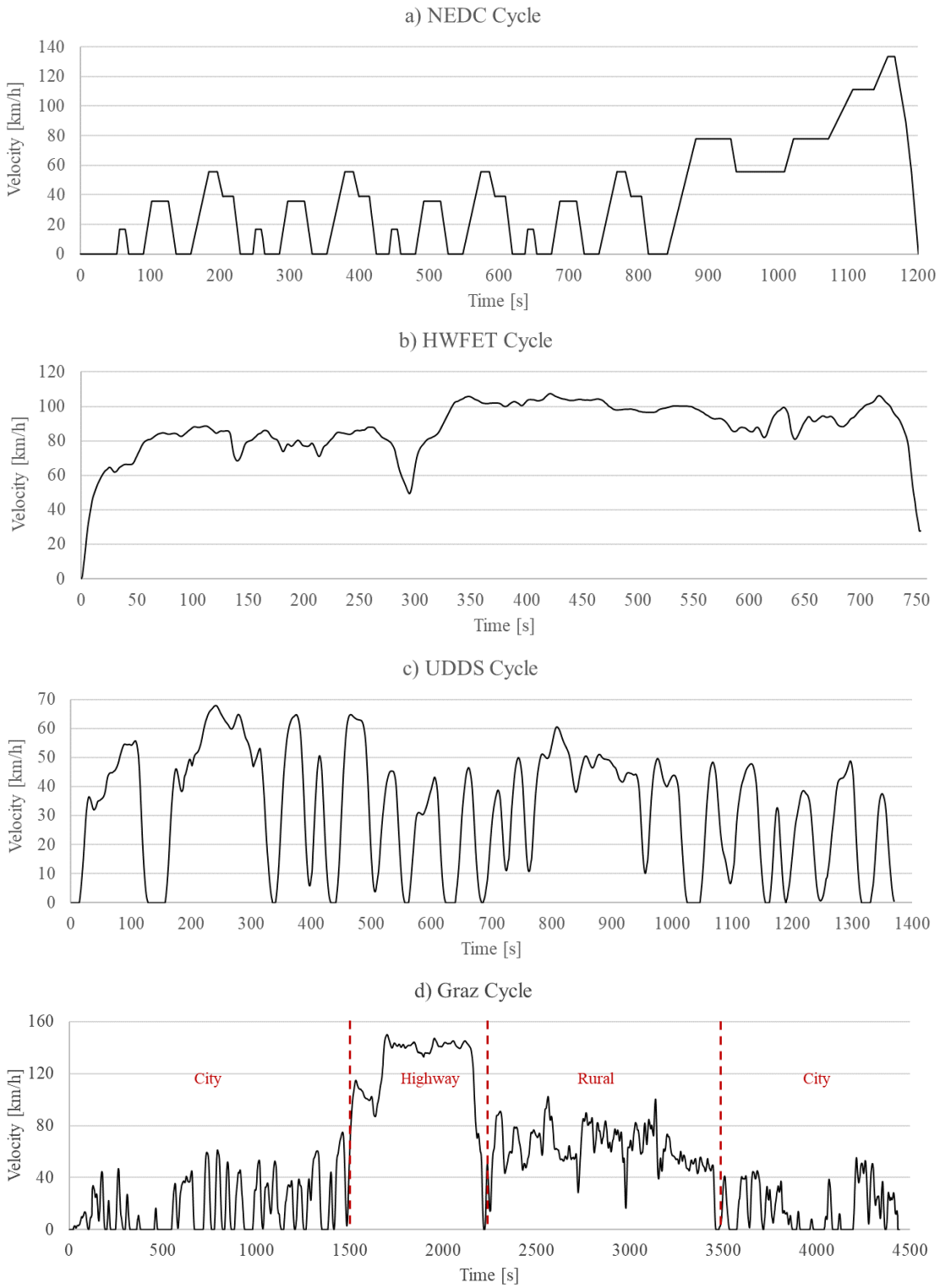


Figure 31 – The velocity profiles of the driving cycles used in chapter 5.

5.2 SIMPLIFIED VEHICLE MODEL VALIDATION

The accuracy of the simplified vehicle model influences the optimality of the trained RL control strategy because it directly affects the fuel consumption and SoC change estimation. Therefore, model validation results shall reflect and quantify the estimation error for fuel consumption and SoC change compared to the high-fidelity vehicle model. Driving cycle information such as speed and road gradient time series in addition to a sequence of selected driving modes following the CDCS strategy are used as inputs for both the simplified and the high-fidelity vehicle models.

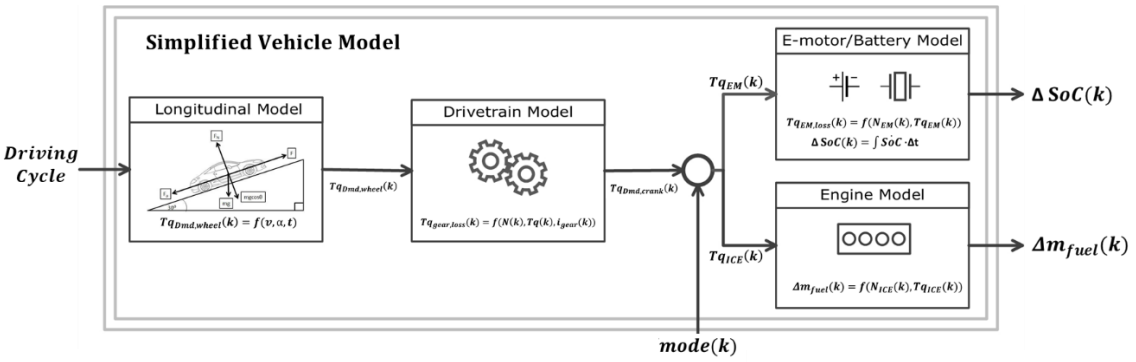


Figure 32 – Simplified vehicle model diagram.

Figure 32 shows the operational flow diagram of the model where the driving cycle information is used to calculate the wheel torque demand in the longitudinal model and then, the crankshaft torque demand is calculated in the drive-train model. Combined with the control strategy selected mode, detailed torque distribution between ICE and EM decision is determined. The torque demand from the ICE (Tq_{ICE}) and the EM (Tq_{EM}) are used in the Engine and EM/Battery models respectively to calculate the SoC change and the fuel consumption as model outputs.

The NEDC driving cycle shown in Figure 31a is used for the model validation. The AER of the considered P2 PHEV is 64 km while a single NEDC cycle has a distance of 11 km. Therefore, 6-repeated NEDC (6-NEDC) are simulated to cover the full battery charge deletion to the minimum threshold.

The model validation experiments begin with using the high-fidelity model to run the driving cycle with control actions following the CDCS strategy implemented in the HCU's calibration data and the rule-based controller provided by the AVL DSP team. The control inputs and model simulation outputs are logged to be implemented into the intended model to be validated. The SoC estimation is evaluated by running the 6-NEDC cycle with a full battery (95% SoC) while the fuel consumption estimation is evaluated better by running the same cycle with an empty battery (20% SoC). Both SoC depletion and fuel consumption validation experiment results are shown in Figure 33 and Figure 34 respectively.

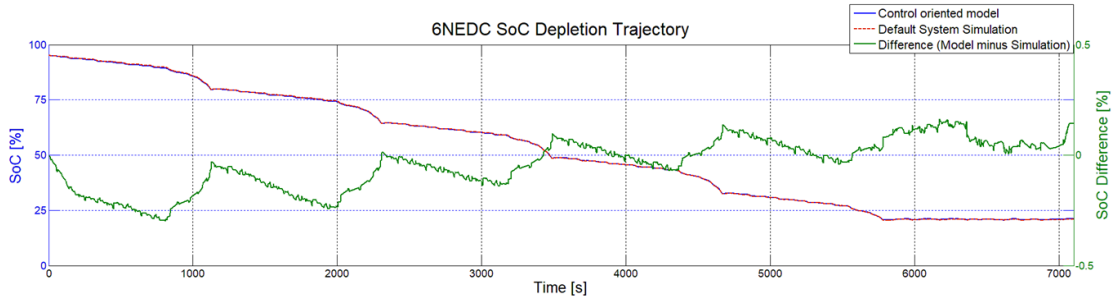


Figure 33 - SoC depletion trajectories comparison with a full battery in the 6-NEDC cycle, source: AVL DSP team.

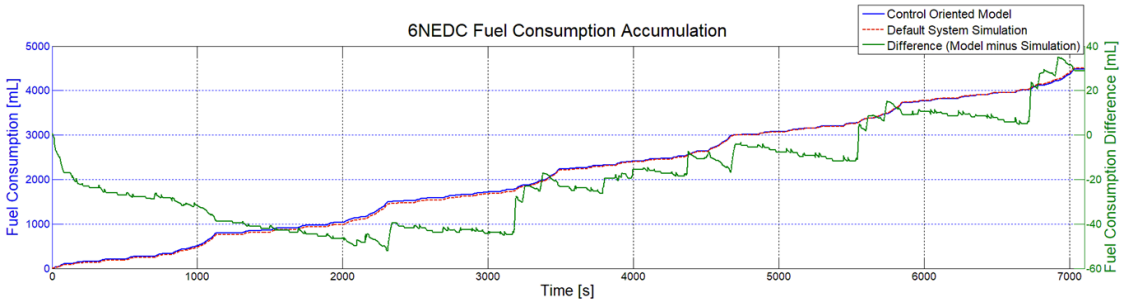


Figure 34 - Fuel consumption trajectories comparison with an empty battery in the 6-NEDC cycle, source: AVL DSP team.

The results assert that SoC and fuel consumption differences are negligible. The SoC difference oscillates between -0.3:0.2% while the fuel consumption difference ranges between 30:50 ml, which is 1.1% of the 4500 ml total fuel consumption. It is realized that the sudden dynamic transient activities such as acceleration, deceleration or gear shift, or

simplification of some parameters lead to the differences between both models. However, being all in an acceptable range, it is concluded that the simplified vehicle model is accurate enough to be used instead of the high-fidelity model for further experimentations.

5.3 REINFORCEMENT LEARNING AGENTS

5.3.1 TABULAR Q-LEARNING BASED EMS

Algorithm 2 showed how the tabular Q-learning method is incorporated into an RL agent to solve the energy management problem in the P2-PHEV. The environment state space (SoC, T_{tot}, V, D_{rem}, E_{on}) is continuous, therefore a discretization of (40, 20, 20, 20, 2) is applied resulting in a Q-table with dimensions of (640,000 x 3). The training hyperparameters are tuned after several experiments and summarized in Table 3.

Table 3 - Q-learning hyperparameters.

Learning rate α	0.01	Epsilon decay rate $d\varepsilon$	0.01
Discount rate γ	0.995	Epsilon minimum ε_{min}	0.1
Epsilon ε	1		

a)

b)

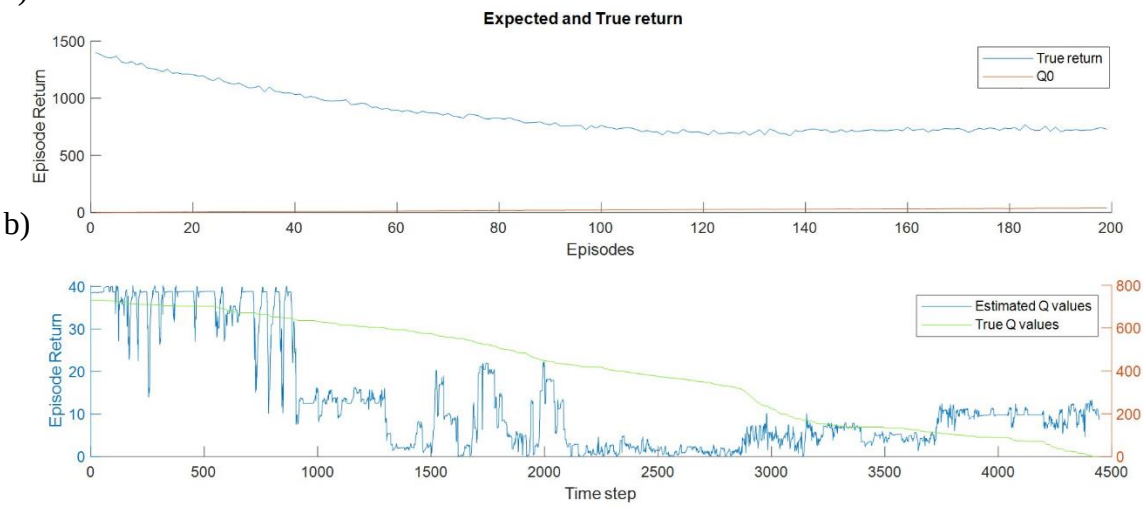


Figure 35 - Q-learning agent results on the GRAZ cycle.

Figure 35 demonstrates the Q-learning performance results on the GRAZ cycle with initial SoC of 75%. Figure 35a shows the episode cumulative return in addition to the Q0 which represents the estimated Q-value at the beginning of the episode. Q0 represents the agent's expected cumulative return and the closer it is to the true return, the better the agent can expect future rewards, and accordingly, the best actions can be taken to minimize the cumulative episode cost. Nevertheless, Q0 did not approach the true return value which reveals the agent does not correctly learn the environment.

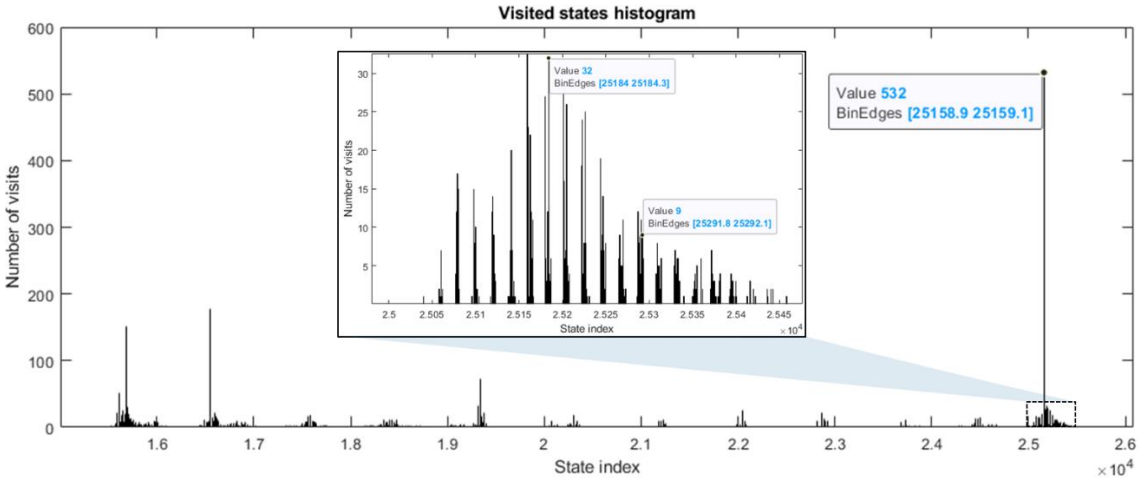


Figure 36 - Discretized states visit frequency per episode.

The Q-learning based agent suffers from the curse of dimensionality due to discretization [45]. Although the discretization level used causes the Q-table to be huge (640,000 x 3) and requires large memory to handle, it is not sufficient for proper segregation and differentiation between adjacent states.

The histogram in Figure 36 shows how frequently each state is visited during a single episode. The results reveal that the majority of the states are visited more than once up to 532 times. In discrete state-space environments such as the grid world, discretization is not applied hence each state-action pair holds a Q-value that fully represents the discounted cumulative return following the optimal policy till the episode end according to the Q-value definition. Although in the discretized continuous state space environments such as HEV's, the discretized states are visited more frequently with different Q-values which causes the Q-function to diverge and never approximate its true representation.

Figure 35b shows the true cumulative return and the Q-value for each time-step/state in the current episode. Proceeding with the agent training, the trend of the estimated Q-values shall follow the true return especially with small values of the exploration term ε. However, due to the coarse discretization and the curse of dimensionality problem, both curves never followed each other with different hyperparameters in several experiments.

The tabular Q-learning approach is concluded not to be suitable for future expanded applications with large state space environments such as HEVs. The deep Q-network approach offers a solution to this problem and avoids the state space discretization by introducing an NN outperforming the tabular Q-learning according to several papers discussed in section 3.2.2.5.

5.3.2 DEEP Q-LEARNING BASED EMS

Algorithms 3 and 4 with the hard and the τ-soft update, introduced in section 4.2.3, are experimented together running the GRAZ cycle with the following set of hyperparameters shown in Table 4. For the upcoming experiments, the same hyperparameters defined in Table 4 are used unless stated otherwise. The DQN network utilized has the critic representation discussed in section 4.2.2.2 with four hidden layers each having 32 neurons followed by a ReLU activation layer.

Table 4 - DQN-learning hyperparameters.

Learning rate μ	0.0025	Discount rate γ	0.995
Epsilon ε	1	n-steps	16
Epsilon decay rate $d\varepsilon$	0.005	Mini-batch size	32
Epsilon minimum ε_{min}	0.1	Experience buffer size	10,000
NN optimizer	ADAM	Experience buffer sampling	Random

The NN performance highly depends on the scale and dimensions of the input features either in the classification or more importantly in the regression problems as is the case with the DRL. ML researchers recommend pre-processing the input data to be normalized in a range of 0-1 or standardized with a zero mean and a standard deviation of 1 [91].

Data normalization would be more practical in the HEV problem domain especially in deploying the RL agent in a test driving cycle with unknown inputs mean or standard deviation. Therefore, inputs normalization is used in state-action pair Q-value estimation and during the critic network training process with inputs and normalization ranges shown in Table 5.

Table 5 – DQN inputs and normalization ranges.

Input	**Range**	**units**	**Normalization range**
State of Charge SoC	20:95	%	0: 1
Torque demand T_{tot}	-90:310	N.m	
Velocity V	0:44	m/s	
Remaining distance D_{rem}	0:160	km	
Engine run E_{on}	0:1	-	
Action a	0:8	-	

The results are shown in Figure 37 and Figure 38 disclose how each agent interacts with the environment and learns differently. The DQN agent has a single network that is used to update the policy and evaluate the target value from the same network. Previous studies did not recommend a single network DQN for learning and training stability issues. However, in our problem domain, it behaves stably and properly.

The DDQN with hard update synchronized the target network with the policy network in every 10 episodes in the aforementioned experimental setup. It is clear in Figure 38b how slow it is for the Q0 to be close to the cost and to keep tracking it steadily without oscillations. Nevertheless, the DDQN with τ-soft update showed a quicker convergence and a stable behavior after episode 150. Starting from episode 200, the Q0 was almost kept constant which indicates that the Q-function captured a proper approximation to the true return value.

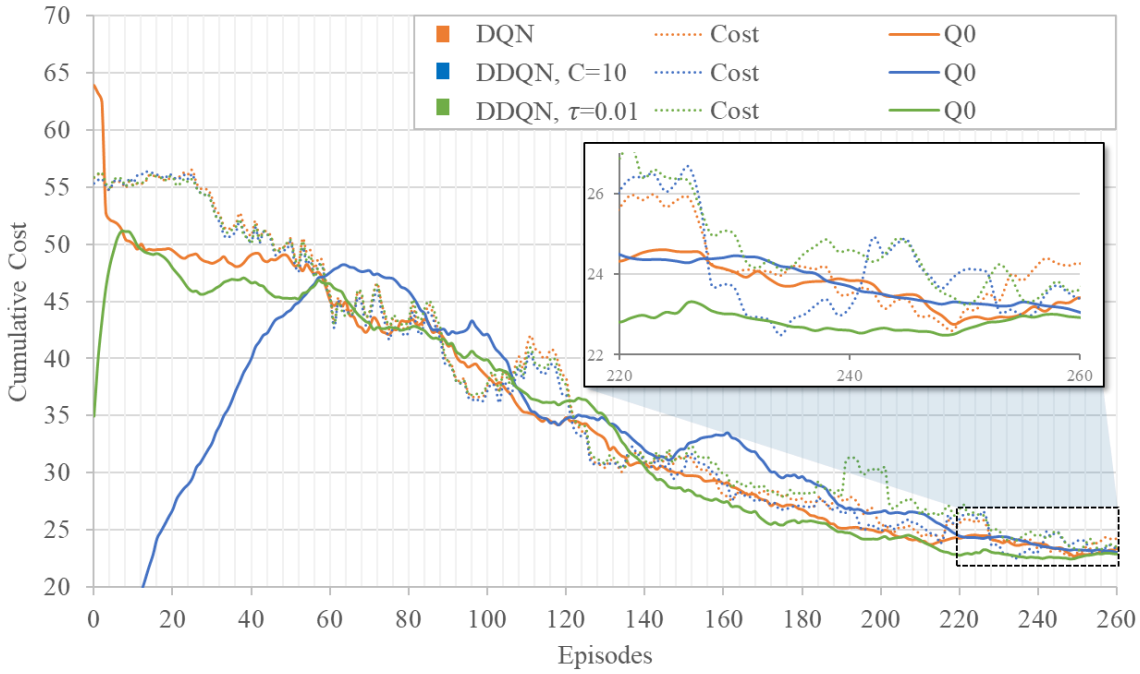


Figure 37 - DQN, DDQN with hard, and soft update results.

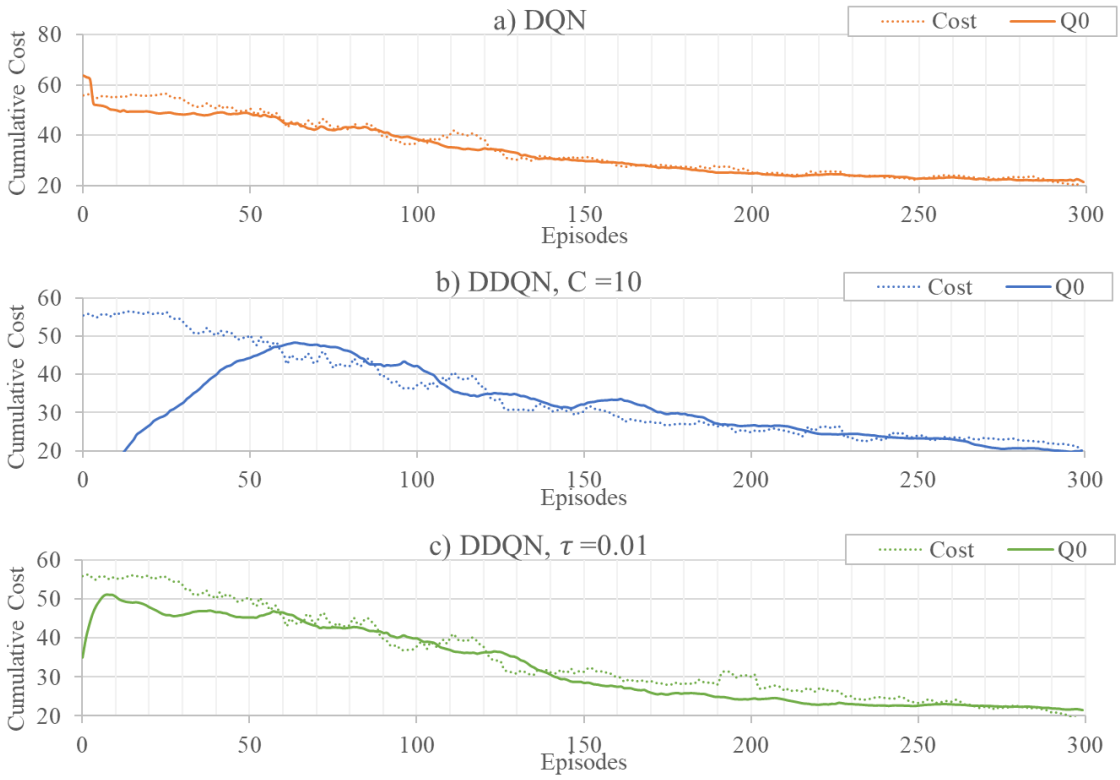


Figure 38 - Detailed view of DQN, DDQN with hard, and soft update results.

It is worth mentioning that the converging Q0 should not match the true return because the epsilon value still causes some randomness in the action selection while the Q0 represents the expected return following the optimal policy completely. This was not the case with the DQN which followed the true return blindly from episode 200 as shown in Figure 38a. Accordingly, DDQN with τ-soft update is recommended for upcoming experiments due to its improved performance in comparison with the other two agents.

5.3.3 DRL IMPROVEMENT TECHNIQUES

5.3.3.1 NEURAL NETWORK ARCHITECTURE OPTIMIZATION

The subsequent experiment is to optimize the DRL critic and powertrain model NN architectures using the supervised learning approach. Several designs were tested including uniform, pyramid, and inverse-pyramid architectures. The number of hidden layers and the number of neurons in each layer are changed to cover a wide range of possible designs.

The experiment data set consisted of 50,000 training instances. The experiment ran three times, each time using a group of 2,000, 10,000, and 50,000 of the training instances respectively. Each group uses 80% of the available dataset to train the NN and 20% to test the generalization error. The experiment was conducted in this way to find the best architecture that minimizes the generalization error in each group. Each group represents a phase in the agent learning process and for the sake of better convergence, the NN shall be powerful to represent the agent knowledge in the early stage of the training such as group one, and the later stages such as group three.

Taking into consideration the weighted contribution corresponding to the number of the training instances in each group in the design selection criteria, Figure 39 showed that two designs achieved the minimum generalization error: [512 512] and [32 32 32 32]. The first design has a total number of 1024 neurons in comparison to 128 neurons in the second design which means the latter achieves 8x faster training with lower memory requirements. Therefore, the second design (four layers with 32 neurons each) is selected for the thesis experiments.

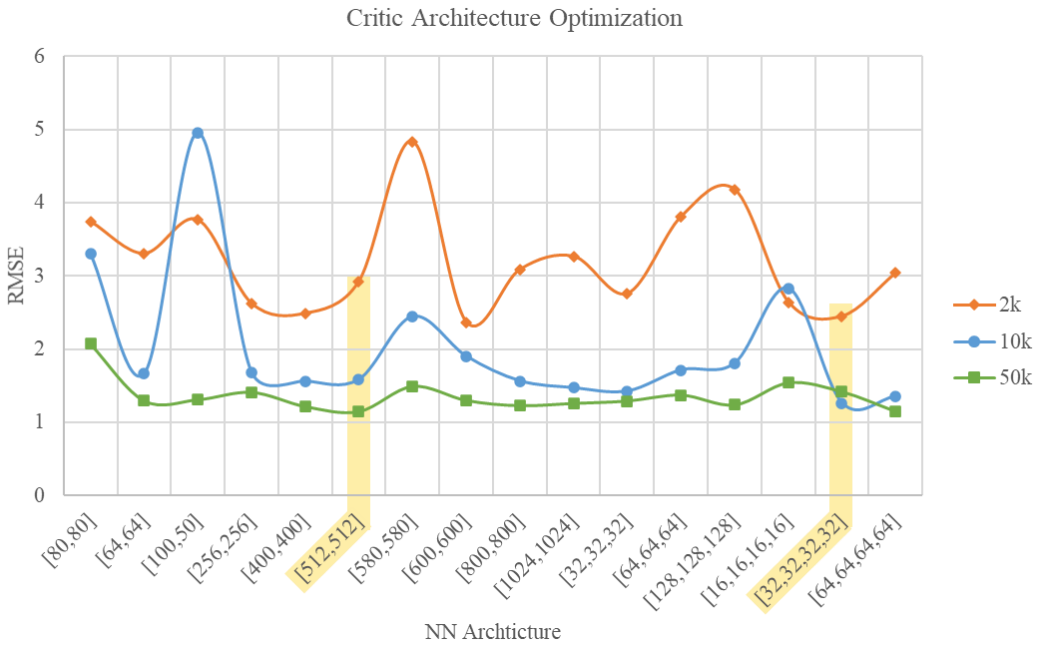


Figure 39 - DRL critic architecture optimization results.

The same approach is used to optimize the powertrain model NN concluding that [5,4] and [8,8,8] are performing the best among the other models. The latter design (three hidden layers with 8 neurons each) is selected due to the more representative power available in the model especially since both models are simple and efficient in computational power with low memory requirements.

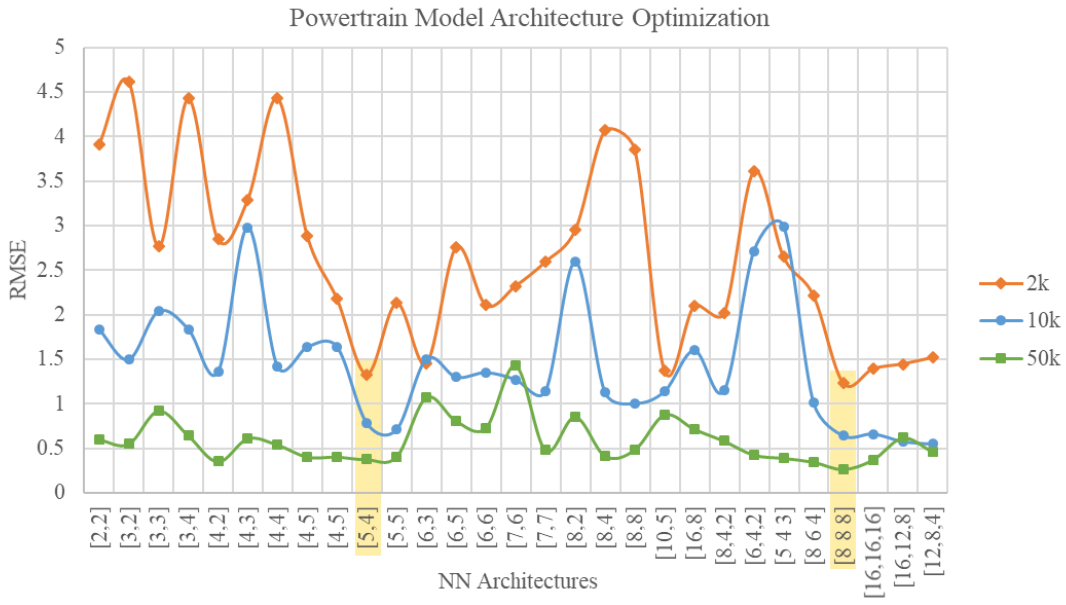


Figure 40 – Powertrain model architecture optimization results.

5.3.3.2 N-STEPS BOOTSTRAPPING

Four agents were tested using the hyperparameters defined in Table 4 and varying the n-steps each as shown in the combined plot in Figure 41. Figure 42a, representing the TD(0) agent with n=1, had a diverging Q-function, however, it started to converge very slowly using n=2 as the case in Figure 42b.

Better performance is achieved by setting n=16 as shown in Figure 42d in the contrary to the recommendations previously discussed in 4.2.3 of setting the n-value to be small around 4 for better off-policy behavior. However, it is noticed that decaying the n-value from 16 to 6, by subtracting 1 every 10 episodes, showed the best performance as demonstrated in Figure 42c. After episode 100, the n-value was 6 which helped the agent to utilize the experience available in the replay buffer to learn more efficiently for 80 more episodes. The Q0 value started to stabilize after episode 100 even with oscillations due to the randomness in the action selection and the cost tried to come closer and closer by proceeding in the training. In the combined plot in Figure 41, it can be concluded that the decaying n-steps agent is outperforming the other agents either in the achieved cost or the Q-function convergence to a lower and more stable value.

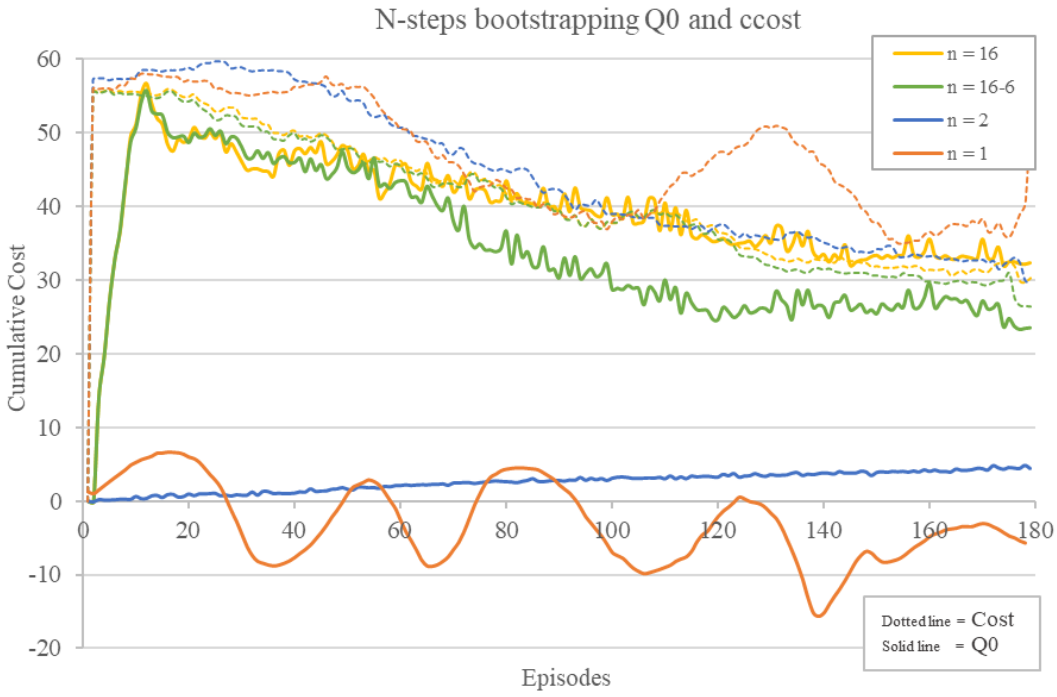


Figure 41 - n-steps bootstrapping combined plot.

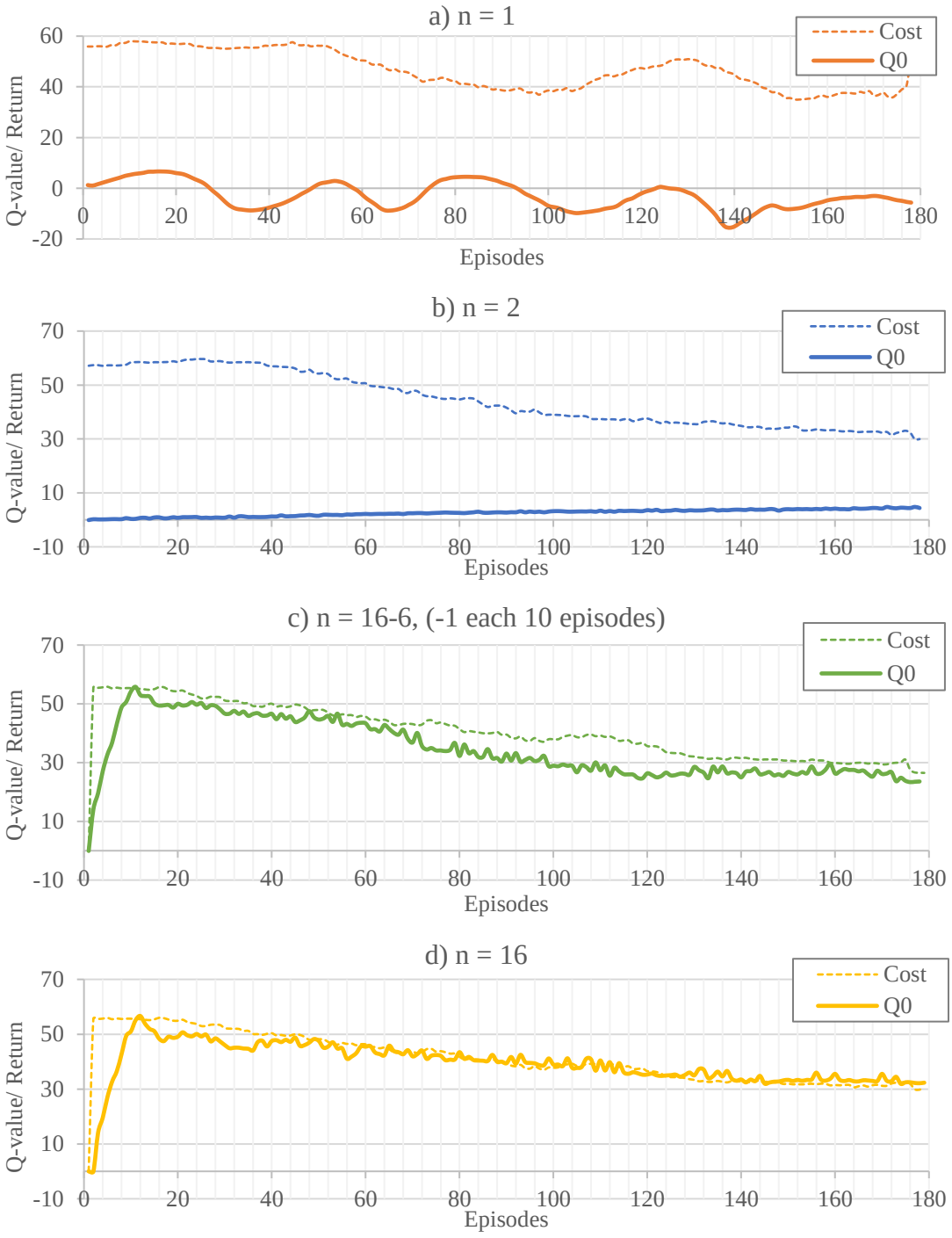


Figure 42 - n-steps bootstrapping results in DRL.

5.3.3.3 PRIORITIZED EXPERIENCE REPLAY

The following experiment focused on examining the performance of the Prioritized Experience Replay (PER) compared to the random sampling experience replay (referred to in the figures as ~PER). The α and ε values introduced in equation (4.29) are selected to be 0.6 and 1e-10 respectively. However, according to Schaul et al., the β value used in equation (4.30) is started with a value of 0.4 and increased gradually to 1 with a rate of 0.005 which is the same rate of the epsilon decay shown in Table 4 [78].

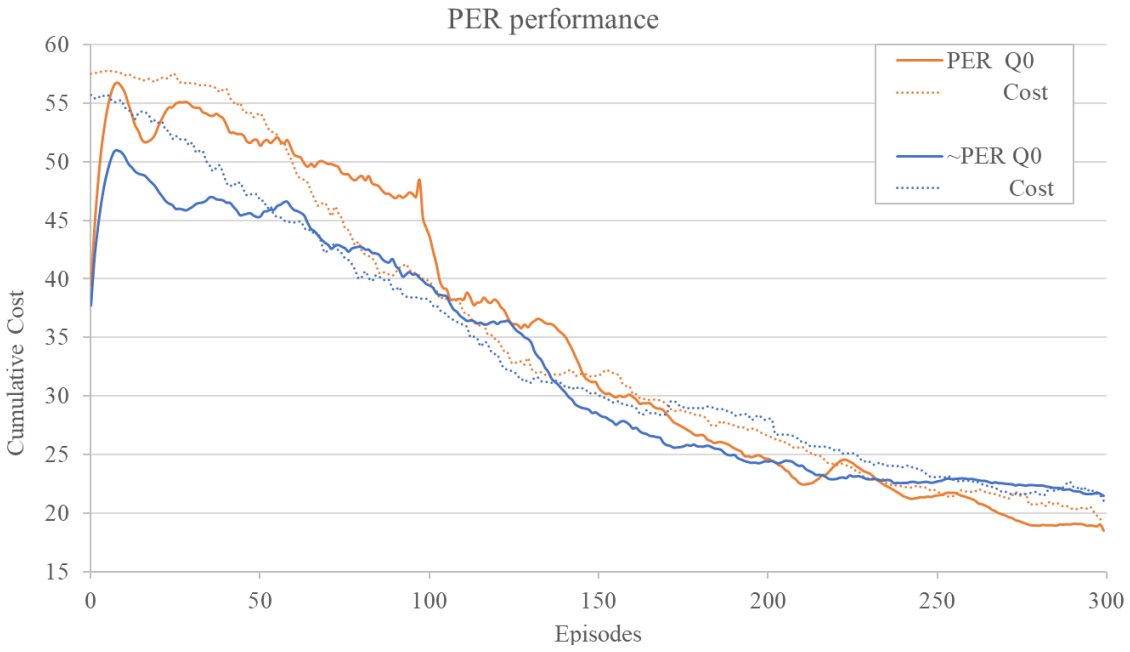


Figure 43 - Prioritized sampling performance compared to random sampling in the experience replay.

The total and expected cost of both agents in addition to the training loss are shown in Figure 43 and Figure 44 respectively. The effect of prioritization became significant after episode 90 where the prioritized experience instances are considered more for the training which caused the MSE to increase rapidly as shown in Figure 44. On the contrary, the PER agent outperformed the other starting at episode 170 by achieving a lower cost and a decreasing MSE till the end of the training which reflects converging to a better policy.

Figure 44 – the MSE of prioritized sampling compared to random sampling in the experience replay buffer.

In conclusion, prioritized sampling offers improved performance over the random sampling-based experience replay buffers with slightly increased computational resources. PER is recommended for later use in the RL agent training for the P2-PHEV domain.

5.3.4 HEVs SPECIAL CONSIDERATIONS

5.3.4.1 ACTION MASKING

Incorporating action masking into the RL domain is illustrated in Figure 30b. The remaining HCU part is modeled by rule-based logic using the HCU's calibration parameters to provide the RL agent with a mode enabler vector. Such a vector enables only the available actions each time-step making sure that the RL agent cannot behave in an unsafe or unrealistic way.

Fixed modes such as AddBoost, Recup, and St/Sndtl modes are shown in Figure 45 in orange, blue, and yellow colors respectively. In the plot background, the velocity profile is plotted where it can be noticed that St/Sndtl mode is selected properly at zero velocity. The driver torque demand is plotted in the background in a silver color with a biased x-

axis to show the negative values. The negative torque corresponds to the Recup mode selected while the AddBoost mode is selected often at the torque peaks especially at low velocities.

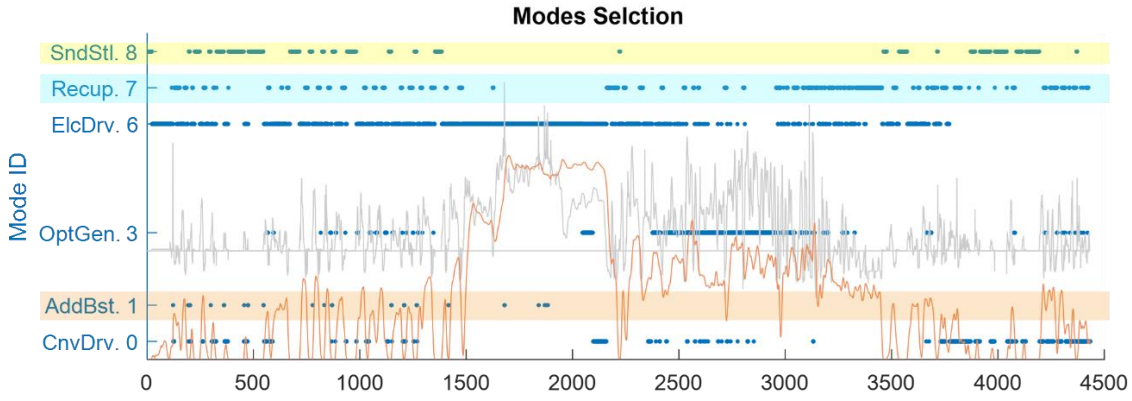


Figure 45 - Action masking for fixed modes selection.

5.3.4.2 MODEL-BASED RL

Algorithm 5 showed the details of the model-based RL agent considered in our P2-PHEV problem. Running a few experiments showed that the model-based agent has higher sensitivity for the learning rate. Therefore, a lower learning rate of 0.0001 is used in the experiment shown in Figure 46.

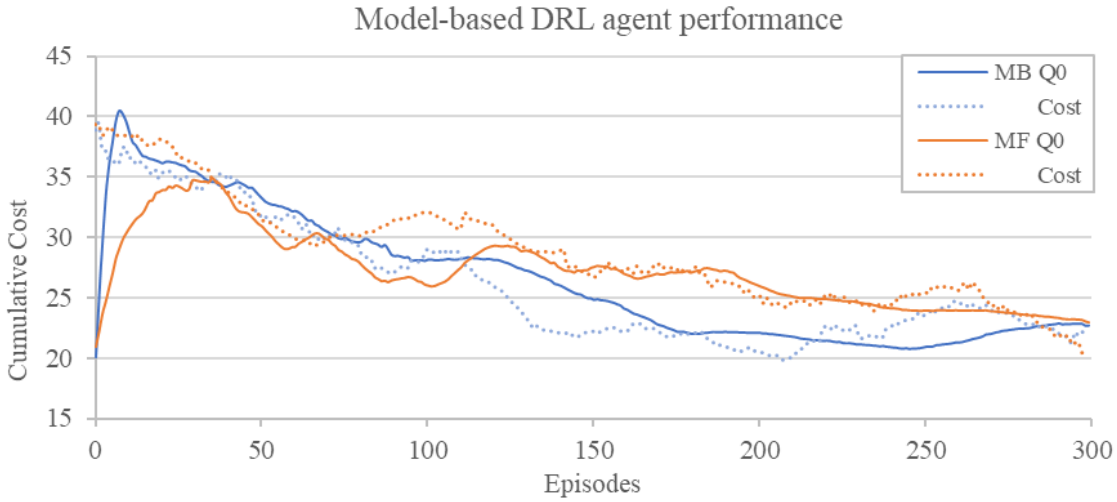


Figure 46 - Model-based and model-free DRL agents' comparison.

The results in Figure 46 showing that the model-based RL agent captured a closer value to the Q-function quicker than the model-free RL agent. The latter showed better

adherence and tracking to the training cost while the first outperformed in terms of the achieved cost between episode 80 till episode 265. Thereafter, both agents behaved similarly achieving approximately the same cost and a very close final Q0 value for each.

It is concluded that the model-based RL can benefit from the known model dynamics to quickly converge to the optimal Q-value while the model-free RL needs a longer time. However, this conclusion is valid in the early stages of the learning process when the Q-function approximation is poor as shown in Figure 46 from episode 80 till episode 200. In later stages, both approaches behave similarly while the model-free approach would be favored due to its generality and optimality characteristics in addition to the lower computational resources required. The aforementioned conclusion agrees with the results of Gu et.al [82]. They recommended using the imagination rollouts in the early stages of the training to improve the Q-function approximation, then turning off the model-based planning later. The fine-tuning in later stages is better to be done by the model-free RL using real-world experiences to assure the optimality and generality of the resulting control policy.

5.3.5 E-DQN AGENT FOR THE P2-PHEV

The following section combines the previous conclusions and results into an extended version of the DQN agent (E-DQN). Experimental results are shown for the agent training and testing on the HWFET cycle, and the generalization is tested using the GRAZ and the 6-UDDS cycles with different initial SoCs for better estimating the agent behavior on new cycles. E-DQN is a model-free RL agent combined with prioritized experience replay, decaying n-steps bootstrapping, and τ-soft update.

5.3.5.1 TRAINING PERFORMANCE

The training is done based on the hyperparameters defined in Table 4. Some hyperparameters are modified and shown in Table 6. The change imposed in this agent training was to decay the learning rate linearly with a value of 0.0005 every 30 episodes. Additionally, the τ value is increased by 0.01 every 20 episodes to get closer to the DQN behavior by proceeding in the training. Based on the conclusion of the DQN and DDQN experiment

with results shown in Figure 38, increasing the τ value as the training proceeds was noticed to improve the convergence and achieve better results.

Table 6 – E-DQN training hyperparameters.

Learning rate μ	0.001	Tau τ	0.01
Learning rate decay $d\mu$	0.0005	Tau growth rate $d\tau$	0.01
Learning rate update frequency	30 eps.	Tau update frequency	20 eps.
Learning rate minimum μ_{min}	1e-05	n-steps	16 - 6
Epsilon ε	1	Experience buffer size	10,000
Epsilon decay rate $d\varepsilon$	0.005	Experience buffer sampling	Prioritized
Epsilon minimum ε_{min}	0.1	NN optimizer	ADAM

The agent training results are shown in Figure 47. The average cost was noticed to be decreasing which indicates an improved policy while the Q0 became close after episode 120 which reflects building proper knowledge about the environment over time.

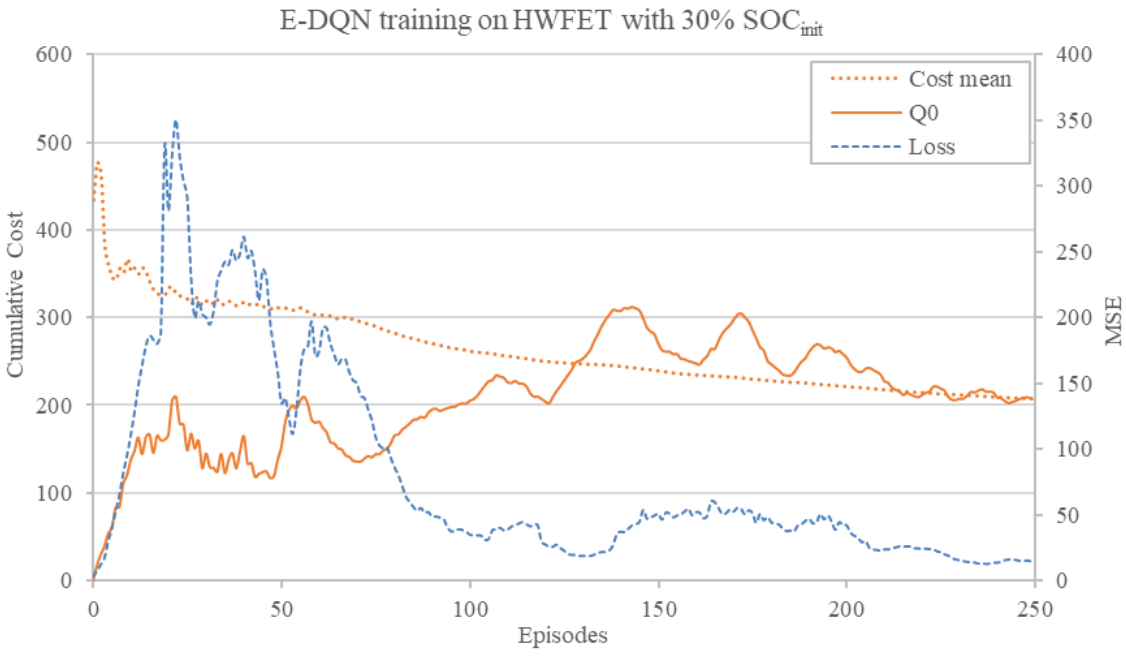


Figure 47 - the training performance of E-DQN agent on the HWFET cycle with 30% SoC$_{init}$.

After 250 training episodes, the agent behavior was simulated which means no more neural network training and the action decision is following the policy completely with no

randomness. The simulation results are summarized in Table 7 while the SoC depletion and fuel consumption trajectories for CDCS, DP, and RL are shown in Figure 48.

Table 7 – E-DQN simulation results on HWFET with 30% SoC_{init}.

HWFET Cycle SoC_{init} = 30%	P2-HCU Existing Strategy	DP-based solver Strategy	RL-based solver Strategy
Final SoC (%)	20.69%	20.71%	20.62%
Fuel Consumption (ml)	652.95	631.41	641.04
Fuel Economy Improvement	96.59%	/	**98.47%**
Number of engine-starts	7	50	**37**

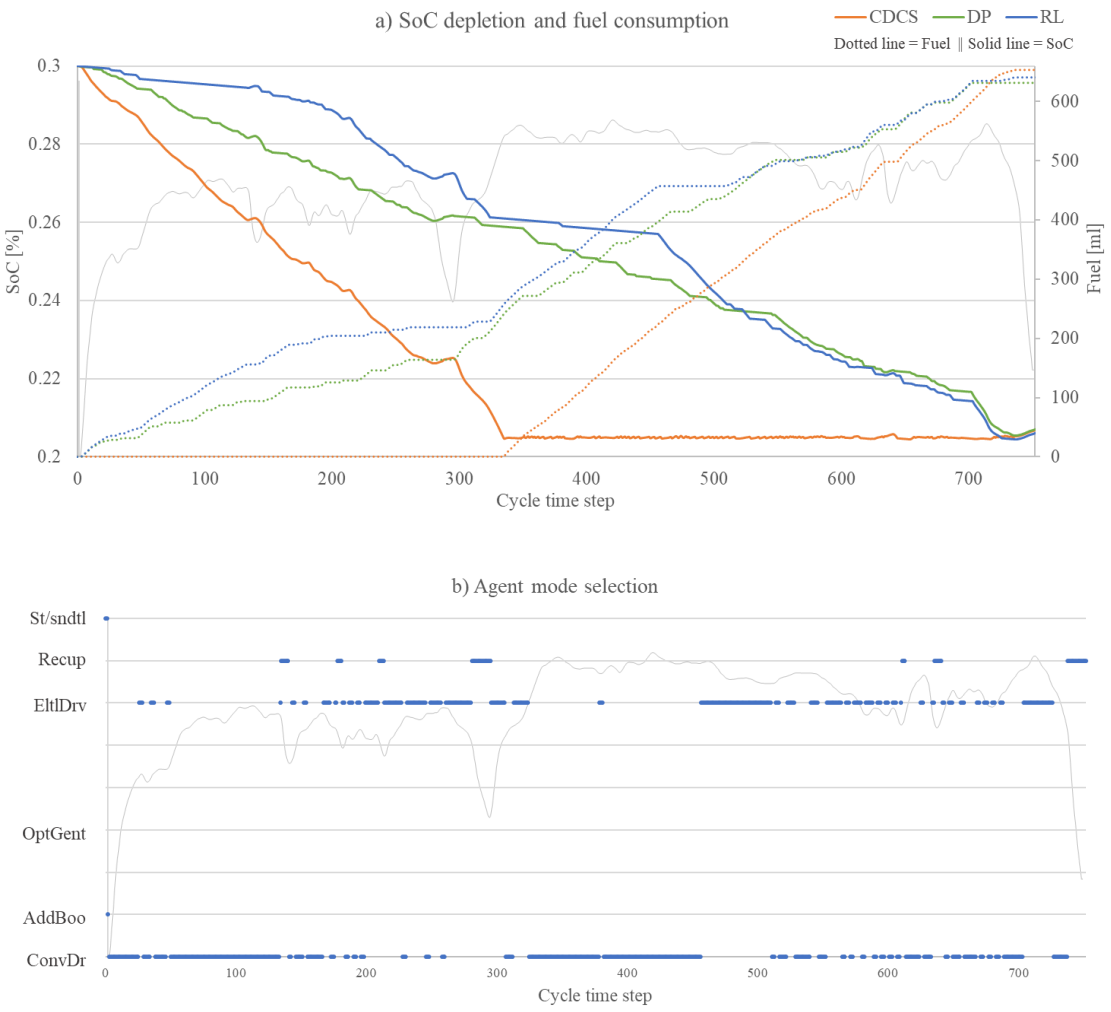


Figure 48 - E-DQN simulation on HWFET with 30% SoC_{init}.

The velocity is shown in Figure 48's background not-to-scale to provide a better understanding of the agent behavior. It is noticed that in the high-speed segments such as 50-130s and 320-460s, the agent mostly selected the ConvDrv mode to utilize the engine in the most efficient load points while the EltlDrv is selected mostly in the relatively low-speed segments.

The DP utilized a linear depletion of the SoC while the RL followed a different strategy achieving 98.47 % of the DP performance in comparison to 96.59% in CDCS. The difference between the CDCS and RL is not significant in the HWFET cycle because the whole cycle is in a high-speed range which means the optimum engine operation zone. Accordingly, the trained agent shall be simulated on other cycles to test its generalization capability and robustness.

5.3.5.2 GENERALIZATION PERFORMANCE

The E-DQN agent's generalization behavior is simulated using the 6-UDDS and the GRAZ cycles. The first cycle repeats the UDDS cycle six times to provide a longer distance outside the considered vehicle AER so that full battery depletion is possible. The fuel consumption and SoC trajectories for 75%, 50%, and 25% SoC_{init} levels, engine BSFC maps for 75% SoC_{init} for both cycles, and the numerical results summarized in Table 8 are used to judge the agent generalization performance for different environmental conditions.

Different SoC_{init} levels highly affected the performance of the E-DQN agent compared to the CDCS and DP as shown in Figure 49. With 25% SoC_{init}, the agent almost followed the CDCS strategy due to the lack of available electric energy onboard, therefore proper energy utilization is not possible. However, at higher SoC_{init} levels such as 50% and 75%, the RL agent was closer to the DP behavior rather than the CDCS by depleting the electric energy wisely throughout the whole trip. This enabled the RL agent to operate the ICE on optimal load points as shown in Figure 50 achieving 98.7%, 97.5%, and 93.7% of the DP fuel economy compared to 97.7%, 90.5%, and 83.3% following the CDCS strategy for the 25%, 50% and 75% SoC_{init} levels respectively.

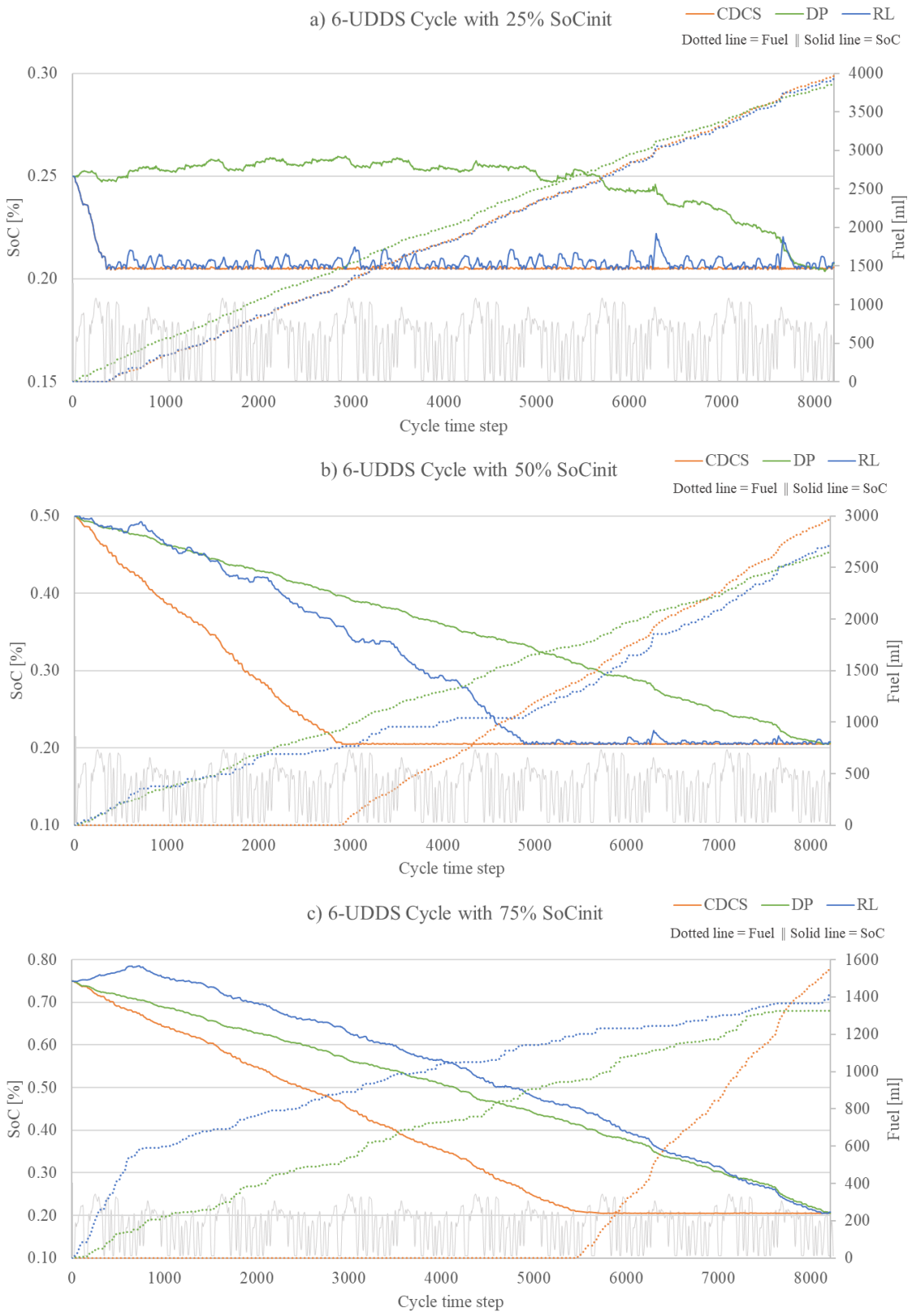


Figure 49 - E-DQN performance on 6-UDDS cycle with 25%, 50%, and 75% SoC_{init}.

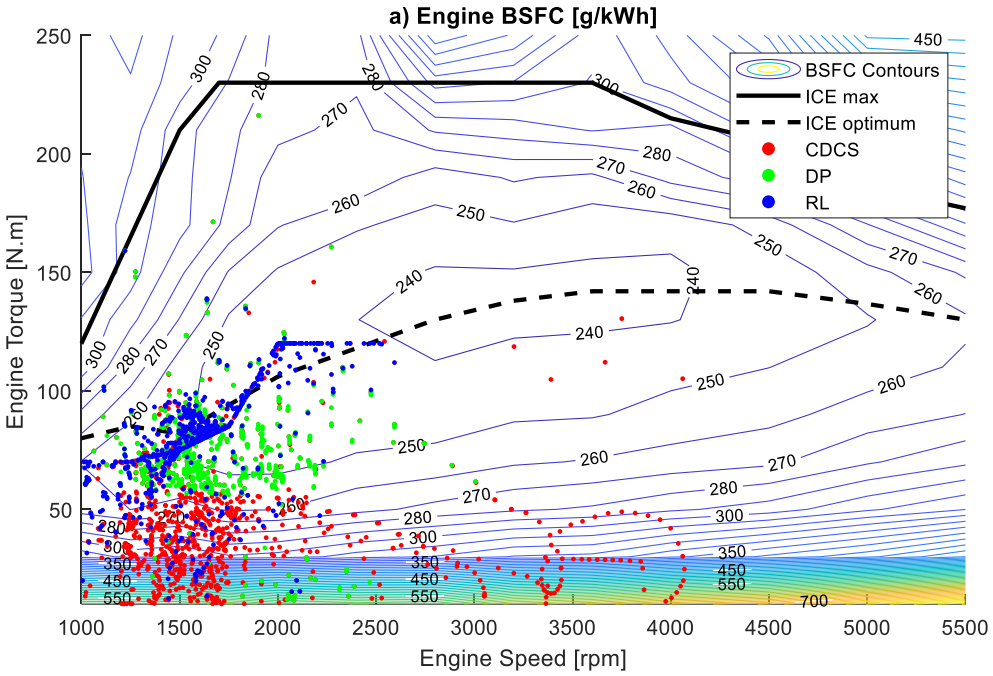


Figure 50 - Engine BSFC map for the 6-UDDS cycle at 75% SoC_{init}.

Figure 50 shows the improvement of the E-DQN agent's strategy over the CDCS compared to the DP. The ICE optimum operation line represents the most fuel-efficient operation load point at each engine rotational speed. The closer the agent to the ICE optimum line, the more fuel it saves. The RL agent achieved way better load points distribution over the CDCS which interprets the 10.46% improvement in the fuel economy.

For further testing and validation of the E-DQN agent capabilities, it is deployed to the GRAZ cycle previously shown in Figure 31d. The agent performance on the three SoC_{init} levels is demonstrated in Figure 51. The CDCS strategy obtained 98.2%, 94.5%, and 86.4% of the DP fuel economy for the 25%, 50%, and 75% SoC_{init} levels respectively while the RL agent outperformed the CDCS in all SoC_{init} levels by 99.2%, 96.6%, and 94.3% of the DP performance.

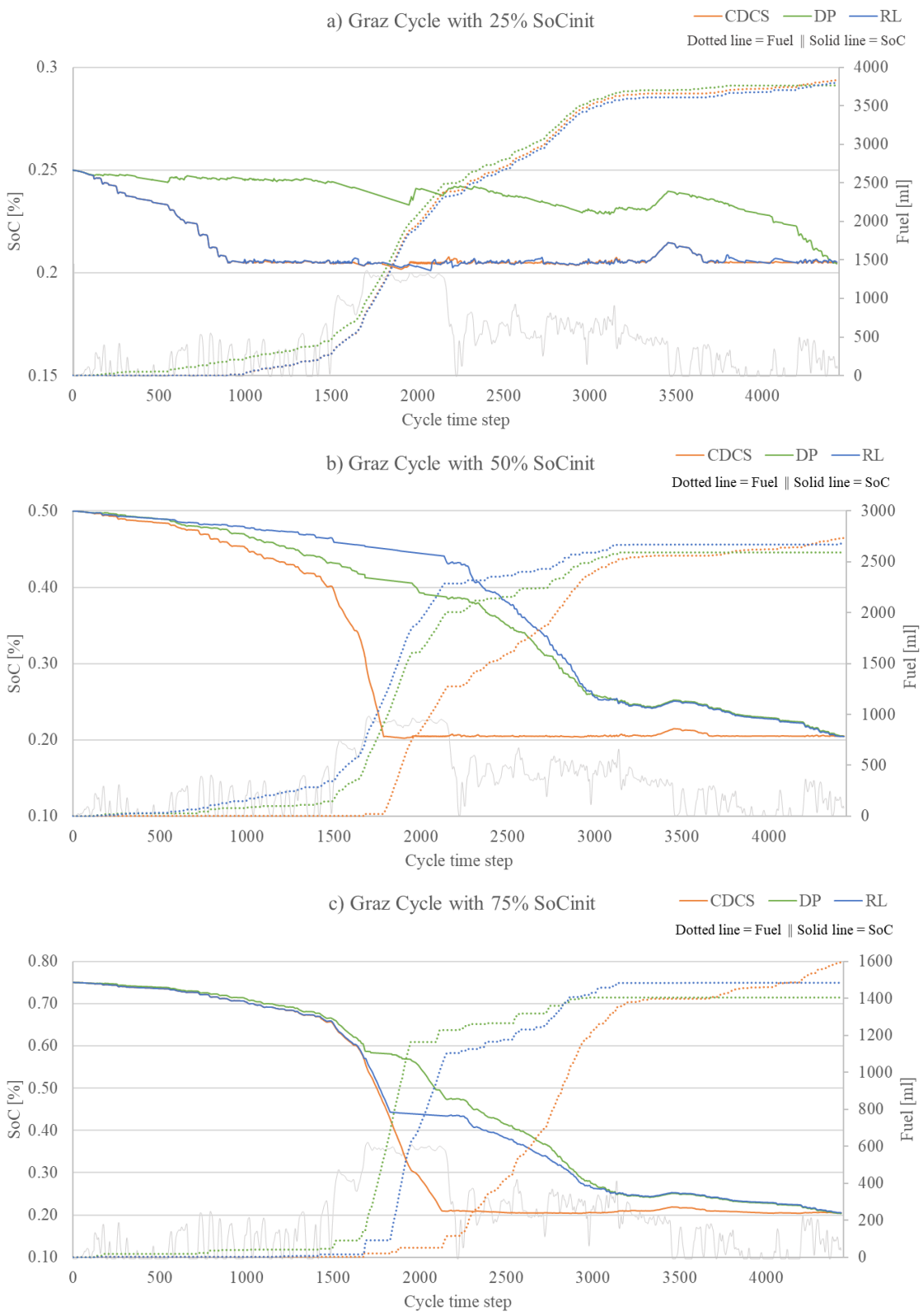


Figure 51 - E-DQN performance on the GRAZ cycle with 25%, 50%, and 75% SoC_{init}.

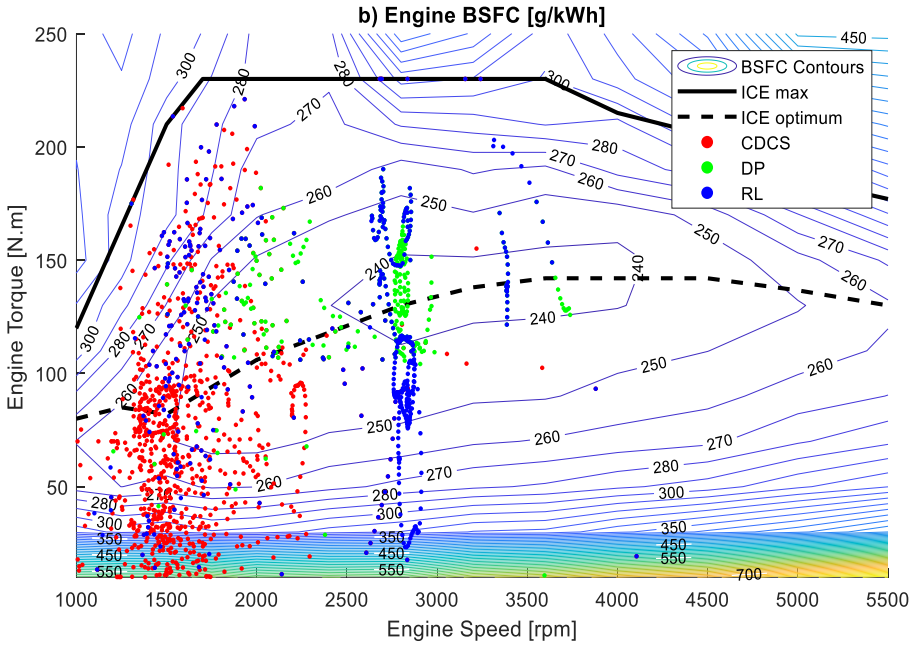


Figure 52 - Engine BSFC map for the GRAZ cycle at 75% SoC_{init}.

For a better understanding of the results, the GRAZ cycle with 75% SoC_{init} is selected as an example to compare the mode selection for the three strategies as illustrated in Figure 53. The CDCS strategy in Figure 53a started depending on the ICE from the 2400s where SoC begins to be sustained. On the contrary, the DP fuel consumption trajectory in Figure 51c dramatically increased between the 1700s to 2000s by selecting ConvDrv. The results reveal that the DP depends more on ConvDrv mode in the highway and rural segments where the velocity is relatively high. Moreover, EltlDrv is used instead for the other cycle segments where the velocity is relatively low.

The main unique strategy of the DP is to use the ICE more in the high-speed segments and depends completely on the electric drive for all the other low-speed segments. Such a strategy strongly concurs with the ICE system characteristics of working more efficiently at high velocities as illustrated in Figure 52 showing the ICE load point locations on the BSFC map. The DP located the load points mostly in the central region within 250 g/kWh BSFC while the ICE load points in the CDCS strategy are located on the left region, which is much less optimal. In addition to the ICU utilization, DP tended to

deplete the SoC gradually through the entire trip which coincides with the conclusion derived previously in section 4.2.4.3.

Figure 53 - Mode selection in GRAZ cycle with 75% SoC_{init}.

On the other side, the RL used a different strategy of prioritizing the EltlDrv mode always till the 1850s where a quick drop in the SoC takes place as shown in Figure 51c. the ConvDrv mode is used instead till the end of the highway segment where the EltlDrv is prioritized again after the 2400s. The ConvDrv mode is used for some segments later to maintain the SoC depletion trajectory within a proper rate while trying to minimize the number of engine starts to minimize the total episode cost. Figure 52 shows the engine BSFC map for the GRAZ cycle at 75% SoC_{init} where the adjacency between the DP and

the RL is closer than between the RL and the CDCS confirming the cognition of better engine utilization and higher fuel economy.

The results in Table 8 validate the surplus of the E-DQN agent compared to the CDCS strategy particularly in the high SoC_{init} levels where much electric energy is available and needs proper utilization. The RL agent outperformed the CDCS with more than 10% and around 8% closer to DP performance in the 6-UDDS and the GRAZ cycles respectively with 75% SoC_{init}. However, its performance was very close to the CDCS by less than 1% improvement with 25% SoC_{init} in both cycles.

Table 8 - E-DQN generalization performance results on 6-UDDS and GRAZ cycles.

		6-UDDS Cycle			**GRAZ Cycle**		
		25%	**50%**	**75%**	**25%**	**50%**	**75%**
Final SoC (%)	**CDCS**	20.6%	20.5%	20.5%	20.5%	20.5%	20.5%
	DP	20.8%	20.6%	20.6%	20.4%	20.5%	20.4%
	RL	20.8%	20.8%	20.8%	20.5%	20.4%	20.5%
Fuel Consumption (ml)	**CDCS**	3962.8	2907.3	1547.6	3788.5	2676.2	1595.3
	DP	**3873.5**	**2656.1**	**1326.4**	**3804.0**	**2592.6**	**1404.2**
	RL	3924.2	2721.7	1409.7	3797.3	2681.3	1483.8
Fuel Economy Improvement	**CDCS**	97.70%	90.54%	83.32%	98.20%	94.50%	86.39%
	DP	-	-	-	-	-	-
	RL	**98.69%**	**97.53%**	**93.72%**	**99.17%**	**96.58%**	**94.33%**
Number of engine starts	**CDCS**	**34**	72	**60**	**66**	72	**32**
	DP	308	320	210	103	76	53
	RL	62	**66**	106	68	**33**	50

Moreover, the number of engine starts for the RL agent was kept in an acceptable range with an average of one engine start each 96.3s for the GRAZ cycle compared to one start

each 61.8s and 89.3s in the DP and CDCS respectively. However, the CDCS achieved longer time for each engine start with an average of 155.4s in the 6-UDDS compared to 30.5s and 105.5s for the DP and the RL respectively. The previous results are shown in Figure 54 relative to the DP in a bar chart for easier comparison.

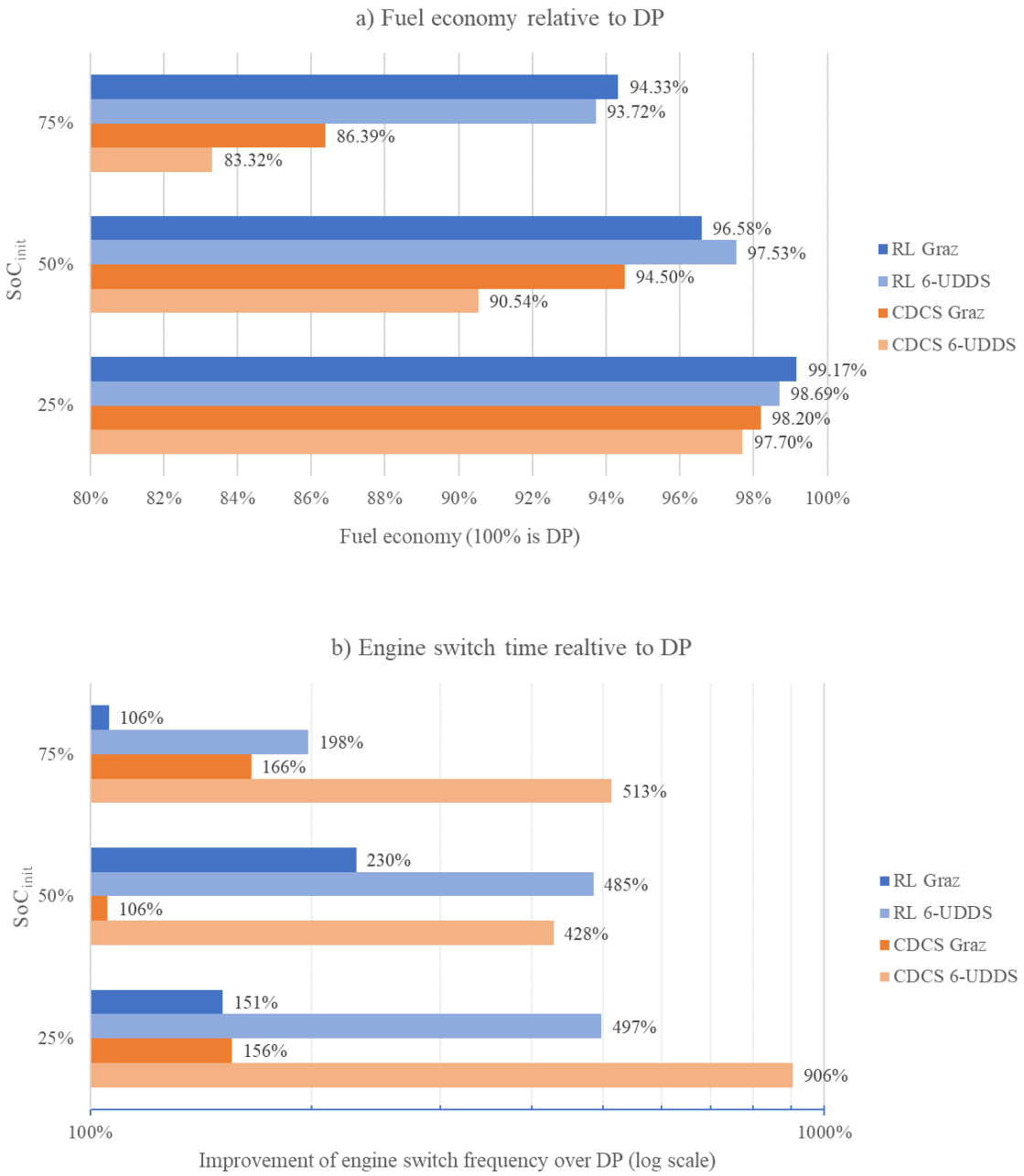


Figure 54 - Results for a) fuel economy and b) engine average switch time relative to the DP

6 CONCLUDING REMARKS

The following chapter summarizes the outcomes of the thesis and recommends some future improvements based on an objective discussion.

6.1 SUMMARY AND CONCLUSION

PHEVs offer a promising solution for the increasing CO_2 emission problem as one of the main categories in the automotive industry electrification process. PHEVs can improve the fuel economy due to the additional large size battery and EM. However, the improved economy strongly depends on the HCU control strategy. Navigation, communication devices, and sensors inspired a growing development of the advanced energy management control strategies especially that the CDCS strategy is neither simple nor advantageous anymore with the increasing control objectives.

Several scholars proved that advanced strategies inspired by AI such as reinforcement learning-based strategies can significantly improve the fuel economy of HEVs. The basement of the thesis is a P2-PHEV while the ultimate objective of the thesis is to improve fuel economy by introducing an adaptive online learning RL agent into the existing HCU architecture.

In this study, an offline GOP tool is developed firstly to obtain a globally optimal solution leading to smart power distributions that contribute mostly to the fuel economy improvement. The offline GOP results are used later to benchmark the subsequent developed algorithms performance to. Thereafter, RL-based algorithms are introduced to provide an online learning control strategy to solve the EMS for near-optimal solutions. The development process began with describing the control problem as an infinite-horizon optimal-control problem. The mathematical formulation is a function of the battery SoC, driver torque demand, vehicle speed, remaining trip distance, and the engine on/off status.

The objective is to minimize the vehicle's expected total fuel consumption besides penalizing both the battery SoC deviation from a linearly space-domain indexed SoC reference, and the frequent engine switch. Accordingly, the design process reasonably considered

the drivability and comfort requirements instead of sacrificing them too much for the sake of fuel economy.

Several RL-based algorithms proposed recently by the AI scientific community have been experimented with using a vehicle model simulation. As a result, an E-DQN agent is proposed by the thesis, trained on the HWFET cycle, and deployed on other two cycles to evaluate the performance. The E-DQN's control strategy outperformed the CDCS strategy in terms of fuel economy, up to 10.46% improvement, alongside providing adequate compliance with the optimization problem objectives. The thesis findings strongly accord to the necessity of enabling AI-based control strategies into the next generation of automobiles particularly in the era of autonomous driving and connected vehicles. The control objectives are becoming more and more complex for traditional methods to handle, thus AI-enabled technologies shall be prioritized by OEMs for further research and development.

6.2 FUTURE PROSPECTS AND RECOMMENDATIONS

The accelerating development of computational resources in recent decades enabled novel complex and intelligent algorithms to be involved in modern control systems. Advancement in the RL field incorporating DL and NNs forms the next trend for the RL-based HEVs/PHEVs energy management strategies. Integration with ITSs to construct a smart city or smart grid is coming soon with more comprehensive and complicated optimization control objectives. In the future connected environment, distributed and multi-agent DRL systems are a necessity for cooperative learning between vehicles on the road.

The sooner the transition towards intelligent control systems by automotive OEMs, the better they are prepared and qualified for the upcoming challenges. The thesis presented an initial step overlooking the way towards realizing an intelligent adaptive energy management system for PHEVs. The upcoming research efforts shall investigate more the following spots:

- **DRL algorithms:** the DQN algorithm is solely considered by the thesis in DRL algorithms. However, other types such as the Policy Gradient family (PG) are promising to be examined [92] including Proximal Policy Optimization (PPO) [93], and Deep Deterministic Policy Gradient (DDPG) [52].
- **Multiple objectives:** for improved performance, including other objectives to be optimized such as greenhouse gas emissions [94], the battery SoH [95], safety, comfort, user convenience [96], and powertrain mobility [97], brings additional benefits and are advantageous.
- **RL agents testing and validation:** besides the theoretical feasibility that is validated by simulation, practical implementation is necessary to be achieved through real vehicle evaluations. Few studies proceeded with their proposed methodologies to the HiL and vehicle-in-the-loop (ViL) testing. Therefore, more research efforts shall further examine the validity of such approaches in real environments.

7 LIST OF ABBREVIATIONS

Abbreviation	Explanation
AdaGrad	Adaptive Gradients
ADAM	Adaptive Moment Estimation
AddBoost	Additive Boost
ADVISOR	ADvanced VehIcle SimulatOR
A-ECMS	Adaptive Equivalent Consumption Minimization Strategy
AER	All-Electric Range
AI	Artificial Intelligence
ANN	Artificial Neural Network
BGD	Batch Gradient Descent
BMS	Battery Management System
BSFC	Brake-Specific Fuel Consumption
BSG	Belt Driven Starter/Generator
C/GMRES	Continuation/Generalized Minimum RESidual
CAVs	Connected and Automated Vehicles
CD	Charge-Depletion
CO	Carbon Oxide
CO_2	Carbon Dioxide
ConvDrv	Conventional Drive
CS	Change-Sustaining
DCT	Dual Clutch Transmission
DDP	Deterministic Dynamic Programming
DDQN	Double Deep Q-Network
DL	Deep Learning
DNN	Deep Neural Network
DP	Dynamic Programming

DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ECU	Engine Control Unit
E-DQN	Extended- Deep Q-Network
EltlDrv	Electric Drive
eMPC	Explicit MPC
ESS	Energy Storage System
EV	Electric Vehicle
FASTSim	Future Automotive Systems Technology Simulator
FIFO	First In First Out
FTP	Federal Test Procedure
GOP	Global OPtimization techniques
HCTI	History Cumulative Trip Information
HCUs	Hybrid Control Units
HEV	Hybrid Electric Vehicle
HF	Hybridization Factor
HMI	Human-Machine Interface
HV	High Voltage
HWFET	Highway Fuel Economy Driving Schedule
ICE	Internal Combustion Engine
IoT	Internet of Things
IRL	Inverse Reinforcement Learning
ISG	Integrated Starter/Generator
ITSs	Intelligent Transportation Systems
ldleGentn	Idle Generation
MC	Monto Carlo
MinGentn	Minimum Generation
MLP	Multi-Layer Perceptron

MPC	Model Predictive Control
NEDC	New European Driving Cycle
NN	Neural Network
NOx	Nitrogen Oxides
NVH	Noise Vibration and Harshness
OptmGentn	Optimum Generation
PER	Prioritized Experience Replay
PHEV	Plug-in Hybrid Electric Vehicle
PMP	Pontryagin Minimum Principle
POMDP	Partially Observable Markov Decision Process
PSO-based MPC	Particle Swarm Optimization-based MPC strategy
QP	Quadratic Programming
Recup	Recuperation
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RMSprop	Root Mean Squared Propagation
RNN	Recurrent Neural Networks
SDP	Stochastic Dynamic Programming
SGD	Stochastic Gradient Descent
SoC	State of Charge
SoH	State of Health
St/Sndtl	Stop/Standstill
SubBoost	Substitute Boost
TCU	Transmission Control Unit
TD	Temporal Difference
TPM	Transition Probability Matrices
Tq_{EM}	EM Torque

Tq_{ICE}	ICE Torque
UDDS	Urban Dynamometer Driving Schedule
V2C	Vehicle to Cloud
V2G	Vehicle to Grid
V2V	Vehicle to Vehicle
V2X	Vehicle to X
ViL	Vehicle-in-the-Loop
WLTP	Worldwide Harmonized Light Vehicles Test Procedure

8 LIST OF FIGURES

9 LIST OF TABLES

10 REFERENCES/ BIBLIOGRAPHY

[1] International Organization of Motor Vehicle Manufacturers, “2019 Statistics | www.oica.net.”https://www.oica.net/category/production-statistics/2019-statistics (accessed Jun. 06, 2021).

[2] Sarah Petit, “World Vehicle Population Rose 4.6% in 2016 :: Wards Intelligence.” https://wardsintelligence.informa.com/WI058630/World-Vehicle-Population-Rose-46-in-2016 (accessed Jun. 06, 2021).

[3] ICCT, “Chart library: Passenger vehicle fuel economy | International Council on Clean Transportation.” https://theicct.org/chart-library-passenger-vehicle-fuel-economy (accessed Jul. 03, 2021).

[4] U. Heyder, S. Schaphoff, D. Gerten, and W. Lucht, “Risk of severe climate change impact on the terrestrial biosphere,” *Environ. Res. Lett.*, vol. 6, no. 3, p. 034036, Sep. 2011, doi: 10.1088/1748-9326/6/3/034036.

[5] International Council on Clean Transportation, “2020 – 2030 CO 2 standards for new cars and light-commercial vehicles in the European Union,” *Icct*, no. November, 2016.

[6] J. Scoltock, “Nissan Leaf Data Sheet - GoElectric,” *Automot. Eng.*, vol. 40, no. 10, pp. 16–17, 2015.

[7] Michael Hughes, “Everything You Need to Know About Charging the Nissan LEAF PLUS | ChargePoint.” https://www.chargepoint.com/blog/everything-you-need-know-about-charging-nissan-leaf-plus/ (accessed Jul. 03, 2021).

[8] Porsche Fahrtraum, “Lohner-Porsche Mixte 1901 | Ferdinand Porsche Erlebniswelten fahr(T)raum Mattsee.” https://fahrtraum.at/en/portfolio-item/lohner-porsche-mixte-1901/ (accessed Jul. 03, 2021).

[9] Alex Crippen, “Warren Buffett’s Electric Car Hits the Chinese Market, But Rollout Delayed For U.S. & Europe.” https://www.cnbc.com/id/28236421 (accessed Jul.

18, 2021).

[10] G. B. Sandra Wappelhorst, “The uptake of plug-in hybrid electric vehicles in Europe’s company car fleets: Trends and policies | International Council on Clean Transportation.”https://theicct.org/blog/staff/phev-europe-company-cars-apr2021 (accessed May 05, 2021).

[11] C. E. M. and E. V. I. (EVI) International Energy Agency (IEA), *Global EV Outlook 2020: Enterign the decade of electric drive?* IEA Publications, 2020. Accessed: Jul. 03, 2021. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2020

[12] N. Kim, S. Cha, and H. Peng, “Optimal control of hybrid electric vehicles based on Pontryagin’s minimum principle,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 5, pp. 1279–1287, Sep. 2011, doi: 10.1109/TCST.2010.2061232.

[13] M. P. O’Keefe and T. Markel, “Dynamic programming applied to investigate energy management strategies for a plug-In HEV1,” *22nd Int. Batter. Hybrid Fuel Cell Electr. Veh. Symp. Expo. EVS 2006*, no. November, pp. 1035–1046, 2006.

[14] W. Liu, *Introduction to Hybrid Vehicle System Modeling and Control*. John Wiley and Sons, 2013. doi: 10.1002/9781118407400.

[15] J. Liu and H. Peng, “Modeling and control of a power-split hybrid vehicle,” *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 6, pp. 1242–1251, 2008, doi: 10.1109/TCST.2008.919447.

[16] Y. Yang, X. Hu, H. Pei, and Z. Peng, “Comparison of power-split and parallel hybrid powertrain architectures with a single electric machine: Dynamic programming approach,” *Appl. Energy*, vol. 168, pp. 683–690, Apr. 2016, doi: 10.1016/j.apenergy.2016.02.023.

[17] Q. Xue, X. Zhang, T. Teng, J. Zhang, Z. Feng, and Q. Lv, “Energy Management Strategy , and Control,” 2020.

[18] R. Bao, V. Avila, and J. Baxter, “Effect of 48 v Mild Hybrid System Layout on

Powertrain System Efficiency and Its Potential of Fuel Economy Improvement," in *SAE Technical Papers*, Mar. 2017, vol. 2017-March, no. March. doi: 10.4271/2017-01-1175.

[19] A. Sciarretta and L. Guzzella, "Optimal Energy-Management Strategies," *IEEE Control Syst.*, vol. 27, no. 2, pp. 60–70, Apr 2007, doi: 10.1109/MCS.2007.338280.

[20] C. Hou, L. Xu, H. Wang, M. Ouyang, and H. Peng, "Energy management of plug-in hybrid electric vehicles with unknown trip length," *J. Franklin Inst.*, vol. 352, no. 2, pp. 500–518, Feb. 2015, doi: 10.1016/j.jfranklin.2014.07.009.

[21] H. Chen, "Predictive Control Strategies of Plug-in HEVs," 2019.

[22] I. Evtimov, R. Ivanov, and M. Sapundjiev, "Energy consumption of auxiliary systems of electric cars," in *MATEC Web of Conferences*, Nov. 2017, vol. 133. doi: 10.1051/matecconf/201713306002.

[23] S. M. Mousavi G. and M. Nikdel, "Various battery models for various simulation studies and applications," *Renew. Sustain. Energy Rev.*, vol. 32, pp. 477–485, 2014, doi: 10.1016/j.rser.2014.01.048.

[24] T. Liu, *Reinforcement Learning-Enabled Intelligent Energy Management for Hybrid Electric Vehicles*, vol. 3, no. 5 2019. doi: 10.2200/s00934ed1v01y2019 07aat009.

[25] R. Guglielmann and L. Ironi, "Generating fuzzy models from deep knowledge: Robustness and interpretability issues," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2005, vol. 3571 LNAI, pp. 600–612. doi: 10.1007/11518655_51.

[26] S. Gang, J. Yuanwei, X. Aidong, and M. Jia, "Study and simulation of based-fuzzy-logic parallel hybrid electric vehicles control strategy," in *Proceedings - ISDA 2006: Sixth International Conference on Intelligent Systems Design and Applications*, 2006, vol. 1, pp. 280–284. doi: 10.1109/ISDA.2006.252.

[27] W. Jun, Q. Wang, P. Wang, and B. Han, "The optimization of control parameters

for hybrid electric vehicles based on genetic algorithm," *SAE Tech. Pap.*, vol. 1, 2014, doi: 10.4271/2014-01-1894.

[28] X. Hu, T. Liu, X. Qi, and M. Barth, "Reinforcement Learning for Hybrid and Plug-In Hybrid Electric Vehicle Energy Management: Recent Advances and Prospects," *IEEE Ind. Electron. Mag.*, vol. 13, no. 3, pp. 16–25, 2019, doi: 10.1109/MIE.2019.2913015.

[29] D. K. Smith and D. P. Bertsekas, *Dynamic Programming and Optimal Control. Volume 1*, vol. 47, no. 6. 1996. doi: 10.2307/3010291.

[30] O. Sundström and L. Guzzella, "A generic dynamic programming Matlab function," in *Proceedings of the IEEE International Conference on Control Applications*, 2009, pp. 1625–1630. doi: 10.1109/CCA.2009.5281131.

[31] R. Wang and S. M. Lukic, "Dynamic programming technique in hybrid electric vehicle optimization," 2012. doi: 10.1109/IEVC.2012.6183284.

[32] Chan-Chiao Lin, Huei Peng, and J. W. Grizzle, "A stochastic control strategy for hybrid electric vehicles," pp. 4710–4715 vol.5, May 2018, doi: 10.23919/ACC.2004.1384056.

[33] C. Vagg, S. Akehurst, C. J. Brace, and L. Ash, "Stochastic Dynamic Programming in the Real-World Control of Hybrid Electric Vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 853–866, May 2016, doi: 10.1109/TCST.2015.2498141.

[34] G. Paganelli, S. Delprat, T. M. Guerra, J. Rimaux, and J. J. Santin, "Equivalent consumption minimization strategy for parallel hybrid powertrains," *IEEE Veh. Technol. Conf.*, vol. 4, pp. 2076–2081, 2002, doi: 10.1109/VTC.2002.1002989.

[35] A. Rezaei, J. B. Burl, and B. Zhou, "Estimation of the ECMS Equivalent Factor Bounds for Hybrid Electric Vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 2198–2205, Nov. 2018, doi: 10.1109/TCST.2017.2740836.

[36] C. Musardo, G. Rizzoni, and B. Staccia, "A-ECMS: An adaptive algorithm for

hybrid electric vehicle energy management," *Proc. 44th IEEE Conf. Decis. Control. Eur. Control Conf. CDC-ECC '05*, vol. 2005, pp. 1816–1823, 2005, doi: 10.1109/CDC.2005.1582424.

[37] K. Patan, "Model predictive control," in *Studies in Systems, Decision and Control*, vol. 197, 2019, pp. 77–129. doi: 10.1007/978-3-030-11869-3_4.

[38] H. A. Borhan, C. Zhang, A. Vahidi, A. M. Phillips, M. L. Kuang, and S. Di Cairano, "Nonlinear model predictive control for power-split hybrid electric vehicles," *Proc. IEEE Conf. Decis. Control*, pp. 4890–4895, 2010, doi: 10.1109/CDC.2010.5718075.

[39] N. Guo, Z. Chen, Y. Wu, J. Shen, and R. Xiao, "A novel velocity forecast method for improving predictive energy management of plug-in hybrid electric vehicles," *2017 IEEE Veh. Power Propuls. Conf. VPPC 2017 - Proc.*, vol. 2018-January, pp. 1–6, Apr. 2018, doi: 10.1109/VPPC.2017.8330915.

[40] A. Taghavipour, N. L. Azad, and J. McPhee, "Real-time predictive control strategy for a plug-in hybrid electric powertrain," *Mechatronics*, vol. 29, pp. 13–27, Aug. 2015, doi: 10.1016/J.MECHATRONICS.2015.04.020.

[41] Y. Wang, X. Wang, Y. Sun, and S. You, "Model predictive control strategy for energy optimization of series-parallel hybrid electric vehicle," *J. Clean. Prod.*, vol. 199, pp. 348–358, Oct. 2018, doi: 10.1016/J.JCLEPRO.2018.07.191.

[42] X. Qi, G. Wu, K. Boriboonsomsin, M. J. Barth, and J. Gonder, "Data-driven reinforcement learning-based real-time energy management system for plug-in hybrid electric vehicles," *Transp. Res. Rec.*, vol. 2572, pp. 1–8, 2016, doi: 10.3141/2572-01.

[43] S. Yue, Y. Wang, Q. Xie, D. Zhu, M. Pedram, and N. Chang, "Model-free learning-based online management of hybrid electrical energy storage systems in electric vehicles," *IECON Proc. (Industrial Electron. Conf.*, pp. 3142–3148, Feb. 2014, doi: 10.1109/IECON.2014.7048959.

[44] X. Lin, Y. Wang, P. Bogdan, N. Chang, and M. Pedram, “Reinforcement learning based power management for hybrid electric vehicles,” *IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD*, vol. 2015-January, no. January, pp. 32–38, Jan. 2015, doi: 10.1109/ICCAD.2014.7001326.

[45] C. Liu and Y. L. Murphey, “Power management for plug-in hybrid electric vehicles using reinforcement learning with trip information,” *2014 IEEE Transp. Electrif. Conf. Expo Components, Syst. Power Electron. - From Technol. to Bus. Public Policy, ITEC 2014*, Jul. 2014, doi: 10.1109/ITEC.2014.6861862.

[46] X. Lin, P. Bogdan, N. Chang, and M. Pedram, “Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost,” *2015 IEEE/ACM Int. Conf. Comput. Des. ICCAD 2015*, pp. 627–634, Jan. 2016, doi: 10.1109/ICCAD.2015.7372628.

[47] A. Vogel, D. Ramachandran, R. Gupta, and A. Raux, “Improving hybrid vehicle fuel efficiency using inverse reinforcement learning,” *Proc. Natl. Conf. Artif. Intell.*, vol. 1, pp. 384–390, 2012.

[48] T. Liu, Y. Zou, D. Liu, and F. Sun, “Reinforcement learning-based energy management strategy for a hybrid electric tracked vehicle,” *Energies*, vol. 8, no. 7, pp. 7243–7260, 2015, doi: 10.3390/EN8077243.

[49] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. J. Barth, “Deep reinforcement learning-based vehicle energy efficiency autonomous learning system,” *IEEE Intell. Veh. Symp. Proc.*, pp. 1228–1233, Jul. 2017, doi: 10.1109/IVS.2017.7995880.

[50] Y. Hu, W. Li, K. Xu, T. Zahid, F. Qin, and C. Li, “Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning,” *Appl. Sci.*, vol. 8, no. 2, Jan. 2018, doi: 10.3390/APP8020187.

[51] P. Zhao, Y. Wang, N. Chang, Q. Zhu, and X. Lin, “A deep reinforcement learning framework for optimizing fuel economy of hybrid electric vehicles,” *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC*, vol. 2018-January, pp. 196–202, Feb.

2018, doi: 10.1109/ASPDAC.2018.8297305.

[52] R. Liessner, C. Schroer, A. Dietermann, and B. Bäker, “Deep reinforcement learning for advanced energy management of hybrid electric vehicles,” *ICAART 2018 - Proc. 10th Int. Conf. Agents Artif. Intell.*, vol. 2, pp. 61–72, 2018, doi: 10.5220/0006573000610072.

[53] J. Wu, H. He, J. Peng, Y. Li, and Z. Li, “Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus,” *Appl. Energy*, vol. 222, pp. 799–811, Jul. 2018, doi: 10.1016/J.APENERGY.2018.03.104.

[54] P. Wang and W. Northrop, “Reinforcement Learning based Energy Management of Multi-Mode Plug-in Hybrid Electric Vehicles for Commuter Route,” *SAE Tech. Pap.*, vol. 2020-April, no. April, Apr. 2020, doi: 10.4271/2020-01-1189.

[55] D. T. Hoang, P. Wang, D. Niyato, and E. Hossain, “Charging and discharging of plug-in electric vehicles (PEVs) in vehicle-to-grid (V2G) systems: A cyber insurance-based model,” *IEEE Access*, vol. 5, pp. 732–754, 2017, doi: 10.1109/ACCESS.2017.2649042.

[56] Z. Zhu, S. Gupta, A. Gupta, and M. Canova, “A Deep Reinforcement Learning Framework for Eco-driving in Connected and Automated Hybrid Electric Vehicles,” pp. 1–14, 2021, [Online]. Available: http://arxiv.org/abs/2101.05372

[57] R. E. Bellman and E. S. Lee, “History and Development of Dynamic Programming,” *IEEE Control Syst. Mag.*, vol. 4, no. 4, pp. 24–28, 1984, doi: 10.1109/MCS.1984.1104824.

[58] D. Ambuhl and L. Guzzella, “Predictive reference signal generator for hybrid electric vehicles,” *IEEE Trans. Veh. Technol.*, vol. 58, no. 9, pp. 4730–4740, Nov. 2009, doi: 10.1109/TVT.2009.2027709.

[59] C. Watkins, “Learning from delayed rewards,” 1989, Accessed: Jul. 19, 2021. [Online]. Available: https://www.academia.edu/download/50360235/Learning_

from_delayed_rewards_20161116-28282-v2pwvq.pdf

[60] P. W.-P. of the 28th I. C. on and undefined 1989, “Neural networks for control and system identification,” *ieeexplore.ieee.org*, Accessed: Jul. 19, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/70114/

[61] R. S.-M. learning 1988, “Learning to predict by the methods of temporal differences,” *Springer*, Accessed: Jul. 19, 2021. [Online]. Available: https://link.springer.com/content/pdf/10.1007/BF00115009.pdf

[62] R. S. Sutton and A. G. Barto, “Reinforcement Learning, Second Edition: An Introduction - Complete Draft,” *MIT Press*, pp. 1–3, 2018.

[63] Juha Kiili, “Reinforcement Learning Tutorial Part 3: Basic Deep Q-Learning.” https://valohai.com/blog/reinforcement-learning-tutorial-basic-deep-q-learning/ (accessed Jul. 09, 2021).

[64] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nat. 2015 5187540*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[65] A. C. Ian Goodfellow, Yoshua Bengio, *Deep Learning (Adaptive Computation and Machine Learning series)*, vol. 19, no. 1–2. Springer US, 2017. Accessed: Jul. 21, 2021. [Online]. Available: https://books.google.com/books/about/Deep_Learning.html?id=Np9SDQAAQBAJ

[66] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, Sep. 1951, doi: 10.1214/AOMS/1177729586.

[67] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, “Randomized smoothing for (parallel) stochastic optimization,” *Proc. IEEE Conf. Decis. Control*, vol. 12, pp. 5442–5444, 2012, doi: 10.1109/CDC.2012.6426698.

[68] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA Neural networks Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[69] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[70] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2094–2100, Sep. 2015, Accessed: Jul. 21, 2021. [Online]. Available: https://arxiv.org/abs/1509.06461v3

[71] Mathworks, “Create Policy and Value Function Representations - MATLAB & Simulink.” https://www.mathworks.com/help/reinforcement-learning/ug/create-policy-and-value-function-representations.html (accessed Sep. 19, 2021).

[72] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, Feb. 2015, Accessed: Sep. 09, 2021. [Online]. Available: https://arxiv.org/abs/1502.03167v3

[73] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural Architecture Optimization,” *Adv. Neural Inf. Process. Syst.*, vol. 2018-December, pp. 7816–7827, Aug. 2018, Accessed: Sep. 09, 2021. [Online]. Available: https://arxiv.org/abs/1808.07233v5

[74] Z. Zhu, Y. Liu, and M. Canova, “Energy Management of Hybrid Electric Vehicles via Deep Q-Networks,” *Proc. Am. Control Conf.*, vol. 2020-July, pp. 3077–3082, 2020, doi: 10.23919/ACC45564.2020.9147479.

[75] C. Z. Chengzhao Yang, “An Energy Management Strategy of Hybrid Electric Vehicl...” https://www.ijeart.com/an-energy-management-strategy-of-hybrid-electric-vehicles-based-on-deep-reinforcement-learning (accessed Sep. 09, 2021).

[76] R. Lian, H. Tan, J. Peng, Q. Li, and Y. Wu, “Cross-Type Transfer for Deep Reinforcement Learning Based Hybrid Electric Vehicle Energy Management,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8367–8380, 2020, doi: 10.1109/TVT.2020.2999263.

[77] W. Fedus *et al.*, “Revisiting Fundamentals of Experience Replay,” *37th Int. Conf. Mach. Learn. ICML 2020*, vol. PartF168147-4, pp. 3042–3052, Jul. 2020, Accessed: Sep. 09, 2021. [Online]. Available: https://arxiv.org/abs/2007.06700v1

[78] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized Experience Replay,” *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, Nov. 2015, Accessed: Sep. 17, 2021. [Online]. Available: https://arxiv.org/abs/1511.05952v4

[79] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, Sep. 2015, Accessed: Sep. 18, 2021. [Online]. Available: https://arxiv.org/abs/1509.02971v6

[80] T. M. Moerland, J. Broekens, and C. M. Jonker, “A Framework for Reinforcement Learning and Planning,” Jun. 2020, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/2006.15009v3

[81] T. Wang *et al.*, “Benchmarking Model-Based Reinforcement Learning,” Jul. 2019, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/1907.02057v1

[82] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous Deep Q-Learning with Model-based Acceleration,” *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 6, pp. 4135–4148, Mar. 2016, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/1603.00748v1

[83] H. Lee, C. Kang, Y. Il Park, N. Kim, and S. W. Cha, “Online data-driven energy management of a hybrid electric vehicle using model-based Q-learning,” *IEEE Access*, vol. 8, pp. 84444–84454, 2020, doi: 10.1109/ACCESS.2020.2992062.

[84] O. Vinyals *et al.*, “StarCraft II: A New Challenge for Reinforcement Learning,” Aug. 2017, Accessed: Sep. 18, 2021. [Online]. Available: https://arxiv.org/abs/1708.04782v1

[85] S. G. Wirasingha and A. Emadi, “Classification and review of control strategies for plug-in hybrid electric vehicles,” *IEEE Trans. Veh. Technol.*, vol. 60, no. 1, pp. 111–122, Jan. 2011, doi: 10.1109/TVT.2010.2090178.

[86] C. M. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao, and M. Wellers, "Energy Management in Plug-in Hybrid Electric Vehicles: Recent Progress and a Connected Vehicles Perspective," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 4534–4549, Jun. 2017, doi: 10.1109/TVT.2016.2582721.

[87] C. Sun, S. J. Moura, X. Hu, J. K. Hedrick, and F. Sun, "Dynamic Traffic Feedback Data Enabled Energy Management in Plug-in Hybrid Electric Vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 3, pp. 1075–1086, May 2015, doi: 10.1109/TCST.2014.2361294.

[88] J. Peng, H. He, and R. Xiong, "Rule based energy management strategy for a series–parallel plug-in hybrid electric bus optimized by dynamic programming," *Appl. Energy*, vol. 185, pp. 1633–1643, Jan. 2017, doi: 10.1016/J.APENERGY.2015.12.031.

[89] Y. Li, H. He, J. Peng, and H. Wang, "Deep reinforcement learning-based energy management for a series hybrid electric vehicle enabled by history cumulative trip information," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7416–7430, Aug. 2019, doi: 10.1109/TVT.2019.2926472.

[90] EPA, "Dynamometer Drive Schedules" https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules (accessed Oct. 16, 2021).

[91] C. M. Bishop, "Neural networks for pattern recognition," p. 482, 1995.

[92] B. V. Mbuwir, L. Vanmunster, K. Thoelen, and G. Deconinck, "A hybrid policy gradient and rule-based control framework for electric vehicle charging," *Energy AI*, vol. 4, p. 100059, Jun. 2021, doi: 10.1016/J.EGYAI.2021.100059.

[93] T. Liu, B. Wang, W. Tan, S. Lu, and Y. Yang, "Data-Driven Transferred Energy Management Strategy for Hybrid Electric Vehicles via Deep Reinforcement Learning," Sep. 2020, Accessed: Oct. 19, 2021. [Online]. Available: https://arxiv.org/abs/2009.03289v2

[94] P. You *et al.*, "Scheduling of EV Battery Swapping - Part I: Centralized Solution,"

IEEE Trans. Control Netw. Syst., vol. 5, no. 4, pp. 1887–1897, Dec. 2018, doi: 10.1109/TCNS.2017.2773025.

[95] X. Wang, C. Yuen, N. U. Hassan, N. An, and W. Wu, "Electric Vehicle Charging Station Placement for Urban Public Bus Systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 128–139, Jan. 2017, doi: 10.1109/TITS.2016.2563166.

[96] H. M. Chung, W. T. Li, C. Yuen, C. K. Wen, and N. Crespi, "Electric Vehicle Charge Scheduling Mechanism to Maximize Cost Efficiency and User Convenience," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3020–3030, May 2019, doi: 10.1109/TSG.2018.2817067.

[97] R. Yu, W. Zhong, S. Xie, C. Yuen, S. Gjessing, and Y. Zhang, "Balancing Power Demand Through EV Mobility in Vehicle-to-Grid Mobile Energy Networks," *IEEE Trans. Ind. Informatics*, vol. 12, no. 1, pp. 79–90, Feb. 2016, doi: 10.1109/TII.2015.2494884.