Optimal Power Management Based on Q-Learning and Neuro-Dynamic Programming for Plug-in Hybrid Electric Vehicles

by

Chang Liu

**A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Information Systems Engineering)
in the University of Michigan-Dearborn
2018**

Doctoral Committee:

Professor Yi Lu Murphey, Chair
Assistant Professor Wencong Su
Staff Researcher Shige Wang, General Motors Co.
Associate Professor Ya Sha Yi

© Chang Liu 2018

All Rights Reserved

DEDICATION

It has been seven years since I started this PhD study. I just realized how much time and efforts I have put in during this longer-than-expected part-time study. I would like to use this dedication page to thank my family for their continuous support and encouragement during all these years - particularly, for all the warm encouragement from my wife Shunyao when I was not able to see the light at the end of the tunnel, for all the valuable advice from my mother Shichong regarding my research, and for my father Keren's great help, which allowed me to have more time working on my research. I also dedicate this dissertation to my lovely son Ethan for bringing so much fun to me throughout this entire long and challenging journey.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Yi Lu Murphey. She introduced me to this program about eight years ago. Her expertise, diligence, and guidance have made me a better thinker, writer, and researcher. I am grateful for her continuous and strong support during my PhD study, for her patience, time, and all the great advice. Without her help, this dissertation would not be possible.

Besides my advisor, I would like to thank the rest of my thesis committee, Prof. Ya Sha Yi, Prof. Wencong Su, and Dr. Shige Wang, for their valuable comments, suggestions, and time. My sincere thanks go to my friend and colleague Dr. Shige Wang, who provided me with such valuable advice based on his own PhD study experience.

I would like to thank all the university staff and faculty members who have provided administrative help during my study. I also thank General Motors for encouraging and supporting its employees to pursue further academic degrees.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Energy optimization for plug-in hybrid electric vehicles (PHEVs) is a challenging problem due to its system complexity and various constraints. In this research, we present a Q-learning based in-vehicle model-free solution that can robustly converge to the optimal control. The proposed algorithms combine neuro-dynamic programming (NDP) with future trip information to effectively estimate the expected future energy cost (expected cost-to-go) for a given vehicle state and control actions. The convergence of those learning algorithms is demonstrated on both fixed and randomly selected drive cycles. Based on the characteristics of these learning algorithms, we propose a two-stage deployment solution for PHEV power management applications. We will also introduce a new initialization strategy that combines optimal learning with a properly selected penalty function. Such initialization can reduce the learning convergence time by 70%, which has huge impact on in-vehicle implementation. Finally, we develop a neural network (NN) for the battery state-of-charge (SoC) prediction, rendering our power management controller completely model-free.

Keywords: Machine learning, Reinforcement learning, Q-learning, Neuro-dynamic programming, Plug-in hybrid electric vehicles, Power management, Energy optimization, Energy cost minimization

CHAPTER I

Introduction

Automotive electrification is advancing rapidly. Multiple vehicle manufacturers have announced significant investments in developing the next generation of hybrid and electrical vehicles. For example, Ford has announced a $4.5 billion investment for developing 13 new hybrids and electric cars by 2020. GM was able to increase the all-electric range (AER) of the Chevy Volt from 38 miles to 50 miles while cutting its price by more than a thousand dollars. Several Chinese OEMs are also adding new plug-in hybrid and electric models to their product portfolios. The main advantages of electrified vehicles are energy cost savings and emissions reduction. Although the fuel price has dropped recently, it still represents the dominant portion of the total energy cost for hybrid electric vehicles (HEVs). In addition, more stringent emission standards are on the horizon. For example, the United Nations Climate Change Conference has proposed zero net greenhouse gas emissions for the second half of the 21st century.

The full electric cars currently available on the market are either very compact in size or expensive due to the on-board battery packs. The supporting charging infrastructure has not been fully established. At this moment, large consumer vehicles such as SUVs, pickup trucks, etc. still rely on hybrid technology to reduce their fuel cost and emissions. Plug-in hybrid technology represents a good solution for the transition of vehicles into full electrification. PHEVs carry both an engine and a rechargeable battery pack, so they can have the same power and driving range as conventional vehicles, but offer fuel cost savings and fewer emissions. In recent years, *plug-in hybrid electric vehicles (PHEVs)* have become more and more popular. Compared to conventional hybrid electric vehicles (HEVs), a PHEV is normally equipped with a bigger battery pack and a more powerful electric motor. The on-board battery can be charged by using a wall outlet. So, a PHEV can offer an *all-electric range (AER)* (the driving distance with fully charged battery and engine off), which is not achievable by a conventional HEV. We foresee the technology trend of moving from conventional HEVs to PHEVs mainly due to the following reasons:

- PHEVs can offer more fuel savings and produce fewer emissions than conventional HEVs.
- Smart grid techniques are being developed for PHEVs to optimize electricity usage.
- The rapid development in battery technology will further reduce the cost of building PHEVs.
- The rapid advancement in charging technologies and infrastructures will greatly facilitate PHEV operations.

Table 1.1 and Table 1.2 list the PHEV models commercially available and the pre-production models that will be on the market in the near future [1].

For current hybrid control applications, energy cost minimization is still one of the primary control goals. This is supported by the obvious reason that most hybrid vehicle customers care about energy cost savings. As a matter of fact, the current PHEV design process involves the proper component sizing for battery, motors, engine, etc. at a very early design stage to meet the fundamental emission and performance requirements. The real-time on-road control still aims at saving energy as much as possible without compromising the driver's performance demand. For these reasons, we will focus on the problem of energy cost minimization for PHEVs. It is worth noting that our energy optimization solution can certainly be extended to incorporate the effect of emissions. The control methodologies presented in this paper would still be applicable.

1.1 Power Management for Hybrid Electric Vehicles

One of the most challenging problems in HEV or PHEV control is power management. The goal of power management is to optimize the energy usage of the vehicle while meeting the requirements for performance and emissions. For example, for a power-split hybrid architecture (details will be given later), at any instance of time, the proper power output and its split between the two power sources, the gasoline engine and the electric motor need to be determined. It is worth noting that the power flow from the electric motor could be negative. In such a case, the electric motor is functioning as a generator to charge the battery. At the time of braking, negative power output is desired, which could be partially contributed by the electric motor subject to the system's physical limitations. As a consequence, partial kinetic energy could be recovered into the battery.

The high complexity of the power management problem is mainly due to the following reasons:

- It is a global optimization problem subject to many system constraints. The optimization is for the overall energy usage over extended time. The focus is not on a local system or for a short period of time. The constraints include system evolution constraints, component physical constraints, etc., and they may be either static or dynamic in nature. Examples are engine/motor torque or power outputs, battery state-of-charge threshold, battery current and power output, transmission static and dynamic characteristics, etc.
- It aims at achieving multiple objectives at the same time and across multiple dimensions. The high-level objectives are minimizing the energy usage, reducing the

tail pipe emissions, and meeting the requirements for vehicle drivability and performance. Energy minimization can further include objectives such as reducing the battery energy loss, improving the motor and engine operating efficiencies, etc. For instance, the discharge of the battery leads to power loss due to the internal resistance of the battery. As a consequence, higher power output will generate higher loss. On the other hand, the engine and motor should be controlled to operate in the high efficiency region as much as possible.

- It is impossible to completely and accurately obtain the mathematical models for the overall system and various components such as engine, battery, motor, driveline, tires, etc. So there is no analytical approach for the optimization problem unless certain assumptions with simplified models are made.
- Driving information related to the energy usage is in general unknown before the trip. Random inputs such as driver's driving style, road and traffic conditions, weather conditions and other environmental conditions, etc. often have significant impact on the vehicle's energy usage.
- Component aging and wear cause system characteristics to change continuously.
- There are a variety of hybrid architectures available. In general, different architectures require different control strategies.
- The control algorithms need to run efficiently in real-time with limited computational and memory resources.

1.2 Power Management Solutions in the Literature

Various power management strategies have been researched in the literature. In his survey, Salmasi [2] classified power management methods into two general trends: rule-based (deterministic and fuzzy) and optimization-based (global and real time). Each approach is explained. The characteristics of various strategies are evaluated and compared qualitatively. Gurkaynak et al. [3] summarized the state-of-the-art power management algorithms for HEVs. The controllers are classified according to their dependency on time. The paper suggests that time-dependent controllers (dynamic controllers) show better performance than their static counterparts, which are rule-based controllers performing the control strategy based on instantaneous inputs. Ganji and Kouzani [4] surveyed the control methods using the look-ahead approach. They conclude that "there is a significant potential to improve the energy usage of HEVs if real traffic conditions are used." For example, intelligent transportation systems can provide road and traffic information, GPS can provide information about the location and elevation of a vehicle, GIS can give information about the traveling route in advance, and radars can determine the distance from the vehicle ahead. All of the above-mentioned trip information can be used to optimize energy efficiency. Very recently, Malikopoulos [5] provided a survey on the supervisory power management control algorithms for HEVs that have been reported in the literature to date. "The exposition ranges from parallel, series, and power split HEVs and PHEVs and includes a classification of the algorithms in terms of their implementation and the chronological order of their appearance." For each hybrid configuration, rule-based, offline, and online control algorithms have been reviewed. Also, learning-based algorithms have been discussed. The author suggested that future development for HEV energy optimization could integrate multi-scale information obtained from new on-board sensors and from vehicle-to-vehicle and vehicle-to-infrastructure communications, etc. By investigating this new optimization infrastructure, "the power management controller would have to account for limited uncertainty about surrounding traffic and commuters and be able to optimize fuel economy, pollutant emissions, as well as battery lifetime and range."

In general, the power management control strategies such as rule-based controls, dynamic programming, offline learning and online learning algorithms, etc. are applicable to either conventional HEVs or PHEVs, with the consideration that battery charge depletion is allowed for PHEVs. Thus, the same amount of SoC drop implies a different energy cost for HEVs and PHEVs. In the first case, the lost SoC has to be recovered from either regenerative braking or the engine. In the latter case, the lost SoC can be recharged by using electricity from the grid.

It has been proved that a simple control strategy such as an aggressive charge-depletion mode followed by a charge-sustaining control for PHEVs is far from optimal [6–8]. The commonly used power management control strategies can be classified as follows:

- Rule-based Control: This category can be subdivided into deterministic approach and fuzzy logic approach [2]. The deterministic rules [9, 10], such as the thermostat strategy to turn on/off the engine based on max and min SoC threshold levels, are defined based on intuition and human control expertise. The fuzzy logic controller uses fuzzy inference rules [11–14], which can be generated, for example, based on the engine, motor, and battery efficiency maps [11] or based on the optimal solution obtained from dynamic programming (DP) [13], to provide the control actions.

  Rule-based control methods have been popular among engineers due to their simplicity for implementation and interpretation. They are suitable for real-time applications. However, it is difficult to prove their optimality.

- Analytical Approach: Zhang et al. [15] developed a mathematical power loss model for a blended mode PHEV and obtained the optimal solution analytically for any constant-speed drive cycle based on simple control rules. The results were extended to other drive cycles. The paper proved that the control strategy yields better fuel economy if the total travel distance is known before the trip. In [8], the authors proposed an analytical approach to find the optimal power management solution. In order to solve the problem analytically, the authors made several assumptions, such as the electrical loss model, the fuel consumption rate, the power demand statistic distribution, etc. The control strategy is based on a set of simplified rules that lead to determining a pair of parameters representing the power demand threshold for turning on the engine and the constant motor power when the engine is on.

The analytical methods provide the optimal solution in the context of their problem formulation, but they are normally built on simplified control rules. Assumptions for the statistics of the drive cycles, power loss models etc. are also required.

- Equivalent Fuel Consumption Minimization Strategy (ECMS): It consists of evaluating the instantaneous cost function as a sum of the fuel consumption and an equivalent fuel consumption related to the SoC variation. The equivalence between electrical energy and fuel energy is basically evaluated by considering average energy paths leading from the fuel to the storage of electrical energy [16]. The real-time control strategy is evaluated under the assumption that every variation in the SoC will be compensated in the future by the engine running at the current operating point. Sciarretta et al. [17] proposed a new method for evaluating the equivalence factor between fuel and electrical energy. This method does not require the assumption of the average efficiencies of the parallel paths, and it is based on a coherent definition of system self-sustainability. Finally, Musardo et al. [18] presented an adaptive equivalent fuel consumption minimization (A-ECMS) algorithm, where the equivalence factor was estimated on-the-fly with respect to real-time driving conditions.

  ECMS-based algorithms aim at minimizing the instantaneous energy consumption. They are robust and easy to implement. Converting the electrical energy into the equivalent fuel cost may not be fully applicable for PHEV applications because the electrical energy can be recovered from the electric grid instead of being recovered from the fuel. Also, it is difficult to prove the algorithm's global optimality.

- Deterministic Dynamic Programming (DDP): DP is by now a widely used approach for optimal energy management in hybrid vehicles because it can solve general dynamic optimization problems by robustly handling the constraints and nonlinearity of the problem and producing a global optimal solution. Lin et al. [19] used DP for the optimal power management of a hybrid electric truck by minimizing a cost function over a driving cycle. They used the DP results to improve a simple control algorithm. Kum et al. [20] used DP results as the benchmark of the global optimization for minimizing the fuel consumption and tail pipe emissions. A new extraction method was developed to extract engine on/off, gear-shift, and power-split strategies from the DP results. Finally, an adaptive supervisory powertrain controller was built based on those extracted strategies. Patil et al. [21] compared two power management strategies, both based on DP. One has no restrictions on the fuel usage and another one permits engine operation only when the SoC is below a certain threshold level. The paper concludes that there is no significant difference in energy consumption between the two methods except if the fuel price is unreasonably cheap.

DP is commonly used in research as a benchmark for the optimization-based control strategies because it yields the global optimal solution. Also, control strategies can be extracted from the DP results to create real-time control rules, fuzzy inference rules, neural network training samples, etc. However, the DP method has mainly two drawbacks: Firstly, it requires the driving information, such as the vehicle speed and power demand, beforehand; secondly, the computational complexity of the DP approach is generally too high to be considered for a real-time implementation.

- Online Learning: Learning-based power management controls founded on artificial intelligence (A.I.) have gained increasing interest recently. They can be generally classified as supervised offline learning using neural networks (NNs) or online reinforcement learning (RL). Lin et al. [22] developed the breakthrough strategies based on stochastic dynamic programming (SDP). In this approach, the power demand at the next step is predicted by assuming that it has a Markov property. The Bellman optimality equation is used to provide the control inputs. The algorithm is called the policy iteration algorithm, which continuously learns the expected future energy usage cost and converges to the global optimal solution. Johannesson et al. [23] compared the results of three algorithms based on SDP but with different levels of trip information access. The first level gives a general description of the type of drive cycle, such as city driving; the second level uses GPS data with the traffic flow information; and the third highest access level assumes that the speed and power demand are exactly known, so deterministic DP is used for obtaining the global optimal results. In [24], the authors proposed a *shortest path stochastic dynamic programming (SP-SDP)* approach, where a terminal state is defined for drive cycles. This new approach allows no discount factor in the cost function and the deviations of battery SoC from a desired setpoint to be penalized only at key off. The simulation results show that SP-SDP provides a better SoC control and has fewer parameters to tune than an SDP approach where the problem is formulated in an infinite horizon. Liu and Peng [25] studied the power management algorithms for a power-split configuration by comparing the results of SDP, ECMS against deterministic DP. The paper concludes that besides a little bit more "jumpy" engine power produced by ECMS, both algorithms produced near optimal results. Moura et al. [26] used SDP over a distribution of the drive cycles. It also demonstrated that the fuel-electricity price ratio may play an important role for minimizing the total cost in an SDP algorithm.

SDP uses the Markov property to predict the next state transition probabilities. The policy iteration technique allows the controller to learn and converge to the optimal solution. This method can be implemented in real time without a priori trip knowledge. Statistical trip data or other driving information, such as the GPS coordination and traffic flow, may help the algorithm to better predict the state transitions. At present, the algorithms are generally model-dependent and their convergence property has not been studied.

SDP is essentially an RL-based approach. The system is modeled as a sequence of state transitions possessing the Markov property. At each state, the future *cost-to-go*, sometimes called the *future total reward* or *state-value* is estimated. This approach suffers a well-known problem named the “Curse of Dimensionality”[27–30]. The cost-to-go is a function of the state. If the cost-to-go value is stored in a table for each state, then, depending on the number of possible states, the table could be too big to be iteratively evaluated and updated. In other words, the number of iterations for each table value to converge becomes enormous, rendering this approach impossible in real-world applications.

Neuro-dynamic programming (Neuro-DP) or approximate dynamic programming (ADP) aims to estimate the cost-to-go function using neural networks or user-defined functions. In this way, the algorithm convergence is independent of the number of states. Instead, it relies on the approximation performance of the neural networks or the user-defined functions. Johannesson and Egardt [31] presented an ADP scheme that gives the possibility of efficiently estimating the value function, i.e., the cost-to-go function. The value function was approximated by using a linear function of engine torque for the fuel consumption and a quadratic term of SoC for the power loss of the battery. Johri and Filipi [32] proposed a neuro-DP structure that includes two networks, actor and critic. The critic network is trained to estimate the approximation of optimal cost-to-go function. The actor network is trained to produce control actions that are greedy with respect to optimal cost-to-go function. The objective is to optimize the desired performance by learning to choose appropriate control actions through interaction with the environment.

Q-learning is a powerful model-free reinforcement learning approach. Compared with the classical control methods, which normally incorporate tedious system identification, construction of detailed mathematical models, and big efforts in developing control synthesis, the Q-learning-based controller simply interacts with the environment and continuously improves the control policy from the rewards/costs observed. The learning process is capable of converging to the optimal control behavior. Q-learning is particularly suitable for applications where the system model is unknown or not accurate. For example, the vehicle on-board battery pack represents such a system [33, 34]. The characteristics of battery systems keep changing as the vehicle goes through numerous charging/discharging iterations. Q-learning can continuously learn from the observed experiences and automatically adjust to the evolving dynamics. Some researchers have also applied Q-learning on the fuel cell hybrid electric vehicles [35] to minimize the fuel cell consumption.

- Offline Learning: Supervised learning approaches have also been studied for HEV power management. Neural networks (NNs) are mostly used to achieve this control objective. Park et al. [36] developed an intelligent machine learning method for conventional vehicle power management. NNs have been trained to produce the optimal control parameters with respect to various road types and traffic congestion levels. The simulation results showed that the intelligent controller had performance very close to the optimal control generated by the offline DP. In 2012, Murphey et al. [37] extended their research on the NN algorithms for power-split HEVs. In the two papers published [37, 38], they demonstrated that their neural network-based control framework can effectively predict the road types and traffic congestion levels as well as the driving trend. Finally, the power-split NNs trained with respect to various road types and traffic congestion levels are capable of achieving quasi-optimal control results. Chen et al. [39] used two NNs trained with the optimized results obtained by the DP method to perform battery current control. Based on the knowledge of the trip length and duration, one of the NNs will be used to determine the power distribution between the ICE and the battery using the sub-optimal battery current control command.

Using NNs can be an effective way for energy optimization of the HEVs. However, NN-based offline learning demands that the pre-trained model be fit to all vehicles and driving cycles.

- Control Strategy Using Trip Information: Much research work has been conducted considering the trip information as future information which is, therefore, not available a priori. This situation can be changed by the recent rapid developments in on-board GPS, GIS, TMC, and intelligent transportation systems (ITS). The trip information generally include the predicted drive cycle for the near- or long-term future, the distance of travel, the time of travel, the road grade change, the road type and the traffic conditions, etc.

  Gong et al. [6] presented a DP-based global optimal power-management scheme for PHEVs by trip modeling with traffic data, where historical traffic data were used for trip modeling. Essentially, a simplified driving cycle model was constructed based on the trip route and the historical traffic information. Then, backward DP with interpolation was used to find the optimal power usages. Later on, Gong et al. [7] proposed a computationally efficient optimal power management for PHEVs based on spatial-domain two-scale DP. The trip is divided into a number of segments of certain length. The SoC and fuel consumption of each segment can be pre-calculated under different speeds and power-splitting ratios. The original optimization problem in the time domain is then converted into that in the spatial domain, which can greatly improve the computational efficiency. Zhang et al. [40] in their paper evaluated the potential gain in fuel economy if road grade information is integrated into the energy management of hybrid vehicles. Simulation results show that road terrain preview enables fuel savings. The level of improvement depends on the cruising speed, control strategy, road profile, and size of the battery. Recently, Yu et al. [41] used the trip preview information to generate an optimized energy usage policy based on predicted driving patterns along a trip. A feedback control system is designed to realize the preplanned optimal energy consumption process by controlling the fuel-to-electricity consumption ratio properly during the driving process.

Trip information can greatly help energy optimization. However, little work has been done combining the trip information with online learning for power management.

- Model Predictive Control (MPC): MPC tries to iteratively solve a finite horizon optimization problem based on a plant model. At one time step, the system state is captured, and an on-the-fly computation explores the state trajectories and find the cost-minimizing control strategy for a relatively short future time horizon based on the prediction of the state transitions until the end of the same time horizon. However, only the first step of the control is applied, and then the plant state is sampled again and the calculations are repeated, yielding a new control and a new predicted state transition path. The prediction time horizon keeps being shifted forward. Although this approach is not globally optimal, in practice it yields very good results [42].

  Kermanil et al. [43] in 2009 proposed an MPC scheme for HEV power management. At each time step, a new optimization problem is formulated. The problem over the prediction horizon depends only on a quantity that is characterized based not on the time order sequence but on its distribution. To compute the control over the predicted horizon, the prediction of the time-ordered sequence for the wheel torque demand and wheel speed is required. However, the control is robust with respect to the prediction error. Borhan et al. [44] in their paper compared the liner time variant MPC method with non-linear MPC control and concluded that the non-linear MPC strategy yields a noticeable improvement in fuel economy. In order to find control inputs for a finite time horizon, they used Bellman's optimality theory and approximated the fuel cost-to-go as a piecewise linear function of the SoC.

- Other Related Applications: Other interesting applications related to vehicle energy management include overall emission optimization [45], where the on-road power management and charging strategy are coupled into a simultaneous framework; smart grid charging [46] to optimize the energy usage and minimize the power loss etc.

1.3 New Contributions Made by This Research

In this research, we present a novel Q-learning-based [47] PHEV energy management solution that is model-free and optimal, and is capable of in-vehicle learning. The new contributions made by this research work can be briefly summarized as follows:

- We formulate the power management problem into two categories, optimization for short trips and for long trips. Short trips are those driving cases during which the SOC will not run below the required SoC minimum threshold under any reasonable energy controller. Long trips are those during which the SoC may go under the required SoC threshold if energy flow is not properly managed. An optimal control algorithm has been developed for each category.
- We incorporate estimated future trip information, such as the remaining travel distance and time, with neuro-dynamic programming (NDP) to make in-vehicle learning possible and effective.
- We show that robust convergence of the proposed Q-learning algorithms on both fixed and random trips can be achieved. Furthermore, a novel initialization strategy has been developed that reduces the learning convergence time by 70%.
- Because the proposed learning algorithms are model-free and applicable for in-vehicle learning, we propose a novel two-stage deployment strategy for real-world PHEV energy optimization applications.

Our solution is carried out in two major energy optimization algorithms. Algorithm *QL-ST* is designed for short trips. We first prove, mathematically, that for short trips a greedy control policy that minimizes the immediate step cost is optimal. An analytical greedy control method is derived from the steady-state powertrain model. To eliminate the dependency on any system model, we developed the Q-learning based algorithm QL-ST that can converge on random short trips and is robust against cycle noises. It is well-understood that the theoretical optimal results from deterministic dynamic programming (DDP) cannot be achieved in real simulations. Q-learning, on the other hand, neither relies on any model nor requires any detailed a-priori trip knowledge. It is capable of converging to the real-world achievable optimal control [47]. The learning algorithm *QL-LT* was developed for optimal energy control in long driving trips subject to the SoC minimum constraint. In algorithm QL-LT, we also developed a neural network for the SoC prediction task to make the entire solution model-independent.

The subsequent discussion is organized as follows: Chapter II formulates the PHEV energy optimization problem and presents the background information about Reinforcement Learning (RL) and Q-learning; Chapter III describes the PHEV model used in this research; in Chapter IV, we derive the analytical description of optimal energy control for short trips that do not run SoC under the minimum requirement; Chapter V introduces the two-stage machine-learning paradigm for in-vehicle deployment of the two Q-learning algorithms for optimal energy management; Chapter VI presents the short-trip learning algorithm QL-ST with its evaluation results; Chapter VII presents the algorithm QL-LT with its evaluation on long trips; finally, the conclusion and some interesting future research topics are given in Chapter VIII.

Table 1.1: PHEV Models Commercially Available

Model	Price	All Electric Range
Chevrolet Volt Gen1 or Gen2	\$34K - \$41k	35 - 53 miles
Opel Ampera	\$40K	35 miles
Toyota Prius Plug-in or prime	\$28k-\$32k	11 - 25 miles
Ford C-Max Energi	\$34k	21 miles
Ford Fusion Energi	\$35.5k - \$39.5k	21 miles
Volvo V60 XC90-T8 S60L Plug-in	\$65k - \$68K	27 - 31 miles
Honda Accord Plug-in	\$40K	13 miles
Mitsubishi Outlander PHEV	\$37k	37 miles
BMW i3	\$42k	80 - 100 miles
BMW i8	\$136k	15 - 23 miles
BMW X5 xDrive40e	\$63k	14 miles
BMW 330e or 740e iPerformance	\$45k - \$101k	14 miles
Porsche Panamera S E-Hybrid	\$99k	20 miles
Porsche Cayenne S E-Hybrid	\$76k	14 miles
BYD Qin or Tang	\$31k - \$48K	43 - 50 miles
SAIC Roewe 550 PHEV	\$41K	36 miles
Cadillac CT6 or ELR	\$76k	35 miles
Volkswagen XL1	\$146K	31 miles
Volkswagen Golf or Passat GTE	\$48K	31 miles
Audi A3 Sportback/Q7 e-tron	\$49K	31 miles
Mercedes-Benz C350e or S500e	\$60K - \$146K	14 - 19 miles
Mercedes-Benz GLC 350e or GLE 550e	N.A.	19 miles
Hyundai Sonata PHEV	N.A.	27 miles
Hyundai Ioniq Plug-in	N.A.	31 miles
Kia Optima PHEV	N.A.	27 miles
McLaren P1	\$1,350k	12 miles

Table 1.2: PHEV Models in Pre-production

Model	All Electric Range
Ford Escape Plug-in	30 miles
Volvo V70 Plug-in	12-19 miles
Suzuki Swift Range Extender	15.8 miles
Dodge Ram 1500 Plug-in	20 miles
Chrysler Pacifica Hybrid	33 miles
VW Golf Variant Twin Drive	35 miles
Audi A1 e-tron	31 miles
BMW 530e iPerformance	19 miles
Mini Cooper SE Countryman All4	25 miles
Honda Clarity Plug-in	40 miles

CHAPTER II

PHEV Energy Optimization and Reinforcement Learning

In this chapter, we first formulate the PHEV energy optimization problem and then present the reinforcement learning technology that is the theoretical foundation of the proposed vehicle energy optimization algorithms.

2.1 PHEV Energy Optimization Problem Formulation

We first define a cost function that quantifies the energy usage by taking into account both battery and fuel. The energy consumption in PHEVs consists of two sources, battery and fuel. Therefore, the cost function is defined as the amount of US dollars needed to bring the fuel and battery SoC at the end of the trip to the same level as at the beginning of the trip. The problem under study is subject to the following constraints:

- Physical constraints of the vehicle components such as the rotational speed and torque limits, electrical current limits, power output limits, etc.
- Battery SoC minimum threshold constraint. This is to ensure the state-of-health (SoH) of the battery.

For a driving trip, the vehicle energy system can be modeled as a sequence of *state transitions* over time. At time $t = 0, 1, 2, ..T$, the state of the system s_t belongs to a finite state space $\mathcal{S} \subset \mathbb{R}^n$, where $n \in \mathbb{N}$ is the number of state variables. The state space definition of the power management is a design choice. The state variables are generally chosen as features related to the current and future vehicle energy usage. They may include battery SoC, speed and torque requested by the driver etc. The possible control actions are also bounded by a finite control space $\mathcal{A} \subset \mathbb{R}^m$, where $m \in \mathbb{N}$ is the number of control actions. In the next section, we will show that our PHEV system has two degree of freedom for control. Thus, two actions can be independently produced by the controller at each time step. The state transition is governed by the discrete time equation $s_{t+1} = f(s_t, a_t)$, where $a_t \in \mathcal{A}$ is the control actions taken at the state s_t. The underlying assumption is that the state transitions posses the Markov property [48], i.e., the probability of transitioning to a future state depends on only the current state and the control actions performed, not on the historical evolution. Along with each time step, an *immediate cost* is incurred. So, the overall energy cost equals the sum of all of the step costs. Due to random inputs such as driver's speed and torque requests from the environment, we attempt to minimize the *expected* total cost $\mathcal{C}$ for any trip.

$$\begin{aligned} \text{Minimize } \mathcal{C} = E\Big\{\sum_{k=0}^{\infty} \gamma^k \cdot c(s_k, a_k)\Big\} \\ \text{subject to: } s_{k+1} = f(s_k, a_k) \\ \text{and } SoC \geq SoC_{thrd} \\ \text{and components physical constraints,} \end{aligned} \tag{2.1}$$

where (s_k, a_k) is the state-action pair at time step k. $\gamma \in [0, 1]$ is a *discount factor*. Because every trip has a finite duration, we choose $\gamma = 1$. $c(s_k, a_k)$ is the *immediate cost* incurred at time step k associated with (s_k, a_k). It is defined as follows:

$$c(s_k, a_k) = \frac{\Delta E_{batt,k}}{\eta_{charg}} \cdot u_e + \Delta F_k \cdot u_f, \tag{2.2}$$

where $\Delta E_{batt,k}$ is the electrical energy used in the battery at time step k. Letting C_{batt} represent the total battery capacity and ΔSoC_k the battery SoC variation for time step

k, we have $\Delta E_{batt,k} = \Delta SoC_k \cdot C_{batt}$. In Eq. 2.2, ΔF_k represents the amount of fuel consumed at step k; u_e and u_f are the unit price in US dollars for electricity and fuel; η_{charg} is a constant battery charging efficiency factor.

2.2 Reinforcement Learning in Vehicle Control

In vehicle energy control, an RL *agent* (i.e. *controller*) interacts with the environment, learns from the past control experiences and continuously improves the control actions. At each time step k, the agent produces a set of *control actions* a_k and observes the *immediate reward or cost* $c(s_k, a_k)$ incurred. Function $\pi(\cdot)$, which maps a state to a set of control actions is called *control policy*. This process is illustrated in Fig. 2.1.

The goal of the RL is to achieve the *optimal control policy* that will maximize the total expected reward, or in terms of "*cost*", minimize the total expected cost.

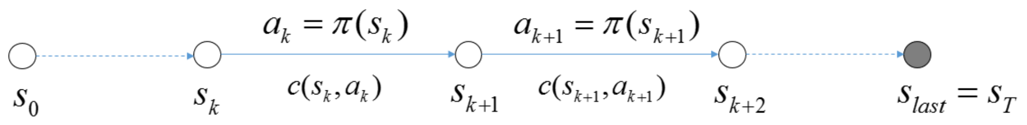

Figure 2.1: An RL problem is modeled as a sequence of state transitions over time.

The RL process can be generally classified as *model-based* or *model-free*. Model-based learning tries to learn an *optimal state-value function* $V^*(s)$, defined as the *minimum expected total future cost*, i.e., the *minimum expected cost-to-go*, following the state s among all the control policies. It has been proven [28] that there exists at least one *optimal policy* π^*, which will generate the control actions leading to the minimum expected cost-to-go. If we assume that the optimal state-value function $V^*(s)$ is learned, then the optimal policy π^* can be derived as

$$\pi^*(s) = a^* = arg \min_a \{c(s, a) + V^*(s_{next}|s, a))\}, \tag{2.3}$$

where $c(s, a)$ is the immediate cost for the state-action pair (s, a); $V^*(s_{next}|s, a)$ is the optimal state-value for s_{next} transitioned from (s, a). It is easy to observe that the policy defined in Eq. 2.3 requires a model to predict both the immediate cost $c(s, a)$ and s_{next} given (s, a).

A model-free RL approach, also referred to as Q-learning, uses a Q-function to estimate the expected cost-to-go from any state-action pair (s, a). So, compared with the model-based state-value estimation, the Q-function essentially tries to estimate the optimal value of each action in a state. A common representation of the Q-function is a look-up table that stores the Q-values. However, due to the huge number of states for the PHEV application, the table look-up approach suffers notably the "Curse of Dimensionality"[27–30]. For this reason, we have developed a neural network (NN) to approximate the Q-function. Similar to the optimal state-value function, an *optimal* Q-function $Q^*(s, a)$ is defined to generate the *minimum* expected cost-to-go from a state-action pair (s, a), and it is also *policy independent*. Assuming the optimal Q-function is learned, then the optimal policy π^* is given by:

$$\pi^*(s) = a^* = arg\min_a \big\{Q^*(s, a)\big\}. \tag{2.4}$$

Eq. 2.4 does not require any model. The optimal policy simply chooses those actions to minimize the optimal Q-function. However, being model-free is normally at the cost of more learning and converging time. For most real-world engineering applications, the knowledge of the underlying model is very limited. For a complicated system such as PHEV, Q-learning offers many advantages over the model-based methods. Although optimal Q-function and optimal policy are both unknown before the learning, there are well-developed learning algorithms that have been mathematically proven to converge to the optimal policy. In this research, we present the RL learning-based algorithm, *the Q-learning with ϵ-greedy exploration* [28, 49], developed for optimal vehicle energy control.

CHAPTER III

Powertrain Control in a PHEV Model

A vehicle model specifies: (1) the state transition governing function defined in Eq. 2.1, (2) the control actions to be produced at each time step, and (3) the immediate cost for a given state-action pair defined in Eq. 2.2.

3.1 PHEV Modeling

We used a midsize power-split PHEV as the testbed of this study. Similar PHEV architecture was adopted by the Toyota Prius Plug-in. The detailed vehicle model has been provided in the Autonomie [50] software. All the simulations in this research were performed using Autonomie. Table 3.1 lists the main vehicle parameters.

The powertrain model equations are dependent on the hybrid configurations. In general, three sets of equations can be derived at the steady state:

- Torque equations: the engine torque, motor torque, and torque demand at the wheel are related through various gear systems.
- Angular Speed equations: relate the engine speed, and motor speed to various gearbox speeds and the wheel speed.
- Power flow equations: energy power should be conserved for the powertrain system.

Table 3.1: Vehicle Parameters

Midsize Power-Split PHEV		
Vehicle	Mass	1126kg
Engine	Type	1.8L Inline 4 Cylinder D.I.
	Power Max	57kw @ 4500rpm
	Torque Max	201Nm @ 4500rpm
Motor/ Generator1	Power Max	66kw @ 1250rpm
	Torque Max	502Nm @ 0-1250rpm
Motor/ Generator2	Power Max	55kw @ 1865rpm
	Torque Max	281Nm @ 0-1865rpm
Battery	Type	39Ah Li-on
	Nominal Cell Volt	3.6v
	Number of Cells	72
	Current Max	145A @ SoC $\geq$ 20%
	Burst Current Max	388A @ SoC $\geq$ 20% for 10s

Basically, the sum of the engine power and motor power discounted by an efficiency factor should be equal to the power at the wheel plus the accessory power. Furthermore, the battery power and motor power can be related through other efficiency factors.

Please note that the power flow equations are redundant to the torque and angular speed equations. So, among the above three sets of equations, only two of them are truly independent.

At non-steady state, the dynamic equations will be used, which will take into account the angular acceleration and inertia of various rotational components such as the gear pinions and the shafts. In this research, we focus on the steady-state behavior of the powertrain model.

In a power-split configuration, the planetary gear set is the key component relating the torque and speed between various components. The planetary gear set imposes one constraint on the angular speeds between the sun gear, ring gear, and carrier. Also, torque at the pinion between the three sub-components are related to each other through a fixed ratio.

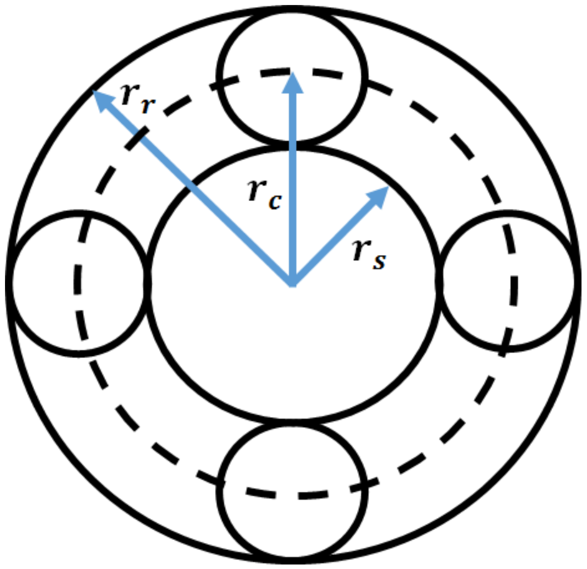

Figure 3.1: A Planetary Gear Set.

A planetary gear set is shown in Figure 3.1, where the outer circle represents the ring gear; the center four connected circles are the carrier and the inner circle represents the sun gear. r_r, r_c and r_s are the radius for ring, carrier and sun gear correspondingly. Let T_r, ω_r, T_c, ω_c and T_s, ω_s be the torque and angular speed of the three components. At the steady state, we have

$$r_c = \frac{r_r + r_s}{2} \tag{3.1}$$

$$\frac{T_s}{r_s} = \frac{T_r}{r_r} = \frac{T_c}{r_s + r_r} \tag{3.2}$$

$$\omega_r r_r + \omega_s r_s = \omega_c (2 r_c) \tag{3.3}$$

The power-split architecture of our PHEV (Toyota Prius Plug-in Hybrid) under study is illustrated in Fig. 3.2, where “S”, “C” and “R” represent sun gear, carrier and ring gear of the planetary. In this configuration, the engine is directly connected to the carrier. Motor1/Generator1 and Motor2/Generator2 are connected to the ring gear and sun gear, respectively. The power flows indicated by the red arrows may be either positive or negative.

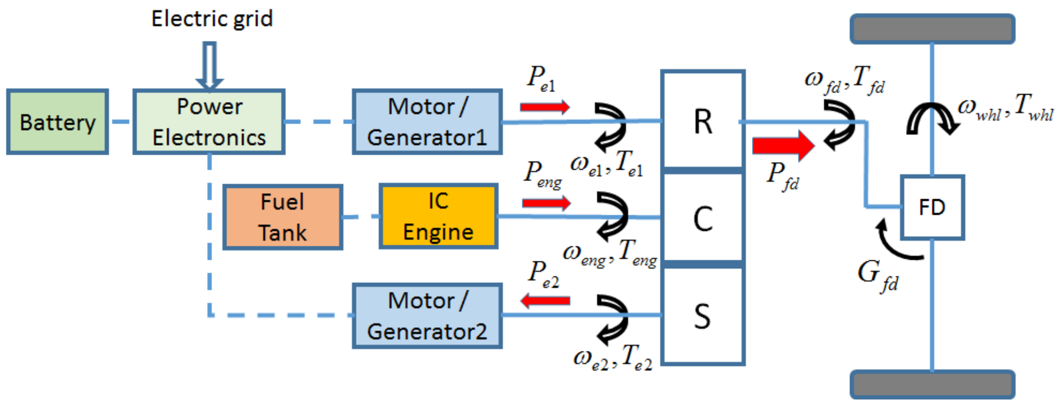

Figure 3.2: A Power-split Hybrid System

The steady state equations of this power split configuration are given in Eq. 3.4:

$$\begin{aligned}
\frac{T_{e2}}{r_s} &= \frac{T_{eng}}{r_s + r_r} \\
\frac{r_r}{r_s + r_r} T_{eng} + T_{e1} &= T_{fd} = \frac{T_{whl}}{G_{fd}} \\
(r_s + r_r)\omega_{eng} &= r_s \omega_{e2} + r_r \omega_{e1} \\
\omega_{e1} &= \omega_{fd} = G_{fd}\omega_{whl},
\end{aligned} \tag{3.4}$$

where T_{eng}, T_{e1}, T_{e2}, T_{fd}, and T_{whl} are the torques for engine, motor1/generator1, motor2/generator2, final drive, and wheel; ω_{eng}, ω_{e1}, ω_{e2}, ω_{fd}, and ω_{whl} are the angular speed for engine, motor1/generator1, motor2/generator2, final drive, and wheel; r_r, r_c, and r_s are the radius for ring, carrier and sun gear; and G_{fd} is the gear ratio of the final drive.

At a time step, the T_{whl} and ω_{whl}, i.e., the wheel torque demand, and the wheel speed demand, are the driver inputs, which can be measured from the sensors. Therefore, the above four equations have six variables: T_{eng}, T_{e1}, T_{e2}, ω_{eng}, ω_{e1}, and ω_{e2}. This implies that the degree of freedom for this control problem is two. For example, our controller is allowed to generate any two of the six variables as the control actions at each time step; the rest of the variables can be then resolved from Eq. 3.4, and the performance requested by the driver is automatically guaranteed.

3.2 Power Flow Modeling

For our PHEV system, the following power flow equations hold:

$$\begin{aligned} P_{fd} &= P_{e1} + P_{e2} + P_{eng} \\ P_{bo} &= \gamma_1 P_{e1} + \gamma_2 P_{e2}, \end{aligned} \tag{3.5}$$

where $P_{eng} = T_{eng}\omega_{eng}$, $P_{e1} = T_{e1}\omega_{e1}$, and $P_{e2} = T_{e2}\omega_{e2}$ represent the power output of the engine, the motor1, and the motor2; $P_{fd} = T_{fd}\omega_{fd}$ is the power demand at the final drive. P_{bo} is the power at the output of the battery illustrated in Fig. 4.2. γ_1 and γ_2 are related to the efficiency factors η_{e1} and η_{e2} of the two motors and inverters based on charging or discharging situation. (The motor power is negative when charging the battery.)

$$\gamma_1 = \begin{cases} \frac{1}{\eta_{e1}} & \text{if } P_{e1} \geq 0 \\ \eta_{e1} & \text{if } P_{e1} < 0 \end{cases} \quad \gamma_2 = \begin{cases} \frac{1}{\eta_{e2}} & \text{if } P_{e2} \geq 0 \\ \eta_{e2} & \text{if } P_{e2} < 0. \end{cases} \tag{3.6}$$

The power flow equations essentially introduced a new variable P_{bo} with one new equation, so the overall degree of freedom of the system is not affected. For a small time step Δt where the P_{bo} can be considered as a constant, we can estimate the ΔSoC based on a simplified battery model. The details are provided when we derive the analytical greedy control actions in the next section. Therefore, Eq. 3.4 and Eq. 3.5 allow us to predict the immediate cost as well as to derive the next transition state based on a given state and power split control action.

CHAPTER IV

OPTIMAL PHEV POWER MANAGEMENT USING ANALYTICAL GREEDY CONTROL FOR SHORT TRIPS

In this chapter, we focus on energy optimization for short trips. We define a driving trip as a ***short trip*** if the battery SOC will not drop below the minimum threshold throughout the entire trip. Based on the current battery technologies in PHEV, short trips are generally characterized by less than 30 minutes of duration with a fully charged battery at the beginning of the trip.

The surveys [51–53] from The United States Census Bureau claim that about 86% of U.S. workers commute from home to work by automobile with a mean travel time of around 26.1 minutes and a mean travel distance of about 18.8 miles. Therefore, energy optimization for shorts trips represents a strong value for practical applications.

We will first introduce our analytical solution for the optimal power management of short trips. The proposed analytical approach is based on the powertrain model described in Chapter III. Without losing generality, we assume that at the end of a *trip*, the battery will be recharged from the electric grid. This charging event will reset our controller for a new trip with a full battery.

In the literature, the battery SoC has always been considered as one of the state variables for power management. However, it is very important to emphasize that *the SoC gives **no indication** about the future trip cost-to-go unless the trip is constrained by a minimum SoC level*. We prove that, without this constraint, a greedy control policy simply yields the optimal cost.

4.1 The Greedy Policy is Optimal for Short Trips

Proposition IV.1. *The optimal solution for the PHEV power management problem can be obtained by using the greedy control policy if the component dynamics and the battery SoC minimum threshold are not considered, or in other words, if we assume that the battery's capacity is infinite within the whole trip.*

A **greedy policy** *is defined as a policy that produces the control actions while minimizing the immediate cost [28].*

$$\pi^{greedy}(s) = arg\min_{a}\big\{c(s,a)\big\}. \tag{4.1}$$

Proof. As shown in Fig. 4.1, let us assume the vehicle is currently at state s. A control action a_1 will cause the system to transition to the next state s_1 with an immediate cost c_1. Similarly, action a_2 will lead the system to the state s_2 with an immediate cost c_2.

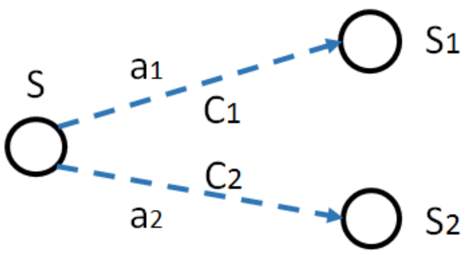

Figure 4.1: From state s, different control actions a_1 and a_2 will cause the system to transition to different future states s_1 and s_2 with the immediate cost c_1 and c_2, respectively.

It is important to observe that *the future states s_1 and s_2 essentially differ on the battery SoC*. If the battery's capacity is not a concern, then the optimal control actions for s_1 or s_2 will not be dependent on the current SoC. Furthermore, the transient dynamics related to the engine's and motor' speed and torque do not cause significant future cost-to-go variations for the real-world applications. By ignoring the component dynamics, the optimal control actions on s_1 and s_2 are not dependent on their current engine and motor speed and torque. As a result, the optimal control actions following s_1 and those following s_2 will be the same. This has an important implication: s_1 and s_2 will have the same optimal cost-to-go. In other words, the optimal cost-to-go of the next state is independent of the control actions we choose for s. We can write the above statement using Eq. 4.2.

$$V^*(s_1) = V^*(s_2) \Longrightarrow V^*(s_{next}|s,a) = V^*_{next}(s), \tag{4.2}$$

where $V^*(s_{next}|s,a)$ represents the optimal expected cost-to-go of the next state s_{next} transitioned from the state-action pair (s,a); $V^*_{next}(s)$ indicates that the optimal cost-to-go $V^*(s_{next})$ is actually a function only of s but not a.

From Eq. 2.3, we know that the optimal policy is given by:

$$\begin{aligned} \pi^*(s) &= arg\min_a \big\{ c(s,a) + V^*(s_{next}|s,a) \big\} \\ &= arg\min_a \big\{ c(s,a) + V^*_{next}(s) \big\} \\ &= arg\min_a \big\{ c(s,a) \big\}. \end{aligned} \tag{4.3}$$

Eq. 4.3 proves that the optimal policy is the one that minimizes the immediate cost $c(s,a)$. By definition, this is the greedy control policy. In other words, the optimal cost of the whole trip can be obtained by minimizing the cost of each single step. $\square$

4.2 Analytical Solution for the Greedy Policy

We will begin with the steady-state powertrain Eq. 3.4 to derive our greedy control policy analytically. Since we have two degrees of freedom to choose our control actions, the immediate cost function has two control variables, whose values should be properly chosen in order to minimize the cost function.

At each time step, the immediate cost associated with a state-action pair (s, a) is given in Eq. 2.2. We can further rewrite this cost using the following:

$$c(s,a) = \alpha P_{eng} \Delta t u_f + \beta P_{batt} \Delta t u_e. \tag{4.4}$$

In Eq. 4.4, the fuel rate is modeled as a linear function of the engine power (Zhang et al. [8, 15]). We also assume the electrical cost being linearly proportional to the battery power. α and β are two linear coefficients, respectively. P_{eng} and P_{batt} are the constant engine and battery power output for this small time step Δt; u_f is the unit price of the fuel in dollars per gallon; and u_e is the unit price of the electricity in dollars per KWH.

From Eq. 3.5 we have $P_{eng} = P_{fd} - P_{e1} - P_{e2}$ and $\omega_{e2} = \frac{P_{bo} - \gamma_1 T_{e1} \omega_{fd}}{\gamma_2 T_{e2}}$, where we replaced ω_{e1} with ω_{fd}. So we can rewrite the immediate cost as:

$$\begin{aligned} \frac{c(s,a)}{\Delta t} &= \alpha (P_{fd} - P_{e1} - P_{e2}) u_f + \beta P_{batt} u_e \\ &= \alpha \big[P_{fd} - T_{e1}\omega_{fd} - \frac{1}{\gamma_2}(P_{bo} - \gamma_1 T_{e1} \omega_{fd}) \big] u_f \\ &\quad + \beta P_{batt} u_e. \end{aligned} \tag{4.5}$$

We use a theoretical battery model to relate P_{bo} with P_{batt}. As shown in Fig. 4.2, assuming the battery open voltage V_{oc} and the internal resistance R are constants, we have:

$$P_{bo} = P_{batt} - \frac{P_{batt}^2}{V_{oc}^2} R. \tag{4.6}$$

Our goal is to represent the immediate cost as a function of the battery power P_{batt} and the engine torque T_{eng} using Eq. 4.5. Based on Eq. 3.4, we can replace T_{e1} with $(T_{fd} -$

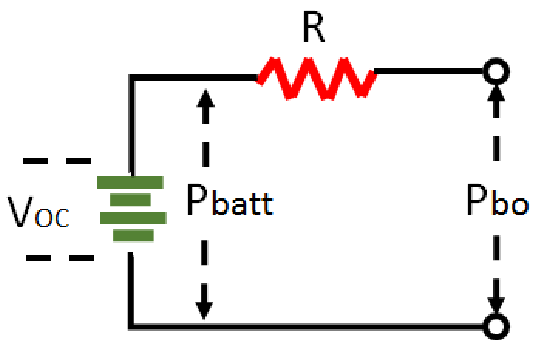

Figure 4.2: A simplified battery model with a constant open circuit voltage V_{oc} and a constant internal resistance R. The SoC change is driven by P_{batt}, and the useful power output is P_{bo}.

$\frac{1}{1+\rho}T_{eng})$, where $\rho = \frac{r_r}{r_s+r_r}$, and replace P_{bo} using Eq. 4.6. So the immediate cost can be represented as:

$$\frac{c(s,a)}{\Delta t} = AP_{batt}^2 + BP_{batt} + C, \tag{4.7}$$

where we have

$$A = \frac{\alpha R}{\gamma_2 V_{oc}^2}u_f \tag{4.8}$$

$$B = \beta u_e - \frac{\alpha}{\gamma_2}u_f \tag{4.9}$$

$$C = \big[(1-\frac{\gamma_1}{\gamma_2})(\frac{1}{1+\rho})T_{eng}\omega_{fd} + \frac{\gamma_1}{\gamma_2}P_{fd}\big]\alpha u_f. \tag{4.10}$$

Eq. 4.7 provides a parabolic form allowing us to find the minimum immediate cost, where A and B are two constants and C is a linear function of the engine torque T_{eng}. The resultant 2nd order parabolic function is mainly caused by the battery internal resistance term. Therefore, a controller that always maximizes the battery energy usage against fuel may not lead to the optimal cost due to the energy dissipation on the battery's internal resistance.

Eq. 4.7 also nicely decouples the dependency of the immediate cost on P_{batt} and on T_{eng}. In the subsequent discussion, we will show how to independently choose the best values for P_{batt} and T_{eng} to minimize the immediate cost. We will also study the impact of the electricity and fuel price ratio on the optimal control strategy.

Fig. 4.3 shows the parabolic function defined in Eq. 4.7, where we have:

$$P^*_{batt} = -\frac{B}{2A} = (\frac{V_{oc}^2}{2R})(1 - \gamma_2 \frac{\beta}{\alpha}\frac{u_e}{u_f}) \tag{4.11}$$

$$\frac{c^*}{\Delta t} = C - \frac{B^2}{4A}. \tag{4.12}$$

At a high-level summary, the cost minimization can be achieved by first selecting the optimal battery power P^*_{batt} and then choosing the proper value for T_{eng} to minimize c^*.

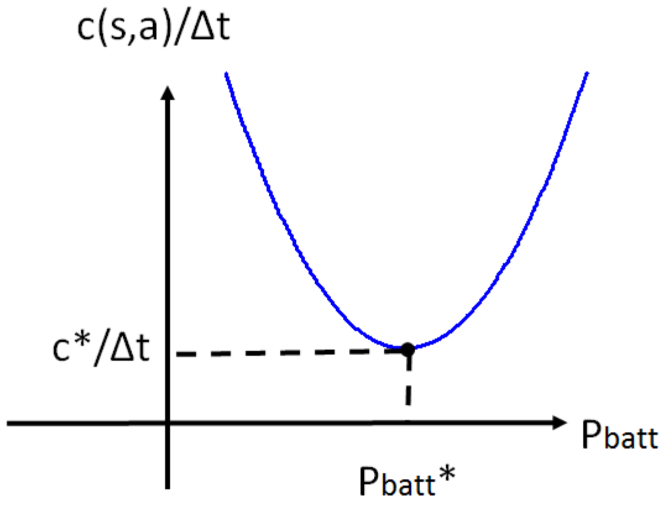

Figure 4.3: The immediate cost in function of P_{batt} assumes a parabolic form.

For more conveniently describing our greedy control, we introduce a new variable P^*_{fd}, which corresponds to the best battery power P^*_{batt} but converted at the point of the final drive.

$$P^*_{fd} = \frac{1}{\gamma_1} P^*_{bo} = \frac{1}{\gamma_1}(P^*_{batt} - \frac{{P^*_{batt}}^2}{V_{oc}^2} R) \tag{4.13}$$

From Eq. 4.7 and using the P^*_{fd} defined in Eq. 4.13, we derive the following greedy control strategy:

- If the power demand from the final drive P_{fd} falls in the range of $[0, P_{fd}^*]$, then we would disable the engine output power and use only the battery to provide the power needed. This case is represented by the left side of the parabola where higher P_{batt} leads to lower cost.
- If P_{fd} falls in the range of $[P_{fd}^*, P_{fd}^* + P_{eng\_max}]$, where P_{eng_max} is the engine power upper limit, we would limit the battery power to P_{batt}^* and find the best T_{eng} value described below to minimize the immediate cost.
- If $P_{fd} > P_{fd}^* + P_{eng\_max}$, we would limit the engine to its maximum power output and let battery to provide all of the rest of the power needed.
- If $P_{fd} < 0$, then we would disable the engine torque output and use regenerative braking to charge the battery as much as possible.

The term C in Eq. 4.7 and Eq. 4.12 is a linear function of T_{eng}. However, the slope of this line can be either positive or negative based on the value $(1 - \frac{\gamma_1}{\gamma_2})$, i.e., based on whether the motors/generators are providing the propulsion or charging the battery. This behavior is illustrated in Fig. 4.4, where we have:

$$T_1^* = (1+\rho)(T_{fd} - \frac{1}{\gamma_1}\frac{P_{bo}^*}{\omega_{fd}}) \tag{4.14}$$

$$T_2^* = (1+\rho)T_{fd} \tag{4.15}$$

$$C_{min}^* = \alpha\frac{\gamma_1}{\gamma_2}P_{fd}u_f. \tag{4.16}$$

In region(I), the motor1 is outputting the power, but motor2 is working as a generator to charge the battery, so the cost is decreasing as T_{eng} increases; in region(II) both motors are providing power, and thus the slope $(1 - \frac{\gamma_1}{\gamma_2})$ could be positive, negative or zero depending on the value of η_{e1} and η_{e2}; in region(III), the torque of the engine is so big that motor1 starts charging the battery and motor2 still produces positive power, the slope of the line becomes positive and the cost increases as T_{eng} increases. Therefore, we would choose

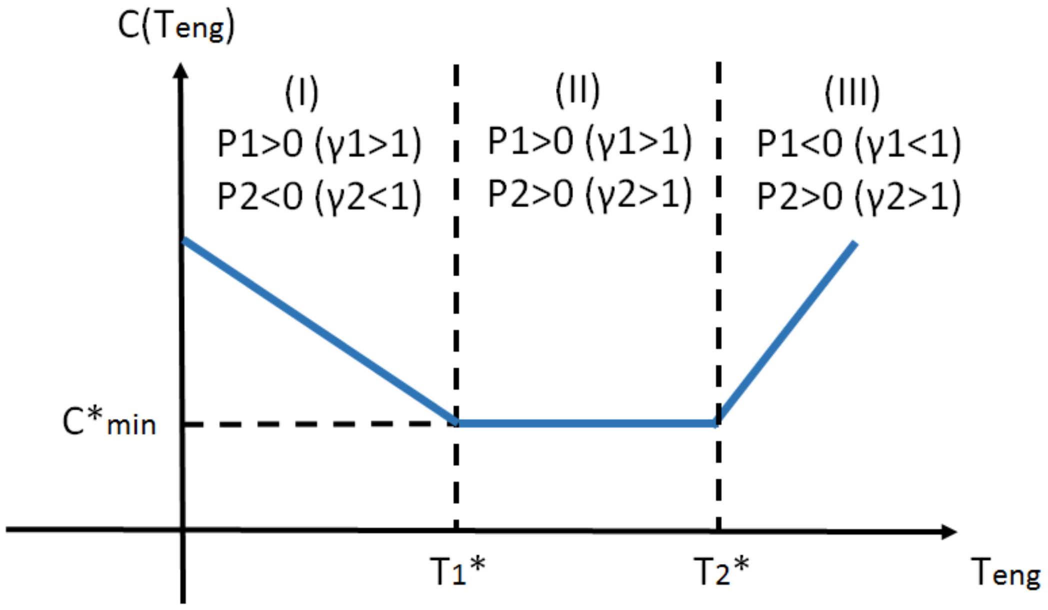

Figure 4.4: The term C in Eq. 4.12 is a linear function of T_{eng}.

$T_{eng} = T_1^*$ if $\eta_{e1} > \eta_{e2}$; $T_{eng} = T_2^*$ if $\eta_{e1} < \eta_{e2}$; or set T_{eng} to any value in the range of $[T_1^*, T_2^*]$ if $\eta_{e1} = \eta_{e2}$.

It is interesting to observe that in Eq. 4.11, the optimal battery power P_{batt}^* is linearly dependent on the fuel and electricity price ratio $\frac{u_e}{u_f}$. As shown in Fig. 4.5, P_{batt}^* will decrease as the price ratio $\frac{u_e}{u_f}$ increases. When $\frac{u_e}{u_f} > r_{u_e/u_f}^*$, the electricity becomes so expensive that we would not use the battery power at all. The r_{u_e/u_f}^* is about 0.09, expressed in Gallon/KWH. In our simulation, we set $u_e = \$0.12/KWH$ and $u_f = \$2.5/Gallon$. This yields the price ratio of 0.048.

It is possible to predict the end-of-trip SoC using the optimal battery power given in Eq. 4.11 based on some attainable trip estimations. We have selected five features for this prediction: travel distance; travel time, excluding the stop time; average speed, excluding the stop time; time duration when the acceleration is greater than $0.5m/sec^2$; and time duration when the deceleration is stronger than $-0.5m/sec^2$. Our experiments show that a simple linear regression model over these five features can make an excellent prediction

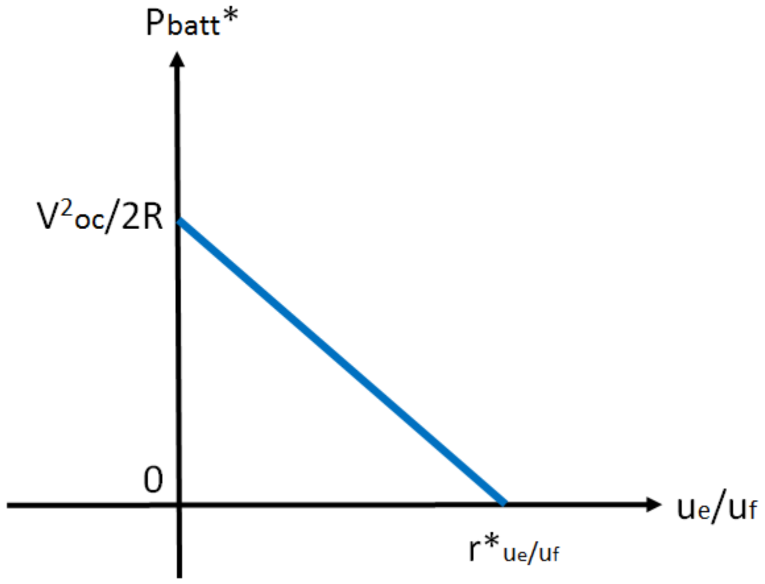

Figure 4.5: The optimal battery power P^*_{batt} is a linear function of the electricity/fuel price ratio.

of whether a trip is short enough for the greedy policy to be applied. For example, we estimate that our analytical greedy actions will cause the SoC to drop about 27% for the UDDS cycle. Thus, as long as the battery is initially charged above 57% (assuming the SoC minimum constraint is at 30% level), we can safely apply the greedy control on the UDDS.

Table 6.4 compares the costs of the analytical greedy strategy with those of the default rule-based control and other Q-learning controllers, which will be covered in the next section. The greedy policy is able to outperform the rule-based control for most of the short drive cycles.

Fig. 4.6 shows the simulated battery power P_{batt} of the greedy policy vs the rule-based control on the US06 drive cycle. The dotted red horizontal line marks the optimal battery power level P^*_{batt} defined in Eq. 4.11. The greedy policy attempts to set the battery power to this optimal level for the purpose of minimizing the immediate cost. This behavior is illustrated by those horizontal red segments taking more than 60% of the entire trip duration. As a result, the greedy policy yields a 16.9% lower trip cost. The red power oscillations are mainly caused by the battery transient dynamics associated with the braking events. The rule-based control (blue line) uses less battery power and more fuel, yielding a higher energy cost and higher emissions.

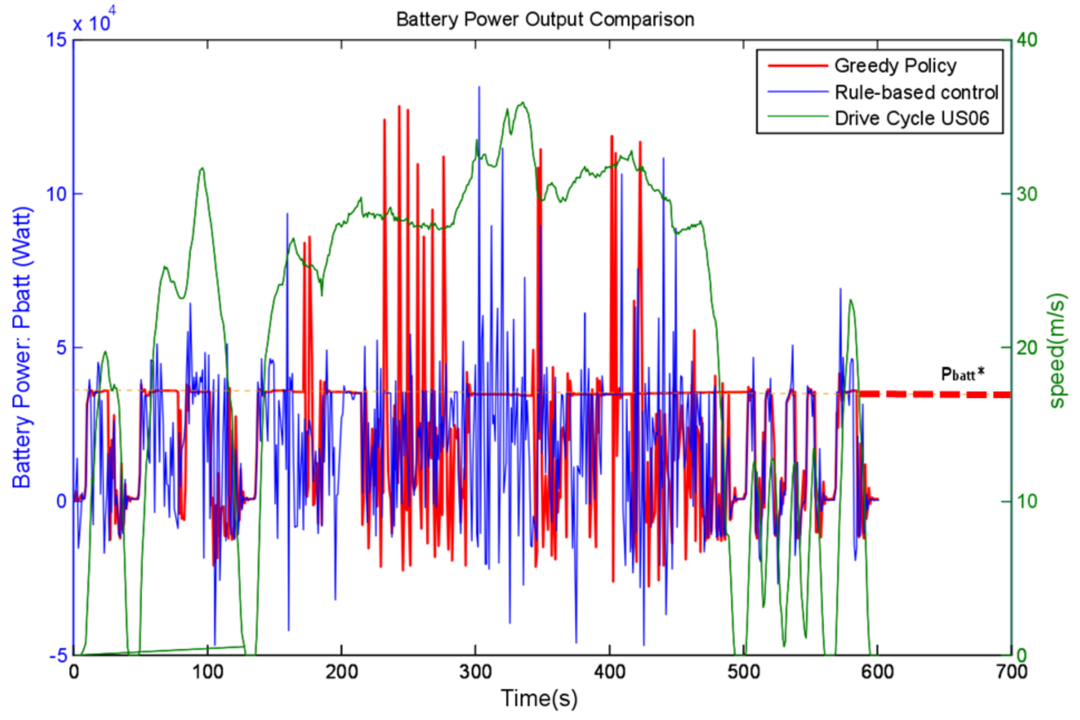

Figure 4.6: Illustration of battery powers generated on drive cycle US06 using two different policies, a greedy control policy and a rule-based controller.

CHAPTER V

A Model-free, In-vehicle Learning Paradigm for Optimal PHEV Power Management

The analytical greedy control solution introduced above relies on simplified powertrain and battery models. For real-world applications, model identification normally involves tedious development and validation efforts. We propose to build a model-free energy controller using Q-learning. As introduced in Section II, it is a model-free learning approach where a controller interacts with the environment and tries to learn an optimal cost-to-go function for any given state-action pair.

5.1 Introduction to the proposed Q-learning Algorithms

We propose two machine-learning algorithms, *QL-ST* and *QL-LT*, for optimizing energy control for the short-trip and long-trip categories, respectively. The core of the two algorithms is a Q-function that accurately estimates the expected future cost-to-go at any given state-action pair (s, a). This function is continuously updated by the controller using only the observed state transitions and costs that are available at the end of each driving trip. The Q-function is approximated by using a neural network (NN). This method is called neuro-dynamic programming (NDP) [27, 32]. The traditional look-up table approach is seriously limited by the "curse of dimensionality" [28] due to the large state space we are dealing with, and it has limited capability to adapt to dynamic environments.

Both algorithms use estimated future trip information, namely, remaining travel distance and time, which are available in most modern vehicles, as input to the neural network designed to learn to estimate the expected cost-to-go. The future trip information is closely related to the future energy consumption; therefore this is effective input to the neural network for learning the cost-to-go function. For long trips, algorithm QL-LT further enforces a battery power cutoff rule to prevent the battery from dropping below its minimum SoC threshold.

For each category, we will evaluate the proposed algorithm by analyzing its convergence on power management applications, a topic not being discussed in the previous literature. In general, the benefit of model-free learning is at the cost of bigger action exploration and longer convergence time. For this reason, we also developed a new initialization strategy for the long-trip learning algorithm QL-LT so that the convergence time can be significantly reduced.

5.2 Deployment Strategy of the Q-learning Algorithms

Our research experiments have shown that both Q-learning algorithms can converge on both fixed and random drive cycles (see Chapter VI and VII), and they yield better performance than the analytical or rule-based controls. This implies that we can directly deploy our algorithms to real-world vehicle energy management to learn optimal control functions. However, our experimental results also show that the number of trips, i.e. learning iterations, needed to achieve the convergence is dependent on the type and length of the cycles. In order to minimize the in-vehicle learning period, i.e., to make the in-vehicle learning converge faster, we propose a two-stage machine-learning paradigm (see Fig. 5.1) for practical implementation of two Q-learning algorithms in vehicle systems.

Stage-1 learning is carried out during the vehicle engineering development period. The QL-ST algorithm performs Q-learning on random short ships, e.g., randomly selected short standard driving cycles, until it converges. The result of the QL-ST learning is an optimal vehicle energy controller *VEC-ST* applicable to short trips. For long-trip learning, QL-LT uses VEC-ST with a proper penalty function to initialize the learning process. We will show that with this initialization step, QL-LT can reduce the learning convergence time by 70%. The output of QL-LT is another optimal energy controller *VEC-LT* for energy control in long trips.

At Stage-2, QL-ST will continue the learning process in-vehicle on the customer trips dedicated to home/work commutes. Our experimental results show that QL-ST can converge robustly on repeated drive cycles even with a certain level of noise (see Section VI). For long trips, QL-LT uses a very small exploration and learning rate during in-vehicle learning. This will make the controllers be finely tuned to each individual vehicle, driving routes, and driving style.

Fig. 5.2 illustrates the details of Stage-1 learning. The block on the left side describes the learning algorithm QL-ST on short trips, which should be completed before the long-trip learning begins. The block on the right side shows the Q-learning algorithm QL-LT on long trips subject to the SoC constraint. Long-trip learning is initialized using the following three components: the optimal controller *VEC-ST* learned from short trips; a properly selected penalty function W(·), and a trained neural network NN_{SoC} for the SoC prediction task. Finally, *VEC-LT* represents the converged optimal controller for long trips. The entire learning process is completely model-independent. Sections VI and VII present detailed descriptions of the QL-ST and QL-LT algorithms, respectively.

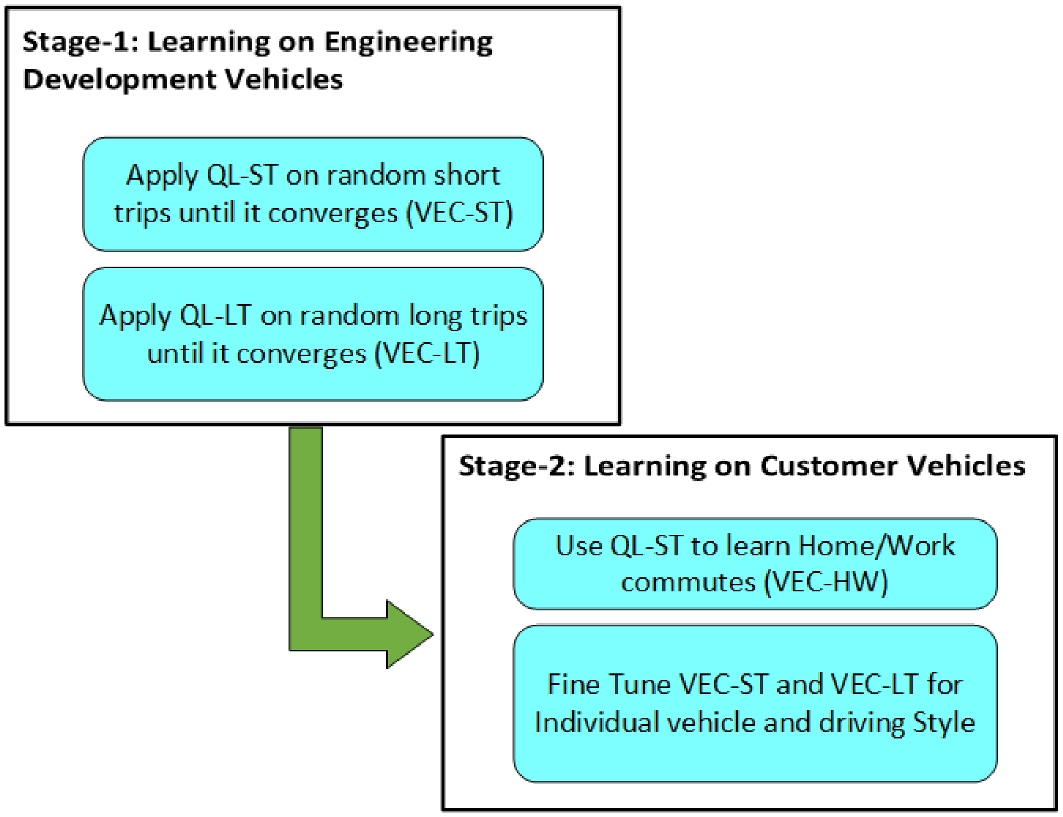
Stage-1: Learning on Engineering Development Vehicles
Apply QL-ST on random short trips until it converges (VEC-ST)
Apply QL-LT on random long trips until it converges (VEC-LT)
Stage-2: Learning on Customer Vehicles
Use QL-ST to learn Home/Work commutes (VEC-HW)
Fine Tune VEC-ST and VEC-LT for Individual vehicle and driving Style

Figure 5.1: Two-stage Q-Learning for real-world deployment.

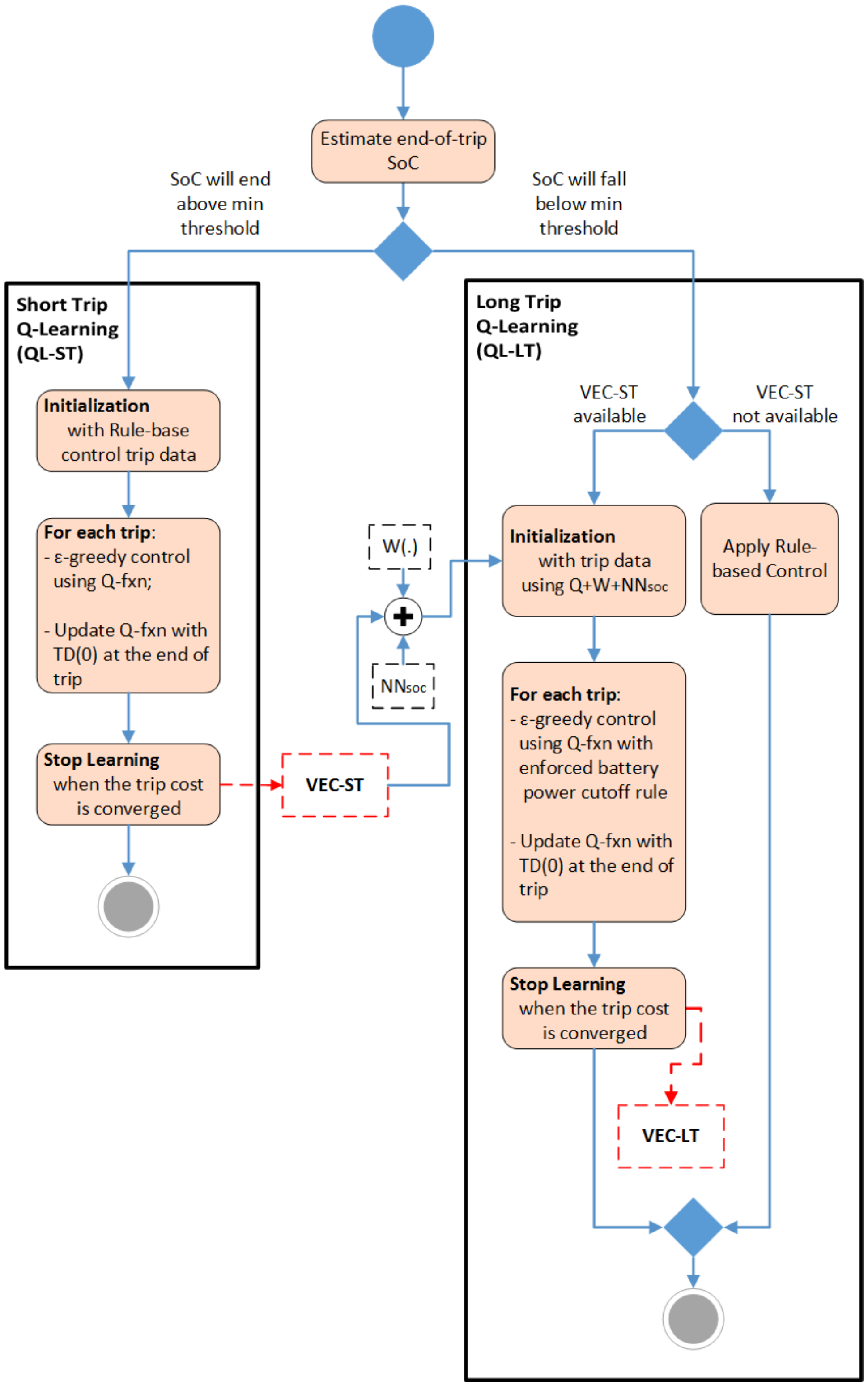
Estimate end-of-trip SoC
SoC will end above min threshold
SoC will fall below min threshold
Short Trip Q-Learning (QL-ST)
Initialization with Rule-base control trip data
For each trip: - ε-greedy control using Q-fxn; - Update Q-fxn with TD(0) at the end of trip
Stop Learning when the trip cost is converged
VEC-ST
W(.)
NNsoc
Long Trip Q-Learning (QL-LT)
VEC-ST available
VEC-ST not available
Initialization with trip data using Q+W+NNsoc
Apply Rule-based Control
For each trip: - ε-greedy control using Q-fxn with enforced battery power cutoff rule - Update Q-fxn with TD(0) at the end of trip
Stop Learning when the trip cost is converged
VEC-LT

Figure 5.2: Stage-1 Q-Learning on engineering development vehicles.

CHAPTER VI

Q-Learning Algorithm QL-ST for Short Trips and Its Evaluation

In this chapter, we will provide details about the QL-ST algorithm and study its convergence and robustness on fixed and random short trips against cycle noises. We will also evaluate the performance of the optimal controller VEC-ST generated by QL-ST.

6.1 QL-ST Algorithm Description

QL-ST is a machine-learning algorithm developed for learning optimal vehicle energy control for short trips. It is developed based on Q-learning and neuro-dynamic programming. It takes a pre-trained neural network ($NN^{(0)}$) and a training data set as input and outputs a neural network (NN^{*}) that accurately approximates the Q-function to estimate the optimal cost-to-go for any state-action pair. The NN^{*} is used for the optimal control policy defined in Eq. 2.4 in the final controller VEC-ST.

Table 6.1 summarizes the setup of the QL-ST algorithm. The state variables include driver's torque request converted at the final drive (T_{fd}); driver's speed request converted at final drive (ω_{fd}); the remaining travel distance (l_r); the remaining travel time, excluding the stop time (t_r); and the average vehicle speed for the remaining trip, excluding the stop time

($\bar{v}_r$). Algorithm QL-ST produces two control actions T_{eng} and ω_{eng} at each time step using the ϵ-greedy control policy. The rest of the actions can be resolved from Eq. 3.4, which guarantees the vehicle performance requested by the driver. A NN is used to approximate the Q-function. The input to the NN are scaled state variables and control actions. The output of the NN is the estimated cost-to-go for the state-action pair presented at the input.

Table 6.1: Summary of QL-ST

State Variables (s):	T_{fd}, ω_{fd}, l_r, t_r, $\bar{v}_r$
Control Action (a):	T_{eng}, ω_{eng}
NN Input:	State-Action pair (s, a)
NN Output:	Expected future cost-to-go for (s, a), i.e. $Q(s, a)$
Control Policy:	ϵ-greedy
Q-func. Update:	Temporal Difference with learning rate (NN weights updated at the end of each trip)

The NN illustrated in Fig. 6.1 is a multilayer perceptron network with H neurons in the hidden layer.

During a trip, at each time step, we use the ϵ-greedy policy to generate control action $a = (T_{eng}, \omega_{eng})$. In order to constrain our *Action Space* to be a finite space, we discretize the engine speed into a vector of $[0 : \Delta_\omega : \omega_{eng\_max}]$. For a fixed engine speed, the engine torque is discretized into $[0 : \Delta_{trq} : T_{eng\_max}(\omega_{eng})]$, where $T_{eng\_max}(\omega_{eng})$ is the max engine torque for a given speed ω_{eng}; Δ_ω and Δ_{trq} are the discretization steps.

We also limit our action space such that all of the components' physical constraints are strictly followed. Our control policy is deterministic, meaning that for a given state, this policy will output deterministic control actions. This is different from stochastic DP (SDP), where each possible control action is accompanied by a probability.

The QL-ST algorithm is designed such that it can incrementally optimize the NN based on

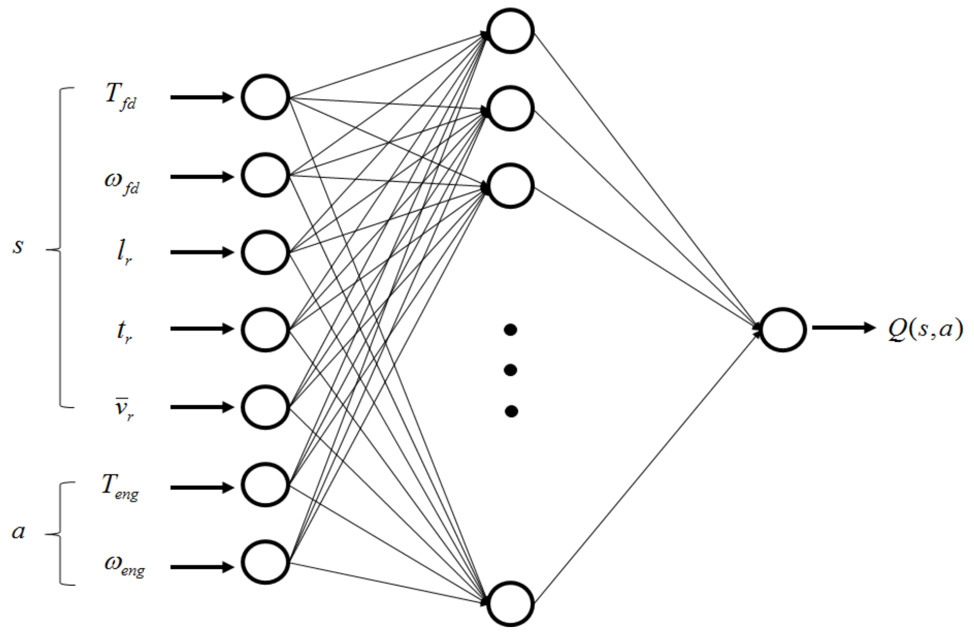

Figure 6.1: The NN used to approximate the Q-Function has seven inputs, one output, and one hidden layer of H neurons.

newly available training data. The whole learning process is carried out while the algorithm is generating the power management control actions. Therefore, the QL-ST algorithm can be used for in-vehicle online learning. The pre-trained neural network input to the QL-ST algorithm, $NN^{(0)}$, can be initialized using a rule-based control method, which will be described below.

We train the NN at the end of each trip. After the NN is trained at the end of trip i using the collected data from trip i, it will be used to generate control actions during the next trip $i + 1$. This process continues until convergence of the trip energy cost is achieved (see Fig. 6.2 as an example). The cost convergence is discussed in detail in the next subsection. In summary, with the continued QL-ST learning from a sequence of new training data, we obtain a sequence of NNs: $NN^{(1)}$, $NN^{(2)}$, $NN^{(3)}$, ... NN^{*}, where $NN^{(i)}$ is the NN trained at the end of trip i. NN^{*} is the final converged one, which approximates the ***optimal*** Q-function. The following describes the training data used as input to all the learning algorithms presented in this paper, including QL-LT.

Any trip i is recorded by the algorithm as a sequence of ***observed*** state-action pairs $\{(s_k, a_k)\}^{(i)}$; $k = 1, 2, ...N$, is the index of each time step. As indicated in Table 6.1,

$$(s_k, a_k) = \{(T_{fd}^k, \omega_{fd}^k, l_r^k, t_r^k, \bar{v}_r^k); (T_{eng}^k, \omega_{eng}^k)\}, \tag{6.1}$$

Besides all the state-action pairs, the QL-ST algorithm also records the energy cost for each time step $\{c_k\}^{(i)}$, $k = 1, 2, ...N$.

The three main parts of the algorithm are described below:

- **Pre-training of** $NN^{(0)}$ We use the rule-based control to make one trip. Any kind of trip can serve the purpose of the pre-training. In our case, we use a UDDS drive cycle. We then compute the ***recorded cost-to-go*** value at each time step k:

  $$ctg_k = (c_k + c_{k+1} + c_{k+2} + ... + c_N) = \sum_{j=k}^{N} c_j, \tag{6.2}$$

  The initial training data set is composed of $\{(s_k, a_k), ctg_k\}$, $k = 1, 2, ...N$, where (s_k, a_k) are the observed state-action pairs presented as the inputs to the NN, and ctg_k are the target values. The result of this initial training is $NN^{(0)}$.

- **Control actions generation during each trip** During a trip i, at any time step k, we observe the current state $s_k = (T_{fd}^k, \omega_{fd}^k, l_r^k, t_r^k, \bar{v}_r^k)$. We combine s_k with all legal control actions $a = (T_{eng}, \omega_{eng})$ and present each pair (s_k, a) to $NN^{(i-1)}$. We then choose the action a_k that minimizes $NN^{(i-1)}$ as the control action generated at this time step k. For a small probability ϵ_i, we choose a control action randomly from all the legal actions. This is called ϵ-greedy control.

  $$a_k = \begin{cases} arg\min\limits_{a \in A}\{NN^{(i-1)}(s_k, a)\}, & \text{probability } 1 - \epsilon_i, \\ \text{a random action in } A, & \text{probability } \epsilon_i. \end{cases} \tag{6.3}$$

  where A is the action space containing all the *legal* actions. For each possible action $a = (T_{eng}, \omega_{eng})$, we make sure it first does not violate the engine power, speed, and torque constraints. We then use Eq. 3.4 to derive other control variables. We consider

a as a *legal* action only if all the other physical constraints associated with motors and battery are also met.

- **Subsequent NN training at the end of each trip** At the end of trip i, the cost-to-go at time step k is computed based on the temporal difference (TD(0)) algorithm:

$$\begin{aligned} ctg_k = &(1-\alpha_i)NN^{(i-1)}(s_k, a_k) \\ &+ \alpha_i[c_k + \min_a NN^{(i-1)}(s_{k+1}, a)] \end{aligned} \tag{6.4}$$

In other words, the target value ctg_k is computed as the observed step cost c_k plus the minimum value of the $NN^{(i-1)}$ evaluated at s_{k+1}. Then this value is adjusted by a learning rate α_i. So, the entire training set is $\{(s_k, a_k), ctg_k\}, k = 1, 2, ...N$. The NN after the training is $NN^{(i)}$, which will be used during the next trip $i+1$ to generate the control actions as described above.

Eq. 6.4 is derived from the temporal difference (TD(0)) method [28, 49]. The original TD(0) algorithm learns on the difference between temporally successive predictions. The predicting function is updated at each time step. However, NN training on the fly is computationally intensive. We modified the original TD algorithm by making the learning update only at the end of the trip. This will help increase the controller's real-time processing throughput. The entire learning iteration process is described in Algorithm QL-ST.

6.2 Evaluation of QL-ST on Fixed Drive Cycles

Algorithm QL-ST was first tested on the repeated UDDS cycles. Each iteration in this case runs through one complete UDDS trip. Fig. 6.2 demonstrates the convergence of the Q-learning for 1000 iterations (i.e., 1000 UDDS trips). Each small circle in the figure represents the recorded energy cost for that trip. The red curve is the cost moving average with a window size of 30 iterations. It represents the trend of the trip cost. During the learning process, the QL-ST algorithm achieved the minimum of \$0.39 after 687 UDDS cycles. The

```
Algorithm QL-ST
Input: $NN^{(0)}$ trained using the trip data of a rule-based control
        multiple trip data recorded $\{(s_k, a_k), c_k\}^{(i)}_{i=0,1,2,...}$
        exploration schedule $\{\epsilon_i\}_{i=0,1,2,...}$
        learning rate schedule $\{\alpha_i\}_{i=0,1,2,...}$
Output: $NN^*$ that approximates the optimal Q-function
1  repeat at every trip $i$, i.e. iteration $i$
2      for each time step $k = 0, 1, 2, ...N$ do
3          observe $s_k = (T^k_{fd}, \omega^k_{fd}, l^k_r, t^k_r, \bar{v}^k_r)$
4          $a_k = \begin{cases} arg \min_{a \in A}\{NN^{(i-1)}(s_k, a)\} & \text{probability } 1 - \epsilon_i, \\ \text{a random action in } A & \text{probability } \epsilon_i. \end{cases}$
5          A contains all legal actions
6          apply $a_k$, record immediate cost $c_k$
7      end
8      at the end of trip, compute new target values:
9      $ctg_k = (1 - \alpha_i)NN^{(i-1)}(s_k, a_k) + \alpha_i[c_k + \min_a NN^{(i-1)}(s_{k+1}, a)]$
10     train NN with $\{(s_k, a_k), ctg_k\}_{k=1,2,...N} \Rightarrow NN^{(i)}$
11 until the trip cost has converged;
```

results in Fig. 6.2 were generated using exploration schedule = [0.60:-0.001:0.01], learning rate = [0.2:-0.00028:0.005], fuel price(u_f) = \$2.5/gallon, and electricity price(u_e) = \$0.12/KWH.

We extended the learning process to 1200 trips but found no further lower cost. Therefore, the converged optimal has been reached after 687 iterations. This suggests that for in-vehicle implementation, the Q-learning will converge after a certain amount of learning from a specific trip. The reason for the small cost oscillations after the minimum set point is because of a constant 1% control exploration used in Q-learning. We call the controller that yields the minimum cost of \$0.39 *VEC-UDDS*.

The 687 iterations for achieving the convergence are equivalent to 5040 miles/256 hours of driving. Our studies show that using analytical greedy control to initialize the NN can make the convergence time significantly shorter. However, the analytical greedy strategy requires a model. In this research, we focus on the model-free solution. If an accurate model can be

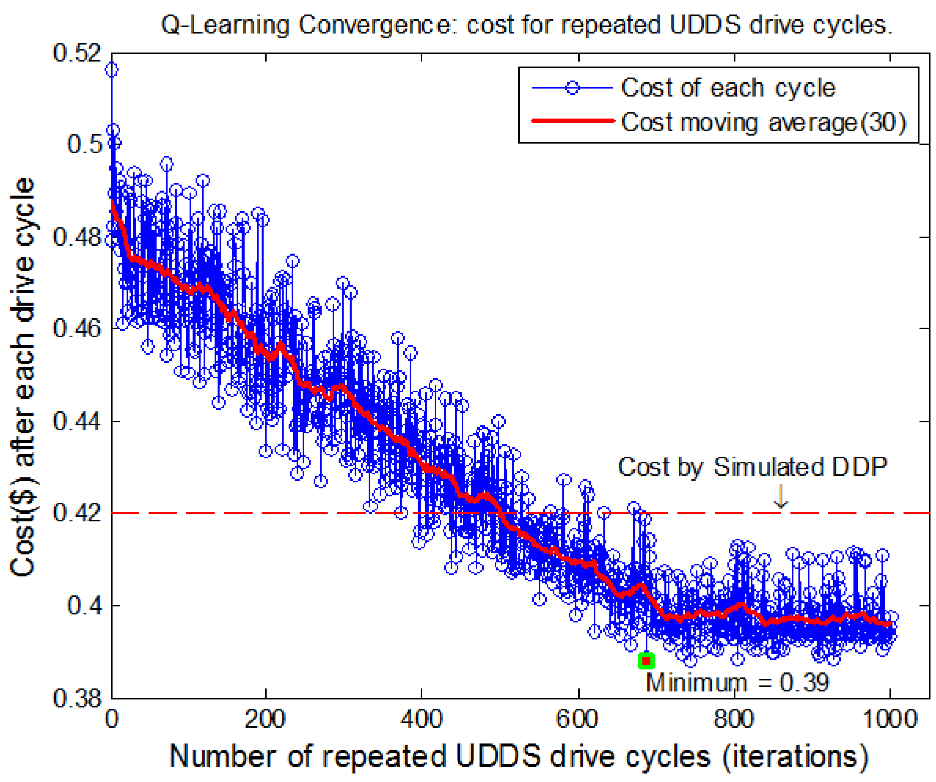

Figure 6.2: QL-ST convergence on repeated UDDS cycles. SoC_{init} is a randomly selected number $\in [70\%, 90\%]$.

developed for real-world applications, then the convergence time can be further optimized.

The theoretical optimal cost calculated using deterministic dynamic programming (DDP) for the UDDS drive cycle is about \$0.34. (Please refer to Appendix A for DDP description.) However, DDP not only requires detailed trip information beforehand but also relies on a simplified vehicle model. As a result, the theoretical minimum cannot be achieved in the real simulation. The control actions derived from the DDP were tested on the UDDS cycle, and they yielded a cost around \$0.42, which is higher than the converged Q-learning cost. This is marked by the horizontal red line in Fig. 6.2.

Fig. 6.3 compares the battery SoC profiles between the initial untrained controller and the converged optimal controller (VEC-UDDS). The optimal policy utilized more battery energy during the trip to minimize the total trip cost. It was able to recover more SoC with regenerative braking toward the end of the cycle.

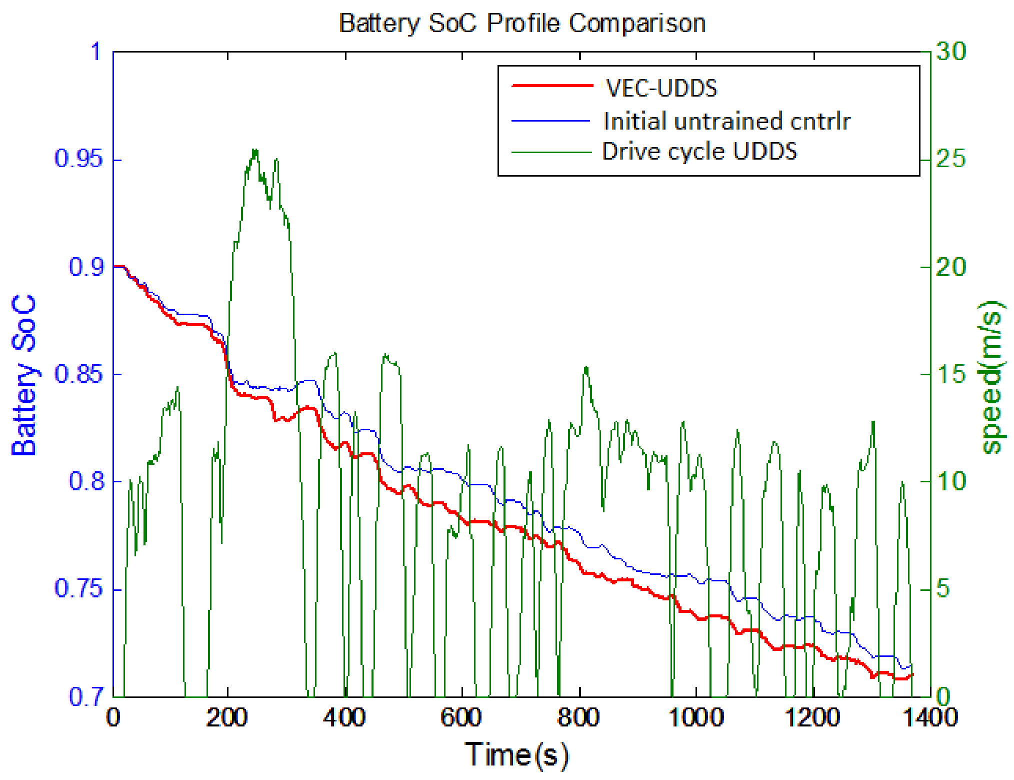

Figure 6.3: Battery SoC profile comparison between converged optimal control (VEC-UDDS) and initial untrained control on the UDDS drive cycle. $SoC_{init} = 90\%$.

Fig. 6.4 shows the recorded cost-to-go from the initial untrained controller and the converged optimal controller (VEC-UDDS). The cost-to-go at time 0 represents the total trip cost. The untrained controller (blue curve) and VEC-UDDS (red curve) yield a trip cost of \$0.47 and \$0.39, respectively. The cost-to-go curve is not monotonically decreasing because the regenerative braking causes the battery energy cost to be negative.

6.3 QL-ST on Home-Work Commutes

Another important characteristic of our Q-learning framework is that it has excellent robustness against noise in cycles. To demonstrate this, we added 20% white noise to each UDDS cycle. Fig. 6.5 shows the first 10 of 1000 UDDS cycles with white noise and the

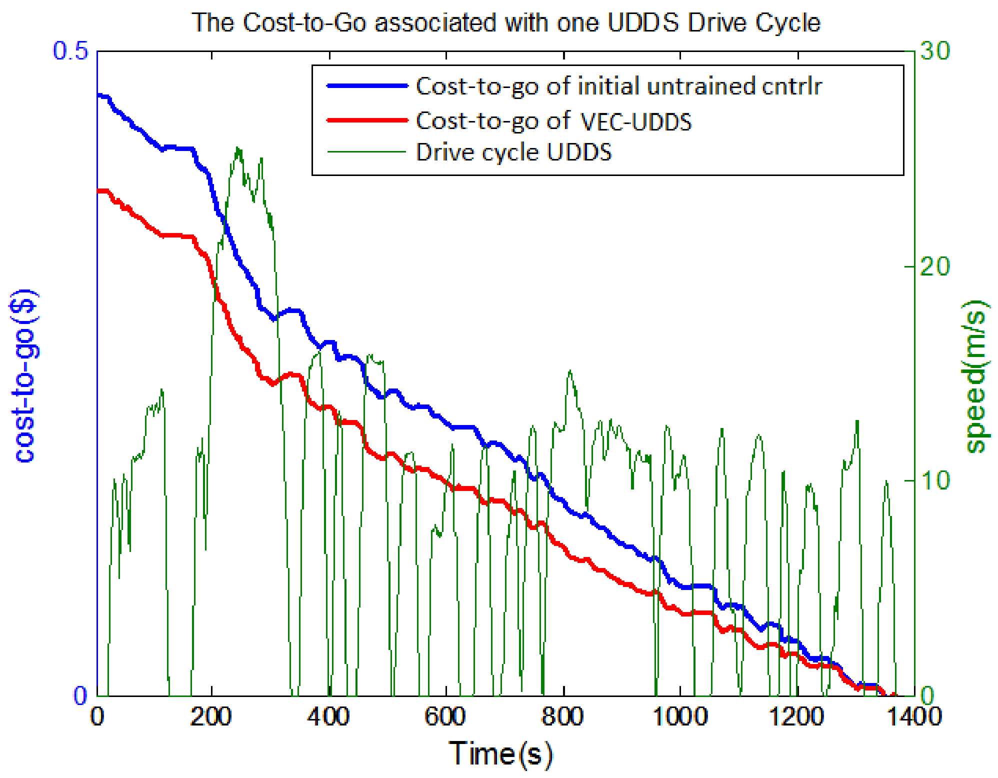

Figure 6.4: UDDS drive cycle and the recorded cost-to-go at each time step for the initial untrained controller and the converged optimal controller (VEC-UDDS).

convergence of the Q-learning. The learning has converged after about 750 iterations, a little bit more than in the case of the original UDDS cycles. The results were generated with exploration schedule = [0.60:-0.001:0.01], learning rate = [0.2:-0.00028:0.005], fuel price(u_f) = \$2.5/gallon, and electricity price(u_e) = \$0.12/KWH. Due to the added white noise, the cost continues to oscillate around the average level after 800 iterations. We name the controller trained after 750 UDDS noise cycles *VEC-UDDS-Noise750*. The robust convergence on the repeated short trips with noise makes Q-learning very suitable for learning optimal energy control for real world driving trips such as home/work commuting trips.

We also evaluated the optimal controller's performances on cycles with noise. We generated four new trips with different white noise levels from the standard UDDS. The max noise levels were limited to 10%, 20%, 30%, and 40%. Table 6.2 compares the performance of VEC-UDDS and VEC-UDDS-Noise750 with the default rule-based controller

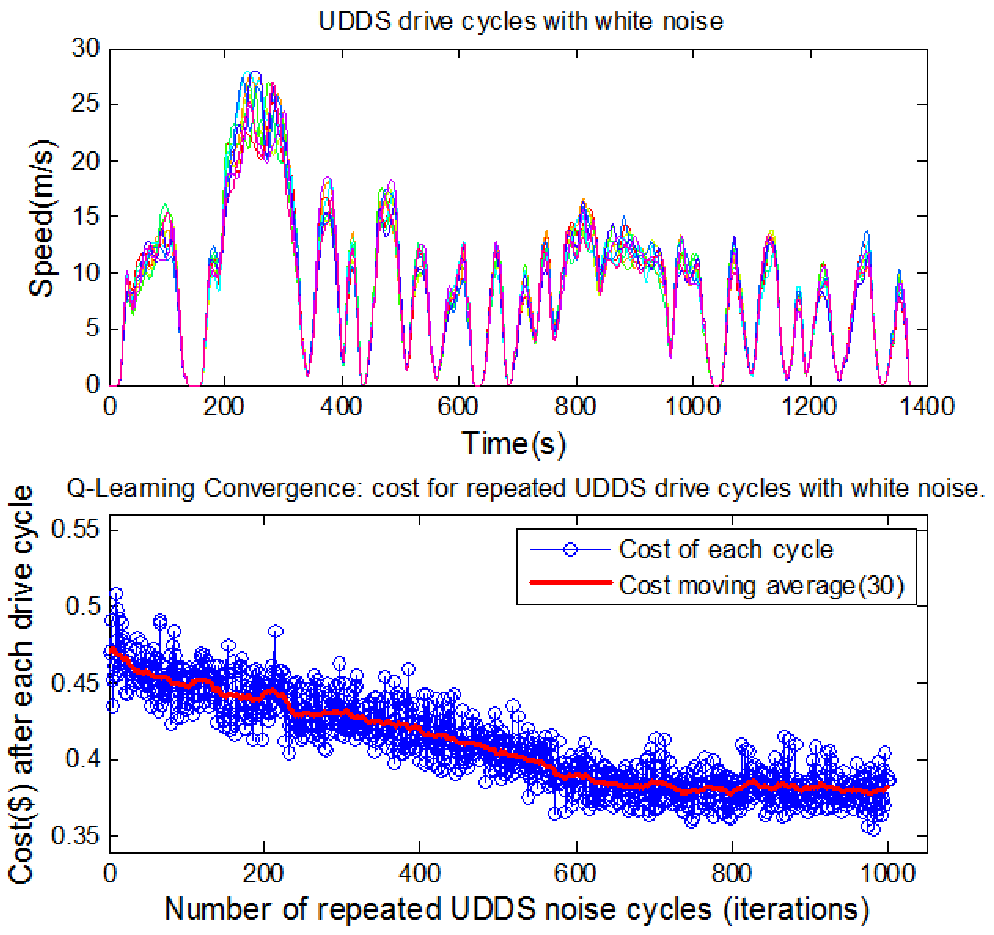

Figure 6.5: QL-ST converges well on the UDDS drive cycles with 20% white noise. SoC_{init} is a randomly selected number $\in [70\%, 90\%]$.

on the four noise cycles. The last column is called the *converged optimal*, which stores the costs obtained by running the Q-learning repeatedly on that specific cycle until it converges. So, in other words, the last column represents the *optimal cost* value for each drive cycle. It can be observed that controllers (VEC-UDDS and VEC-UDDS-Noise750) generated by the Q-learning achieved lower costs than the rule-based control. The controller VEC-UDDS-Noise750 trained on the noise cycles has even slightly better performance than VEC-UDDS trained on the original UDDS cycles.

Table 6.2: Trip Cost ($) Comparison on UDDS Cycles with Noise

$SoC_{init} = 90\%$, u_f=\$2.5/Gallon, u_e=\$0.12/KWH				
Drive Cycles	VEC- -UDDS	VEC-UDDS -Noise750	Rule-based	Converged Optimal
UDDS	0.39	0.40	0.41	0.39
UDDS+10%Noise	0.38	0.38	0.41	0.38
UDDS+20%Noise	0.40	0.39	0.47	0.39
UDDS+30%Noise	0.39	0.38	0.43	0.38
UDDS+40%Noise	0.45	0.44	0.58	0.44

6.4 Evaluation of QL-ST on Random Short Trips

Our next objective is to explore whether Q-learning can learn and converge on random drive cycles, because real world applications consist of random trips. For this purpose, we let QL-ST learn from 1200 random short trips. At each iteration, one trip was randomly selected from the following five cycles: UDDS, US06, NYCC, LA92, and NEDC. Table 6.3 counts the number of times each cycle was selected.

Table 6.3: Count of Drive Cycles Randomly Selected

UDDS	US06	NYCC	LA92	NEDC	**Total**
242	252	232	229	246	1200

Fig. 6.6 shows the first 20 of the 1200 trips. Each small circle is a recorded trip cost for that cycle after using the Q-function updated at the end of previous trip. The results were obtained with exploration schedule = [0.60:-0.001:0.01], learning rate = [0.2:-0.00028:0.005], fuel price(u_f) = \$2.5/gallon, and electricity price(u_e) = \$0.12/KWH. Every cycle type is marked by a different color. For example:

- The 1st trip is US06. The recorded cost is \$0.71. This trip is completed by using the NN initialized with the rule-based control.
- The 2nd trip is NEDC. The recorded cost, using the Q-function updated at the end of

the 1st trip, is \$0.47.

- The 3rd trip is UDDS. The recorded cost, using the Q-function updated at the end of the 2nd trip, is \$0.49, etc.

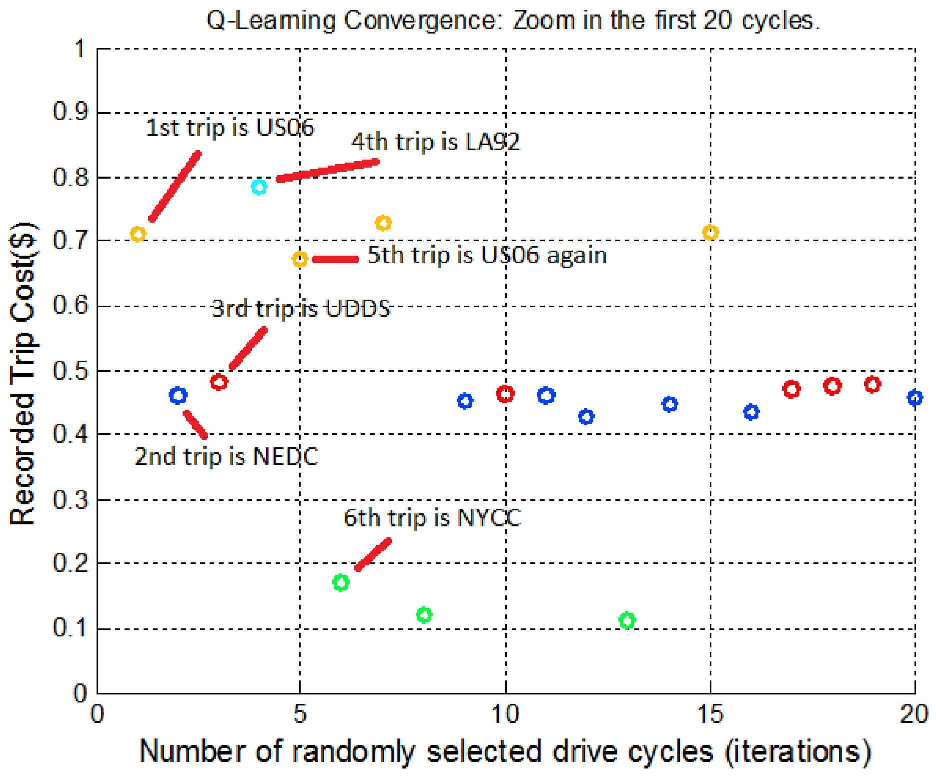

Figure 6.6: The recorded cost of the first 20 randomly selected trips. SoC_{init} is a randomly selected number $\in [70\%, 90\%]$.

The convergence of the trip energy cost on 1200 random trips is shown in Fig. 6.7. For clarity, we plotted only the cost trend, i.e., the moving average of the recorded cost with a window size of 30 iterations, for each type of cycle by coloring them differently. The cost values are normalized so that the convergence can be properly compared. The parameters associated with the learning simulations are exploration schedule = [0.60:-0.001:0.01], learning rate = [0.2:-0.00028:0.005], fuel price(u_f) = \$2.5/gallon, and electricity price (u_e) = \$0.12/KWH.

All the cycles reached their own minimum costs after about 850 iterations. From 900 to

1200, there is no further improvement. Compared with the convergence on fixed UDDS cycles, this result suggests that more variations in the drive cycles require more iterations to converge. We give the name *VEC-Rnd200*, *VEC-Rnd400*, *VEC-Rnd600*, *VEC-Rnd800*, and *VEC-ST* to the controllers that learned after 200, 400, 600, 800 and 1000 random trips through the procedure described above.

It is very interesting to observe that the convergence on the standard New York City drive cycle (NYCC) was most significant. On the other hand, the convergence on the highway cycle (US06) was least significant. This implies that Q-learning can do a better optimization on city trips with frequent start-stops.

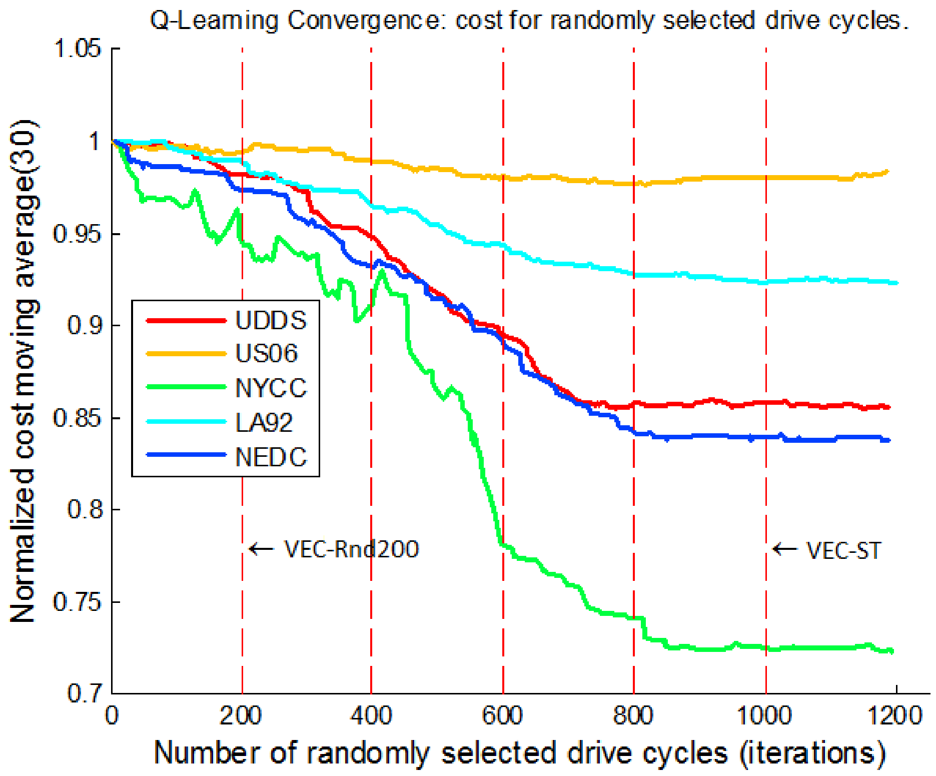

Figure 6.7: QL-ST convergence on randomly selected short trips. SoC_{init} is a randomly selected number $\in [70\%, 90\%]$.

The NN used to approximate the Q-function in VEC-ST is represented by a 3D surface in Fig. 6.8. The outputs of the NN are the expected cost-to-go, and they were plotted against the remaining travel time t_r and the remaining travel average speed $\bar{v}_r$. An interesting fact is that when the remaining trip time t_r is short, a small average speed increase (e.g. from 15m/s to 20m/s) can cause a sudden cost-to-go jump. As the t_r becomes longer, the cost-to-go smooths out to an almost linear function of the $\bar{v}_r$.

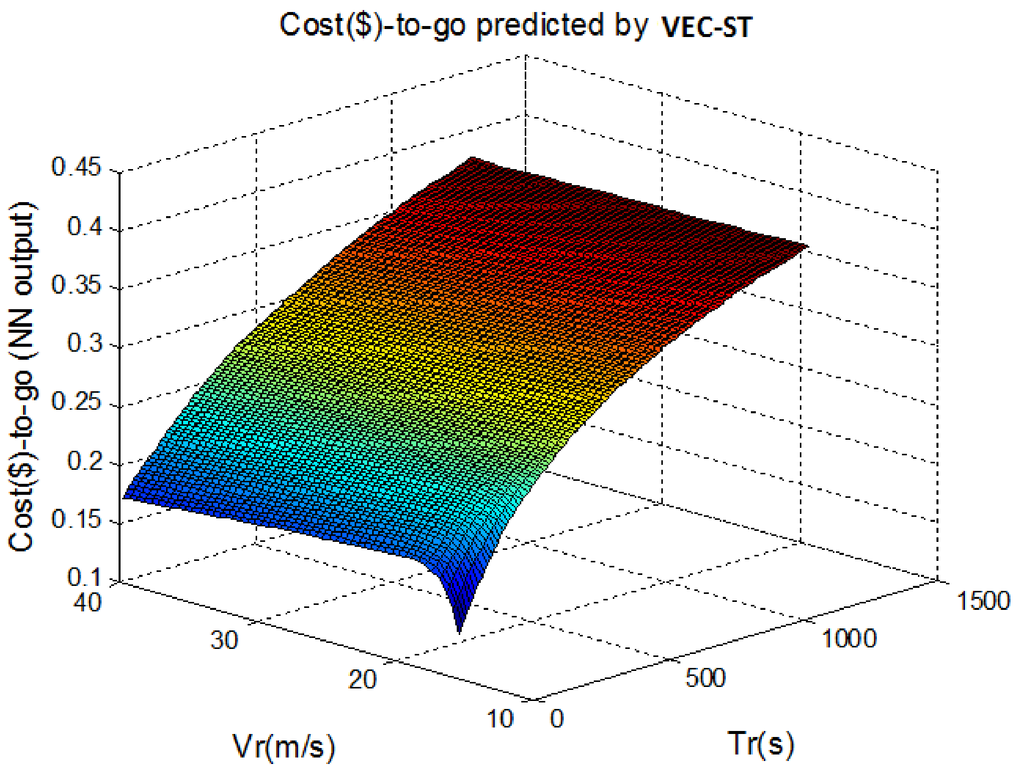

Figure 6.8: Plot of the NN outputs from VEC-ST. The NN is trained to generate the expected cost-to-go.

To validate the Q-learning process, we applied VEC-Rnd200, VEC-Rnd400, VEC-Rnd600, VEC-Rnd800, and VEC-ST to four test cycles chosen differently from the learning cycles. These four cycles are: HWFET, JAPAN1015, WLTC, and *Dearborn*. The Dearborn cycle is a real-world short driving trip collected in an urban area in southeast Michigan. This trip consists of 38 minutes/16 miles of combined highway-urban driving with 15 short traffic stops. Fig. 6.9 plots the energy cost by applying the above five Q-learning controllers to the four test cycles. The cost has also converged on each of those test cycles.

Table 6.4 summarizes the performances of the five vehicle energy controllers applied to the nine drive cycles. The *Converged Optimal* values in the last column were again obtained

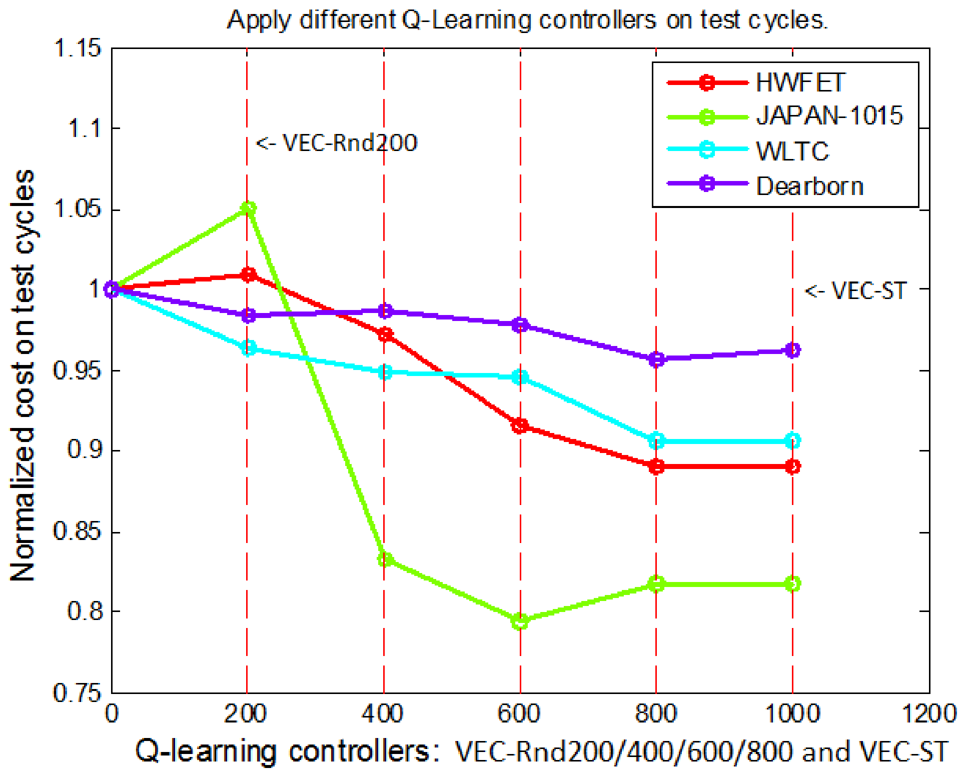

Figure 6.9: Trip costs obtained by applying VEC-Rnd200, VEC-Rnd400, VEC-Rnd600, VEC-Rnd800, and VEC-ST to four test cycles: HWFET, JAPAN-1015, WLTC and Dearborn.

by applying Q-learning repeatedly on that specific cycle until it converged. These numbers are considered as the upper bound of the performance on each drive cycle. The cost associated with the analytical greedy control was generated by applying the method described in Section IV on each drive cycle. The numbers shown in parentheses are the cost reduction in percentage from the rule-based control. VEC-UDDS, VEC-ST, and analytical greedy all showed cost savings compared with the default rule-based control. VEC-UDDS and VEC-ST demonstrated more significant cost savings for trips such as US06, NYCC, LA92, and WLTC. VEC-ST, the controller trained on random short trips, yields performances better than or equal to VEC-UDDS on all cycles other than UDDS, which is reasonable since VEC-UDDS was trained on UDDS drive cycles only.

For the HWFET cycle, the rule-based control achieved the lowest cost among all of the strategies. However, with some further investigation we found that the vehicle speed error (calculated as the accumulated speed errors in percentage between the target drive cycle and the real simulation output) generated by the rule-based controller was about 9%. In our study, all of the simulations using any Q-learning controller must have a speed error less than or equal to 5%. So in this case, the energy savings from the rule-based control was achieved by compromising vehicle driving performance, i.e. the energy control did not provide sufficient power to meet the driving speed demand.

Table 6.4: Trip Cost ($) Comparison on Different Drive Cycles

SoC_{init} is random $\in [70\%, 90\%]$, u_f=$2.5/Gallon, u_e=$0.12/KWH					
Drive Cycles	VEC-ST	VEC-UDDS	Analytical Greedy	Rule-based	Converged Optimal
UDDS	0.40(-2.4%)	0.39 (-4.9%)	0.40(-2.4%)	0.41	0.39(-4.9%)
US06	0.71(-14.5%)	0.71(-14.5%)	0.69(-16.9%)	0.83	0.68(-18.1%)
HWFET	0.58(+7.4%)	0.59(+9.3%)	0.58(+7.4%)	0.54	0.55(+1.8%)
NYCC	0.09(-10.0%)	0.09(-10.0%)	0.09(-10.0%)	0.10	0.09(-10.0%)
LA92	0.75(-9.6%)	0.77(-7.2%)	0.78(-6.0%)	0.83	0.73(-12.0%)
NEDC	0.38(-5.0%)	0.39(-2.5%)	0.38(-5.0%)	0.40	0.37(-7.5%)
JAPAN1015	0.13(0%)	0.13(0%)	0.13(0%)	0.13	0.13(0%)
WLTC	0.87(-8.4%)	0.88(-7.4%)	0.89(-6.3%)	0.95	0.86(-9.5%)
Dearborn	0.72(-4.0%)	0.72(-4.0%)	0.72(-4.0%)	0.75	0.72(-4.0%)
Overall	4.63(-6.3%)	4.67(-5.5%)	4.66(-5.7%)	4.94	4.52(-8.5%)

CHAPTER VII

Q-learning Algorithm QL-LT on Long Trips and Its Evaluation

In this section, we start with the introduction of a penalty function used for the cost-to-go estimation. This penalty term takes into account how far the current state's SoC is above the minimum threshold level.

7.1 Optimal Control Policy w.r.t the Penalty Function

With the additional penalty term, the optimal state-value function $V^*(s)$ was changed to a new optimal function $\hat{V}(s)$ defined in Eq. 7.1.

$$\hat{V}(s) = V^*(s) + W(s) = V^*(s) + W(SoC). \tag{7.1}$$

The new optimal cost-to-go function $\hat{V}(s)$ is the sum of two terms: $V^*(s)$ is the optimal cost-to-go following the state s without considering the SoC threshold; $W(SoC)$ represents the additional cost-to-go due to the SoC constraint. SoC in this case is the battery state-of-charge in s.

The optimal control action for the $\hat{V}(s)$ can be derived as:

$$\begin{aligned} a^* &= arg\min_a \big\{ c(s,a) + \hat{V}(s_{next}|s,a) \big\} \\ &= arg\min_a \big\{ c(s,a) + V^*(s_{next}|s,a) + W(s_{next}|s,a) \big\} \\ &= arg\min_a \big\{ c(s,a) + V^*_{next}(s) + W(SoC_{next}|s,a) \big\}, \end{aligned} \tag{7.2}$$

where SoC_{next} is the SoC of the next state from (s,a), so $W(SoC_{next}|s,a)$ is an action-dependent term. $V^*_{next}(s)$ is introduced in Eq. 4.2 and is not action dependent.

Recall that we have learned the $Q^*(s,a)$ on short trips. This Q-function estimates the optimal cost-to-go following a given state-action pair (s,a) without the SoC constraint. Thus, we can substitute the term $c(s,a) + V^*_{next}(s)$ with $Q^*(s,a)$ in Eq. 7.2. This gives us the new optimal control action:

$$a^* = \pi^*(s) = arg\min_a \big\{ Q^*(s,a) + W(SoC_{next}|s,a) \big\}. \tag{7.3}$$

Based on the above equations, the following observations can be made.

- We can directly apply the Q-function learned from short trips to generate optimal control actions a^* w.r.t. a specific penalty function. Our experiments in the next section will show that in most cases, by properly choosing the penalty function, this approach can already produce good results without the need for any further learning.
- The control policy presented in Eq. 7.3 is optimal only for that specific penalty function, so it is not guaranteed to be globally optimal. Later on, we will demonstrate that Eq. 7.3 can be used to setup the initial policy of QL-LT to significantly reduce the convergence time to the global optimal control, which is not dependent on the penalty function.
- In order to generate the control actions defined in Eq. 7.3, we need to predict SoC_{next} following a given state-action pair (s,a).

7.2 QL-LT Algorithm Description

In order to achieve optimal control independent of any penalty function, we still need to enable action exploration and let the algorithm learn and converge on long trips. However, in this case, we enforce a *battery power cutoff rule* to protect the battery state-of-health (SoH): when the SoC drops to the minimum threshold, we will not allow any battery output power. A free learning scheme such as proposed for short trips may cause the SoC to drop far below the minimum threshold, thus damaging the battery.

We add $(SoC - SoC_{thrd})$ as a new state variable to help in estimating the cost-to-go for those long trips subject to the SoC constraint. The Q-function is approximated by a new $\hat{NN}$ taking an extra input. Please note that there is no penalty function involved during the learning. We will show that the penalty function is used only to pre-train $\hat{NN}^{(0)}$. The new QL-LT framework is summarized in Table 7.1.

Table 7.1: Summary of QL-LT

State Variables (s):	T_{fd}, ω_{fd}, l_r, t_r, $\bar{v}_r$, $(SoC - SoC_{thrd})$
Control Action (a):	T_{eng}, ω_{eng}
$\hat{NN}$ Input:	State-Action pair (s, a)
$\hat{NN}$ Output:	Expected future cost-to-go for (s, a), i.e. $Q(s, a)$
Control Policy:	ϵ-greedy with battery power cutoff rule
Q-func. Update:	Temporal Difference with learning rate (NN weights updated at the end of each trip)

Algorithm QL-LT is given below for any trip subject to the SoC constraint. The NN training process and the overall learning iterations are the same as QL-ST. Please refer to Section VI for those details. However, comparing with QL-ST, QL-LT has two modifications: a new initialization strategy and a battery power cutoff rule. The details of the initialization method will be provided in the next subsection. The output of QL-LT is $\hat{NN}^*$, which approximates the optimal cost-to-go function. $\hat{NN}^*$ is used in the final controller VEC-LT.

Algorithm QL-LT

```
Input: $\hat{NN}^{(0)}$ trained with VEC-ST and a penalty function
        multiple trip data recorded $\{(s_k, a_k), c_k\}^{(i)}_{i=0,1,2,...}$
        exploration schedule $\{\epsilon_i\}_{i=0,1,2,...}$
        learning rate schedule $\{\alpha_i\}_{i=0,1,2,...}$
Output: $\hat{NN}^*$ that approximates the optimal Q-function
1  repeat at every trip $i$, i.e. iteration $i$
2      for each time step $k = 0, 1, 2, ...$ do
3          observe $s_k = (T^k_{fd}, \omega^k_{fd}, l^k_r, t^k_r, \bar{v}^k_r, SoC^k - SoC_{thrd})$
4          if $SoC^k > SoC_{thrd}$ then
5              $a_k = \begin{cases} arg\min_{a\in A}\{\hat{NN}^{(i-1)}(s_k, a)\} & \text{probability } 1-\epsilon_i, \\ \text{a random action in } A & \text{probability } \epsilon_i. \end{cases}$
6              $A$ contains all legal control actions
7          else
8              battery power cutoff ($P_{bo} \leq 0$): $a_k = arg\min_{a\in A'}\{\hat{NN}^{(i-1)}(s_k, a)\}$
9              $A'$ contains all legal control actions such that $P_{eng} \geq P_{fd}$
10         end
11         apply $a_k$, record immediate cost $c_k$
12     end
13     at the end of trip, compute new target values:
14     $ctg_k = (1-\alpha_i)\hat{NN}^{(i-1)}(s_k, a_k) + \alpha_i[c_k + \min_a \hat{NN}^{(i-1)}(s_{k+1}, a)]$
15     train $\hat{NN}$ with $\{(s_k, a_k), ctg_k\}_{k=1,2,...N} \Rightarrow \hat{NN}^{(i)}$
16 until the trip cost has converged;
```

7.3 Evaluation of QL-LT on Long Trips

We first showcase the long-trip algorithm QL-LT on the repeated UDDS cycles, but this time with the initial SoC set at 45% and the minimum threshold at the 30% level. A greedy control policy will cause the SoC to drop far below the 30% level. So, the UDDS cycle in this case becomes a *long trip* by our definition. The $\hat{NN}$ was initially trained with the trip data from a rule-based control, and the battery power cutoff rule was enforced during the entire learning process.

After 1033 iterations, the controller has converged to the minimum cost, which is demonstrated in Fig. 7.1. We denote this converged controller as *VEC-SoC4530-UDDS* in the later context. In order to reach an optimal control point, the learning rate should be sufficiently small. However if the learning rate is too small at the very beginning of the learning process, it may take a long time for the learning to converge to the optimal point. So our method is to use a relatively larger learning rate at the beginning and then reduce the learning rate after 1000 iterations. The results in Fig. 7.1 were obtained with exploration schedule = [0.80:-0.001:0.01], learning rate = [0.2:-0.00028:0.005], fuel price (u_f) = \$2.5/gallon, and electricity price (u_e) = \$0.12/KWH. Short trip learning took 687 iterations to converge on repeated UDDS cycles. With the SoC restriction, the number of iterations required to converge is much greater.

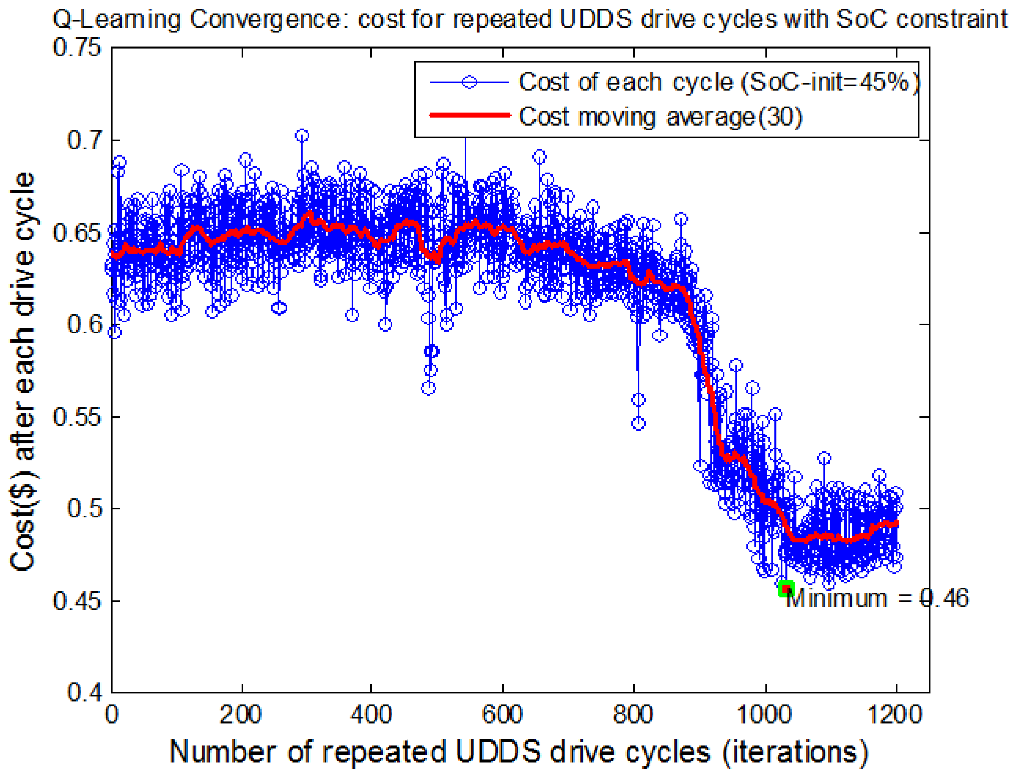

Figure 7.1: QL-LT applied to repeated UDDS cycles with $SoC_{init} = 45\%$, $SoC_{thrd} = 30\%$.

SoC profiles can provide insightful information about the control strategies. Fig. 7.2 compares the SoC generated by the initial untrained controller and by the converged one (VEC-SoC4530-UDDS). We reach the same conclusion that an aggressive charge depletion followed by charge sustaining is not optimal.

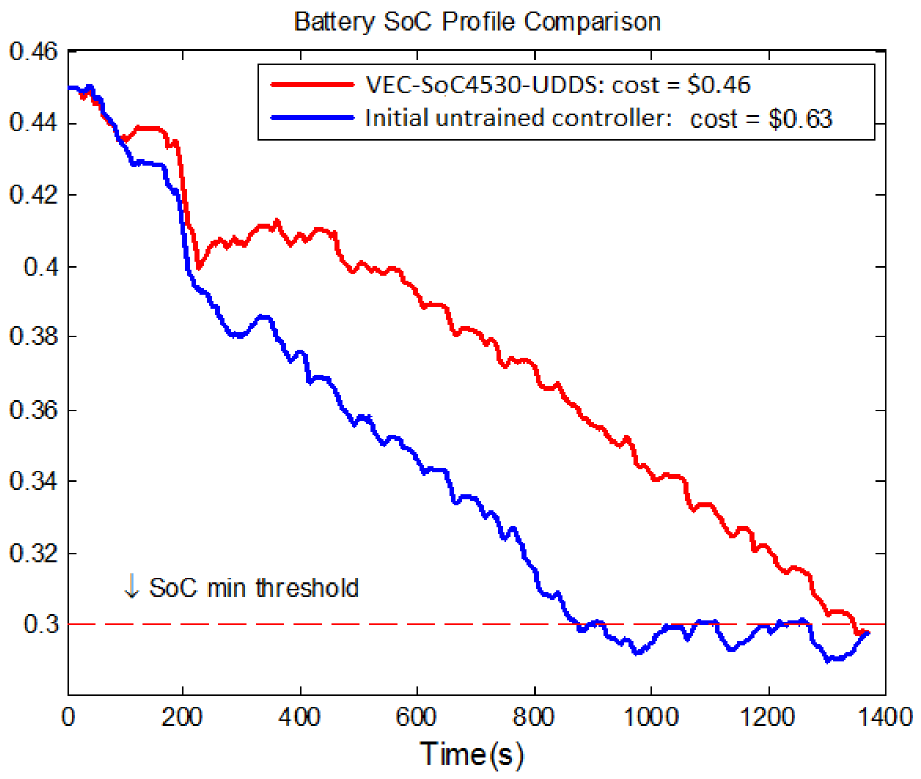

Figure 7.2: SoC profile comparison between the initial untrained controller and the converged optimal controller (VEC-SoC4530-UDDS) on the UDDS drive cycle. $SoC_{init} = 45\%$, $SoC_{thrd} = 30\%$.

The battery discharge curve produced by VEC-SoC4530-UDDS inspired us to use Eq. 7.3 to define a new initialization strategy. More specifically, we will combine the VEC-ST learned from short trips with a penalty function $W(SoC)$ that can gradually increase the penalty value as SoC gets close to its threshold. This penalty function will cause the battery to gradually deplete to its minimum level and produce an SoC profile similar to the optimal one generated by VEC-SoC4530-UDDS. The penalty function $W(SoC)$ for this new initialization purpose is defined in Eq. 7.4 and plotted in Fig.7.3.

$$W(SoC) = \beta e^{-\alpha(SoC - SoC_{thrd} - \Delta)}. \tag{7.4}$$

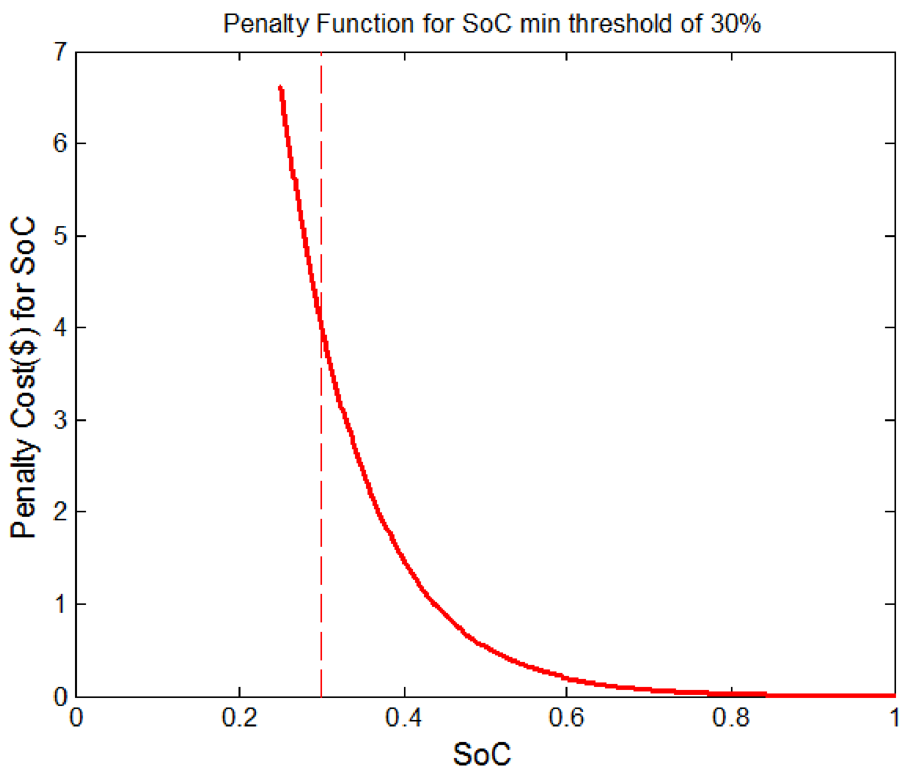

Figure 7.3: SoC penalty function defined in Eq. 7.4. $\alpha = 10$, $\beta = 0.2$, $\Delta = 0.3$

The new initialization strategy is carried out as follows: We use VEC-ST and $W(SoC)$ to generate control actions (defined in Eq. 7.3) for a long trip. We then collect the trip data to perform the initial training for $\hat{NN}$. This initialization is expected to make the controller start with close-to-optimal behavior. In order to use the policy defined in Eq. 7.3, we have developed another nerual network NN_{SoC}, to predict the SoC of the next step. The details about NN_{SoC} are given in the next subsection.

The QL-LT convergence was studied on random long drive cycles using the new initialization strategy. The set of cycles we randomly choose from is composed of four trips. Two trips, Mixed and Mixed2, were created by mixing some standard drive cycles and have a duration of over an hour. The third trip is a real-world trip driven from Dearborn, Michigan, to Ann Arbor, Michigan. This trip is 39.6 miles in distance with about 32 miles (33 minutes) on the I-94 freeway. The fourth trip is the standard 30-minute WLTC cycle composed of four speed phases, low, medium, high, and extra-high speed, and has a driving distance of 14.45 miles. The initial SoC for each iteration was set to 70%. Fig. 7.4 plots the speed profiles of these 4 cycles.

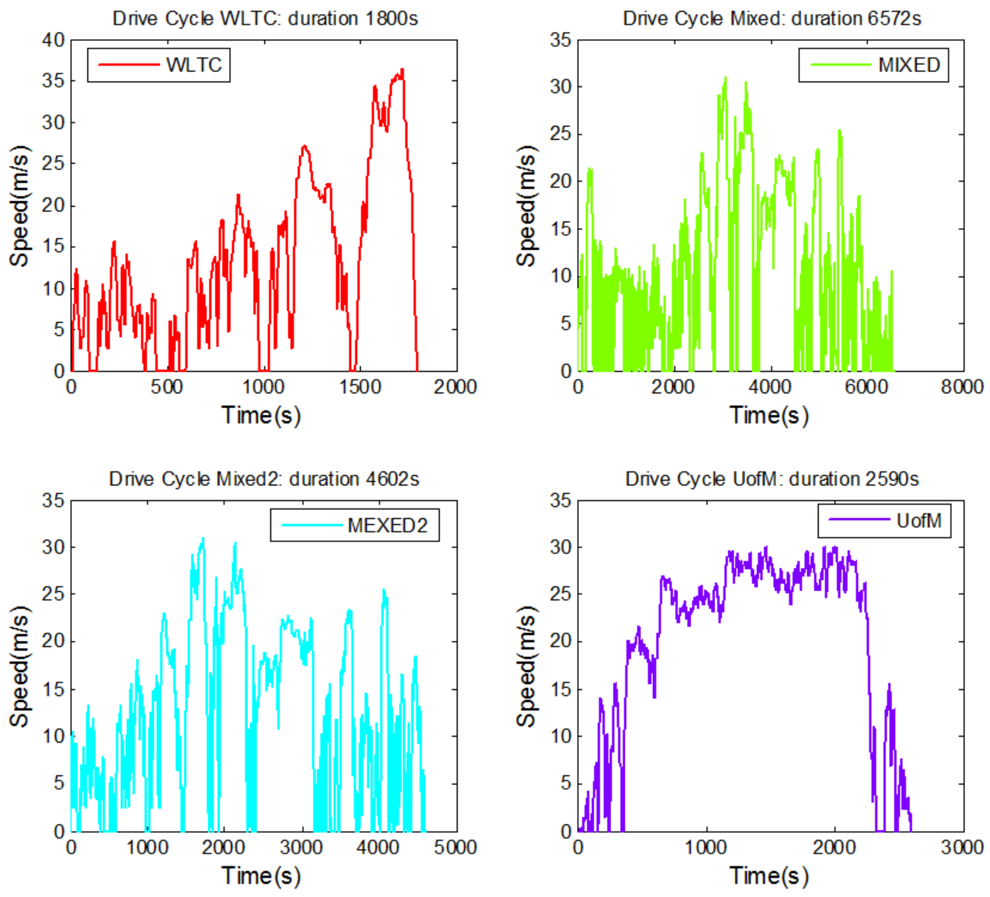

Figure 7.4: Four trips used to evaluate QL-LT algorithm. Two of them have durations of over one hour.

We expect our initial Q-function ($\hat{NN}^{(0)}$) to be close to optimal. Thus, a small exploration rate (10%) with a small learning rate (0.05) is used to kick off the learning. In our previous learning experiments, we had chosen the initial exploration rate to be 60% or 80% and set the learning rate to 0.2. At each QL-LT iteration, we apply the QL-LT algorithm to a drive cycle randomly selected from the four drive cycles shown in Fig. 7.4.

The trip cost moving averages (window size set to 30 iterations) of 600 simulated iterations are shown in Fig. 7.5. For a relatively short cycle such as WLTC, the controller out of the initialization is almost optimal. Additional learning beyond 150 cycles did not yield any better result. However, for three longer trips, the cost achieved the minimum after about 300 cycles, 70% quicker than the rule-based initialization (Fig. 7.1). We denote the controller trained after 300 iterations as *VEC-LT*. The small cost oscillations after 300 trips are due to the continuous action exploration. In this experiment, we used exploration schedule = [0.10:-0.0005:0.0005], learning rate = [0.05:-7.5E-5:0.005], fuel price (u_f) = \$2.5/gallon, and electricity unit price (u_e) = \$0.12/KWH. We extended the learning iterations to 1000 and there was no lower cost reached beyond the first 300 trips for all four types of cycles.

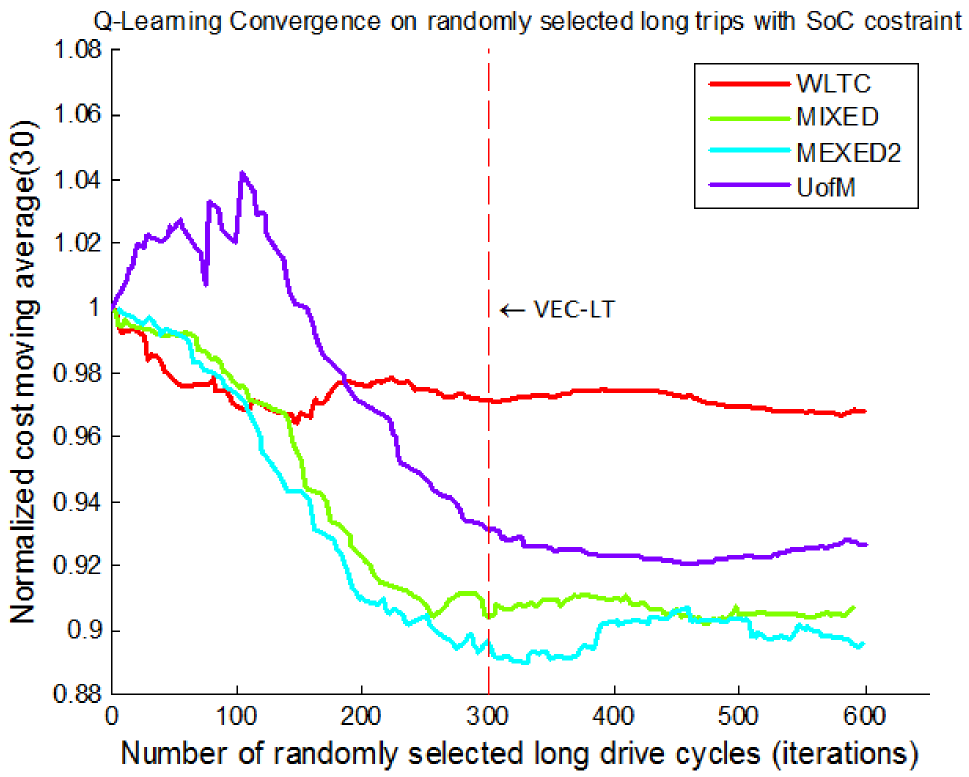

Figure 7.5: QL-LT convergence on randomly selected long trips subject to the SoC constraint. $SoC_{init} = 70\%$, $SoC_{thrd} = 30\%$.

For the purpose of comparison, the performance of various controllers within our study is summarized in Table 7.2. The column "VEC-ST+$W(\cdot)$" represents the trip costs obtained by directly applying VEC-ST learned from short trips with the penalty function $W(SoC)$ defined in Eq. 7.3, and no further learning is performed. "VEC-LT" is the controller after 300 iterations on random long cycles (Fig. 7.5). "VEC-SoC4530-UDDS" is the converged controller from the repeated UDDS drive cycles (Fig. 7.1). The 4th controller is the default rule-based one. The "*Converged Optimal*" values are the optimal costs achieved by applying Q-learning repeatedly on that specific drive cycle until it converged. These values are considered the upper bound of system performances.

Along with the trip cost, we also list the lowest SoC level SoC_{min} reached for a given controller and drive cycle. For shorter trip WLTC(14.5 mile), VEC-ST+$W(\cdot)$ yields higher costs than the rule-based control. This is expected, as the penalty function will penalize the battery usage as the SoC drops to its threshold. This approach will lead to a higher end-of-trip SoC and higher fuel cost. For longer trips, i.e., Mixed (43.4 mile), Mixed2 (36.1 mile), and UofM (32.4 mile), the penalty function is capable of producing close-to-optimal SoC profiles.

Similar behaviors can be observed for VEC-LT and VEC-SoC4530-UDDS. Both controllers outperformed the rule-based one and VEC-ST+$W(\cdot)$ for longer trips. In particular, VEC-LT has the best performance for Mixed, Miexed2, and UofM cycles among all the strategies. One note is that despite the battery power cutoff rule, the lowest SoC level of several simulations still touched below 30%. This is due to the fact that the battery/motor's power was used when the engine itself could not meet the propulsion demand.

Based on the results presented in Table 7.2, we can reach the conclusion that QL-LT trained on random cycles with the penalty function-based initialization yields the best performance for longer trips. However, this learning scheme is not suitable for short trips, to which we should apply algorithm QL-ST.

Table 7.2: Trip Cost ($) Comparison on Different Drive Cycles with SoC Constraint

$SoC_{init} = 70\%$, $SoC_{thrd} = 30\%$, u_f=$2.5/Gallon, u_e=$0.12/KWH										
	VEC-ST+W(·)		VEC-LT		VEC-SoC4530-UDDS		Rule-based		Converged Optimal	
	cost($)	SoC_{min}	cost($)	SoC_{min}	cost($)	SoC_{min}	cost($)	SoC_{min}	cost($)	SoC_{min}
WLTC	1.19(+12.3%)	31.6%	1.23(+16.0%)	46.6%	1.26(+18.9%)	38.3%	1.06	30.9%	1.06(0%)	31.5%
MIXED	7.84(-8.4%)	28.0%	7.18(-16.1%)	29.1%	8.15(-4.8%)	27.7%	8.56	26.4%	6.58(-23.1%)	30.2%
MIXED2	5.98(-1.5%)	28.8%	5.69(-6.3%)	27.6%	5.76(-5.1%)	27.6%	6.07	27.4%	5.45(-10.2%)	28.0%
UofM	3.85(-4.0%)	24.3%	3.49(-13.0%)	25.9%	3.50(-12.7%)	26.1%	4.01	22.6%	3.44(-14.2%)	26.2%
Overall	18.86(-4.3%)		17.59(-10.7%)		18.67(-5.2%)		19.7		16.53(-16.1%)	

7.4 SoC Prediction Using Neural Network

The new initialization strategy in QL-LT requires the application of the control actions defined in Eq. 7.3 with the prediction of SoC_{next}. In this section, we present two methods for predicting SoC_{next}: the first one is an analytical solution derived from the powertrain model; the second one is using a neural network, NN_{SoC}, which makes the prediction model-free. The performance of the two methods will be compared.

From Eq. 3.4 and Eq. 3.5, we have:

$$P_{bo} = \gamma_1 P_{fd} - \gamma_2 P_{eng} + \left(\frac{\gamma_2 - \gamma_1}{1+\rho}\right) T_{eng}\omega_{fd}. \tag{7.5}$$

Using the relationship between P_{bo} and P_{batt} defined in Eq. 4.6, we can estimate the battery power P_{batt} and therefore calculate the SoC_{next} given in Eq. 7.6.

$$SoC_{next} = SoC - \frac{P_{batt}\Delta t}{C_{batt}}. \tag{7.6}$$

To eliminate the use of an analytical model, we experimented with a multilayer perceptron network, NN_{SoC}, with H neurons in the hidden layer. The inputs to the NN_{SoC} are: power demand from the wheel, speed demand from the wheel, battery SoC, motor1/generator1 power, motor2/generator2 power, engine power, and engine torque. The output of the NN_{SoC} is the predicted SoC change (ΔSoC) for this time step.

Our controller collects training examples directly from the trip data and performs the training at the end of the trip. For real world applications, the training can be periodically scheduled according to the vehicle's life cycle. This strategy allows the prediction to adapt to the changing vehicle dynamics due to component wear and other usage factors.

Fig. 7.6 compares three SoC profiles on the Mixed2 cycle with the initial SoC set to 70%. The red curve was the result of the controller defined in Eq. 7.3 (VEC-ST+$W(\cdot)$), where SoC_{next} was predicted at each time step using the analytical Eq. 7.6. The green SoC curve represents the same controller except that SoC_{next} was estimated by NN_{SoC}. The blue curve was associated with the rule-based control. The similarity of the first two SoC curves can be easily observed: they both held a slower discharging profile than the rule-based control due to the SoC penalty term. The rule-based controller adopted the charge sustaining strategy around the 30% level and caused the SoC to drop to 27.4% at the peak power demand. The first two controllers also yield lower trip costs.

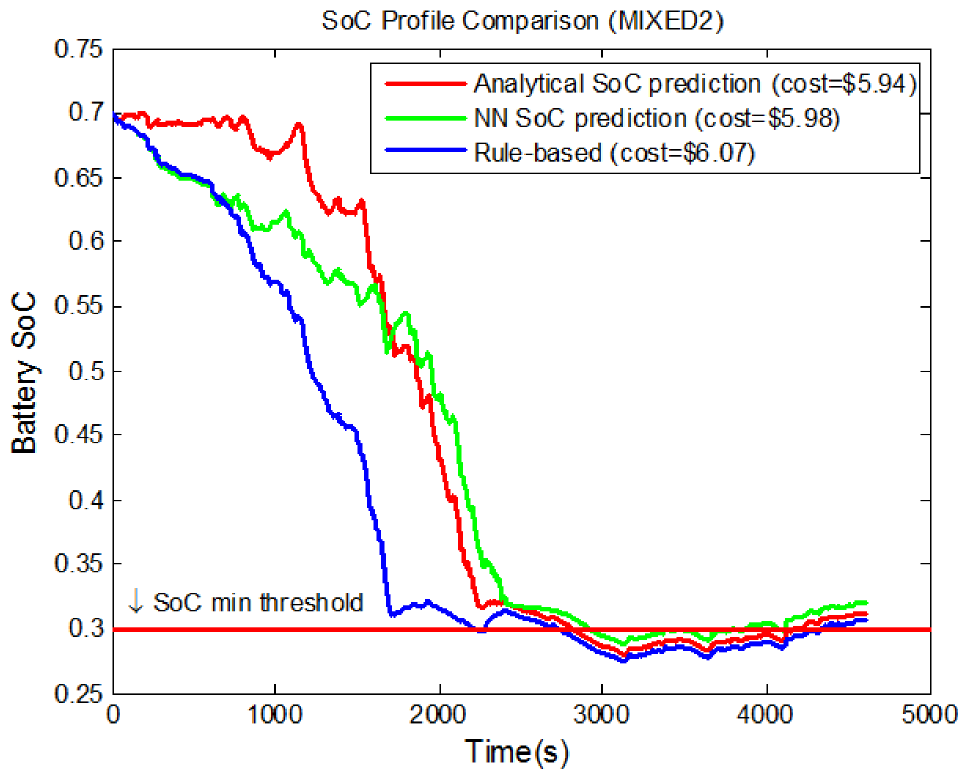

Figure 7.6: SoC profile comparison on Mixed2 drive cycle between the analytical SoC prediction, the NN-based (NN_{SoC}) prediction and Rule-based control. $SoC_{init} = 70\%$, $SoC_{thrd} = 30\%$

To further assess the performance of NN_{SoC}, we plot the histograms of SoC_{next} prediction errors shown in Fig. 7.7. The errors generated by the NN_{SoC} have a higher mean but a smaller standard deviation (mean=-1.61E-4, std=2.82E-4) compared with the analytical prediction (mean=-9.15E-5, std=3.38E-4). The overall performance of the two methods is comparable. The error means can be corrected by shifting the prediction with a corresponding offset.

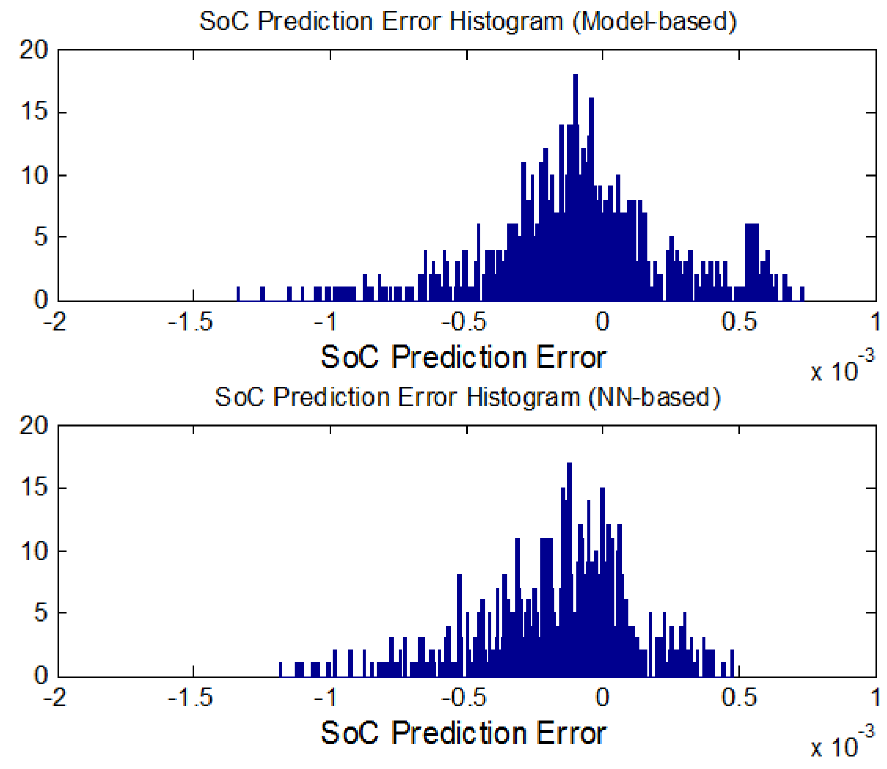

Figure 7.7: The histograms of the SoC prediction errors made by analytical method and NN_{SoC}.

Finding the optimal power management for long trips is a more challenging problem because it involves the battery minimum SoC constraint. A free learning scheme is no more appropriate as it may let the battery discharge below its threshold level, causing permanent damage. We derived a new learning algorithm, QL-LT, by enforcing a battery power cutoff rule. The difference between the current state's SoC and its minimum threshold was used as a new input to the Q-function to improve the cost-to-go estimation. Finally, to support our new initialization strategy, we have developed a neural network for the SoC prediction, which rendered our solution completely model-free.

CHAPTER VIII

Conclusion and Future Work

In this research work, we presented a machine learning approach that is model-free and capable of in-vehicle learning for minimizing PHEV energy cost. The approach consists of optimal Q-learning based algorithms, QL-ST and QL-LT, for optimizing energy cost for short and long trips, respectively. Q-learning is a powerful model-free reinforcement learning approach. The QL-ST learning algorithm is aimed at most of the home-work daily commuting trips, which usually can be driven solely with battery power, i.e., the energy optimization has no SoC constraint. We proved mathematically that a greedy policy is optimal for short trips without the SoC constraint and derived an analytical greedy control strategy from a steady-state powertrain model. The QL-LT algorithm is designed for long trip energy optimization, which incorporates SoC constraints.

Both algorithms are built upon Q-learning and neural dynamic programming (NDP). We demonstrated through extensive experiments that both the QL-ST and the QL-LT algorithms give superior performance over other control methods, are inherently robust against cycle noises, and have the capability of converging to the optimal policy on both fixed and random trips. With an innovative initialization strategy, QL-LT is capable of reducing the convergence time by about 70%.

Another significant new contribution of this work is a new learning paradigm proposed for deployment of the two learning algorithms, namely QL-ST and QL-LT, for in-vehicle applications. In this paradigm, in order to speed up in-vehicle learning convergence, the two learning algorithms are first applied to pre-selected trips on engineering development vehicles to obtain the energy controllers that converge. These energy controllers are then directly applied to production vehicles and continue to be improved based on knowledge learned from new completed trips. For in-vehicle learning, it is recommended that a very small action exploration be used. In-vehicle learning allows the controller to adapt to each individual vehicle and driver style.

Finally we would like to point out that the proposed machine learning algorithms can be easily applied to cost functions that incorporate an emission control component. Our future work will be to incorporate the effect of emissions into the optimal learning algorithms. Subsequently, we would like to explore Q-learning applications in battery performance and smart grid optimization. As automotive engineering moves to fully electrified solutions, we believe that Q-learning technologies can play an important role in solving future challenging vehicle control problems.

APPENDIX

APPENDIX A

Deterministic Dynamic Programming by Forward Induction For PHEV Energy Optimization

Deterministic dynamic programming (DDP) is an approach for resolving a complex optimization problem by dividing it into a sequence of stages, where each stage poses a simpler problem. Essentially, at each stage, we deal with a much simpler optimization problem, and its solution relies only on the information from the current stage, and is not dependent on the history data.

DDP has become a commonly used technique to obtain the theoretical optimal results for a vehicle energy optimization problem. In order to find the optimal energy usage, the algorithm requires knowledge such as the wheel torque and wheel speed demands for the entire trip. It also relies on a simplified vehicle model.

In the literature, DDP by backward induction [13, 19–21, 21, 36, 37, 39] has been widely used. In this algorithm, an end-of-trip SoC is chosen and the algorithm works backward to find the minimum energy consumption paths from a set of starting SoCs. Although this setup works well for conventional HEVs, where the battery works mostly in a charge-sustaining mode, it does not fit the real-world PHEV applications because the combined fuel and battery energy optimization for PHEVs imposes only one condition, which is the lowest SoC level allowed, i.e., SoC minimum threshold (SoC_{thrd}). There is no constraint on the end-of-trip SoC as long as it is above the minimum threshold level.

We found that DDP by forward induction is a better alternative for the PHEV energy optimization problem. The forward induction algorithm is mathematically equivalent to the backward induction. However, the forward induction algorithm starts with a beginning SoC and works forward to find the optimal energy paths for a set of end-of-trip SoCs. Thus, the minimum SoC constraint can be easily incorporated into the algorithm.

The DDP by forward induction algorithm is able to provide all of the optimal paths toward different end-of-trip SoCs by one pass. We consider the set that includes all of the paths with the lowest SoC level above SoC_{thrd}. The global optimal solution subject to the SoC_{thrd} constraint is simply the path with the minimum energy consumption in this set. On the other hand, DDP by backward induction would require running the algorithm through multiple passes, where each pass is for a fixed end-of-trip SoC.

Fig. A.1 below illustrates the DDP by forward induction algorithm. An exemplary optimal energy usage path has been indicated by the red arrows.

The standard UDDS drive cycle used in our simulations has a duration of 1369 seconds. So we setup 1370 (i.e. N=1370) stages, where each stage represents a one-second time-step. As shown in Fig. A.1, the SoC is the state variable, and it has been discretized into M levels representing M states. Also, we have the wheel torque and wheel speed demands known at each stage. They are the a-priori trip knowledge, which imposes the conditions on the stage transitions.

In Fig. A.1, C is the matrix to record the minimum energy cost from starting to any (state, stage) point. For example, $C(i, n)$ is the minimum energy cost from the starting point s_0 to (state-i, stage-n). We have $C(i, 0) = 0$ for all $i = 1, 2, ...M$.

Assuming that we know $C(i, n-1)$ for all i in stage-(n-1), we can determine the $C(i, n)$

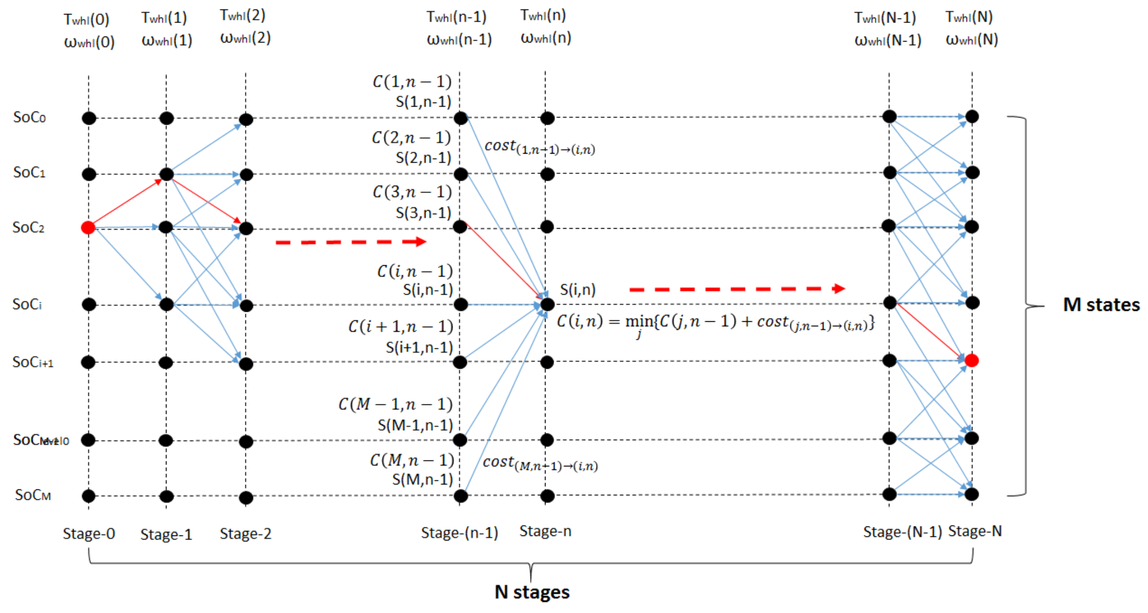

Figure A.1: Dynamic Programming for PHEV Energy Optimization

for each state-i in stage-n:

$$C(i,n) = \min_{j}\left\{C(j,n-1) + cost_{(j,n-1)\to(i,n)}\right\}, \tag{A.1}$$

where $cost_{(j,n-1)\to(i,n)}$ is the minimum energy cost associated with the transition from (state-j, stage-(n-1)) to (state-i, stage-n). It is worth noting that $cost_{(j,n-1)\to(i,n)}$ has to be determined with respect to the torque and speed demands and the powertrain Eq. 3.4.

The transition from $(j, n-1)$ to (i, n) imposes a constraint on the ΔSoC, which means that the electrical energy cost for this step transition is fixed. Therefore, the powertrain system represented by Eq. 3.4 becomes a system with one degree of freedom. In our DDP algorithm, we choose the best engine torque value T_{eng} to find the minimum step cost, "$cost_{(j,n-1)\to(i,n)}$". The methodology is the same as for the analytical solution presented in Chapter IV, section 4.2.

The optimal path derived from DDP by forward induction allows us to find the best control actions at each stage, i.e., $a_i^* = \{P_{batt}^*, T_{eng}^*\}_{i=1,2,...N}$. We deployed these control actions into the UDDS drive cycle simulation. However, the true trip energy cost from the simulation is \$0.42, 23.5% higher than the theoretical DDP result (\$0.34). Therefore, we conclude that the Q-learning algorithms proposed in this research are the better solutions for finding the real achievable optimal energy cost.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Wikipedia. "Wikipedia Web Source". `http://en.wikipedia.org/wiki/Plug-in_hybrid/`, 2017. [Online; accessed 10-September-2017].

[2] F. R. Salmasi. "Control Strategies for Hybrid Electric Vehicles: Evolution, Classication, Comparison, and Future Trends". *IEEE Transactions on Vehicular Technology*, 56(5), September 2007.

[3] Y. Gurkaynak, A. Khaligh, and A. Emadi. "State of the Art Power Management Algorithms for Hybrid Electric Vehicles". In *Proceedings of Vehicle Power Propulsion Conference*, pages 388–394, September 2009.

[4] B. Ganji and A. Z. Kouzani. "A Study on Look-ahead Control and Energy Management Strategies in Hybrid Electric Vehicles". In *2010 8th IEEE International Conference on Control and Automation*, June 2010.

[5] A. A. Malikopoulos. "Supervisory Power Management Control Algorithms for Hybrid Electric Vehicles: A Survey". *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–17, March 2014.

[6] Q. Gong, Y. Li, and Z. Peng. "Trip Based Optimal Power Management of Plug-in Hybrid Electric Vehicles". *IEEE Transactions on Vehicular Technology*, 57(6), November 2008.

[7] Q. Gong, Y. Li, and Z. Peng. "Computationally Efficient Optimal Power Management for Plug-in Hybrid Electric Vehicles Based on Spatial-Domain Two-Scale Dynamic Programming". In *Proceedings of the 2008 IEEE International Conference on Vehicular Electronics and Safety*, September 2008.

[8] M. Zhang, Y. Yang, and C. Mi. "Analytical Approach for the Power Management of Blended-Mode Plug-In Hybrid Electric Vehicles". *IEEE Transactions on Vehicular Technology*, 61(4), May 2012.

[9] N. Jalil, N. A. Kheir, and M. Salman. "A Rule-Based Energy Management Strategy for a Series Hybrid Vehicle". In *Proceedings of the American Control Conference*, June 1997.

[10] V. H. Johnson, K. B. Wipke, and D. J. Rausen. "HEV Control Strategy for Realtime Optimization of Fuel Economy and Emissions". *SAE 2000-01-1543*, 2000.

[11] N. J. Schouten, M. A. Salman, and N. A. Kheir. “Fuzzy Logic Control for Parallel Hybrid Vehicles”. *IEEE Transactions on Control Systems Technology*, 10(3):460–468, May 2002.

[12] A. Kahrobaeian, B. Asaei, and R. Amiri. “Comparative Investigation of Charge-Sustaining and Fuzzy Logic Control Strategies in Parallel Hybrid Electric Vehicles”. In *IEEE Vehicle Power and Propulsion Conference, 2009. (VPPC 2009)*, pages 1632–1636, September 2009.

[13] Z. Chen and C. Mi. “An Adaptive Online Energy Management Controller for Power-split HEV Based on Dynamic Programming and Fuzzy Logic”. In *IEEE Vehicle Power and Propulsion Conference (VPPC 2009)*, September 2009.

[14] S. G. Li, S. M. Sharkh, F. C. Walsh, and C. N. Zhang. “Energy and Battery Management of a Plug-In Series Hybrid Electric Vehicle Using Fuzzy Logic”. *IEEE Transactions on Vehicular Technology*, 60(8), October 2011.

[15] B. Zhang, M. Zhang, and C. Mi. “Charge-Depleting Control Strategies and Fuel Optimization of Blended-Mode Plug-in Hybrid Electric Vehicles”. *IEEE Transactions on Vehicular Technology*, 60(4):1516–1525, May 2011.

[16] G. Paganelli, S. Delprat, T. M. Guerra, J. Rimaux, and J. J. Santin. “Equivalent consumption minimization strategy for parallel hybrid powertrains”. In *IEEE 55th Vehicular Technology Conference (VTC 2002)*, May 2002.

[17] A. Sciarretta, M. Back, and L. Guzzella. “Optimal Control of Parallel Hybrid Electric Vehicles”. *IEEE Transactions on Control Systems Technology*, 12(3), May 2004.

[18] C. Musardo, G. Rizzoni, and B. Staccia. “A-ECMS: An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management”. In *Proceedings of the 44th IEEE Conference on Decision and Control*, December 2005.

[19] C. Lin, H. Peng, J. W. Grizzle, and J. Kang. “Power Management Strategy for a Parallel Hybrid Electric Truck”. *IEEE Transactions on Control Systems Technology*, 11(6), November 2003.

[20] D. Kum, H. Peng, and N. K. Bucknor. “Optimal Energy and Catalyst Temperature Management of Plug-in Hybrid Electric Vehicles for Minimum Fuel Consumption and Tail-Pipe Emissions”. *IEEE Transactions on Control Systems Technology*, 21(1), January 2013.

[21] R. M. Patil, Z. Filipi, and H. K. Fathy. “Comparison of Supervisory Control Strategies for Series Plug-In Hybrid Electric Vehicle Powertrains Through Dynamic Programming”. *IEEE Transactions on Control Systems Technology*, 22(2), March 2014.

[22] C. Lin, H. Peng, and J. Grizzle. “A Stochastic Control Strategy for Hybrid Electric Vehicles”. In *Proceedings of American Control Conference*, 2004.

[23] L. Johannesson, M. Asbogard, and B. Egardt. "Assessing the Potential of Predictive Control for Hybrid Vehicle Powertrains Using Stochastic Dynamic Programming". *IEEE Transactions on Intelligent Transportation Systems*, 8(1), March 2007.

[24] E. D. Tate Jr, J. W. Grizzle, and H. Peng. "Shortest path stochastic control for hybrid electric vehicles". *International Journal of Robust and Nonlinear Control*, 18:1409–1429, December 2008.

[25] J. Liu and H. Peng. "Modeling and Control of a Power-Split Hybrid Vehicle". *IEEE Transactions on Control Systems Technology*, 16(6), November 2008.

[26] S. J. Moura, H. K. Fathy, D. S. Callaway, and J. L. Stein. "A Stochastic Optimal Control Approach for Power Management in Plug-In Hybrid Electric Vehicles". *IEEE Transactions on Control Systems Technology*, 19(3), May 2011.

[27] D. P. Bertsekas and J. N. Tsitsiklis. "Decision and Control, Proceedings of the 34th IEEE Conference on". In *Proceedings of the 2008 IEEE International Conference on Vehicular Electronics and Safety*, December 1995.

[28] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge MA, 1998.

[29] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* John Wiley and Sons, Hoboken, New Jersey, 2011.

[30] G. Tesauro. "Temporal Difference Learning and TD-Gammon". *Communications of the ACM (Association for Computing Machinery)*, 38(3), March 1995.

[31] L. Johannesson and B. Egardt. "Approximate Dynamic Programming Applied to Parallel Hybrid Powertrains". In *Proceedings of the 17th World Congress on the International Federation of Automatic Control*, July 2008.

[32] R. Johri and Z. Filipi. "Self-Learning Neural Controller for Hybrid Power Management Using Neuro-Dynamic Programming". *SAE 2011-24-0081*, 2011.

[33] 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". In *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, 2014.

[34] S. Mohan, Y. Kim, and A. G. Stefanopoulou. "Estimating the Power Capability of Li-ion Batteries Using Informationally Partitioned Estimators". *IEEE Transactions on Control Systems Technology*, 24(5), September 2016.

[35] R. Hsu, S. Chen, W. Chen, and C. Liu. "A Reinforcement Learning Based Dynamic Power Management for Fuel Cell Hybrid Electric Vehicle". In *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems and 2016 17th International Symposium on Advanced Intelligent Systems*, 2016.

[36] J. Park, Z. Chen, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Y. L. Murphey. “Intelligent Vehicle Power Control Based on Machine Learning of Optimal Control Parameters and Prediction of Road Type and Trafc Congestion”. *IEEE Transactions on Vehicular Technology*, 58(9), November 2009.

[37] Y. L. Murphey, J.Park, Z. Chen, M. L. Kuang, M. A. Masrur, and A. M. Phillips. “Intelligent Hybrid Vehicle Power ControlPart I: Machine Learning of Optimal Vehicle Power”. *IEEE Transactions on Vehicular Technology*, 61(8), October 2012.

[38] Y. L. Murphey, J. Park, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Q. Wang. “Intelligent Hybrid Vehicle Power ControlPart II: Online Intelligent Energy Management”. *IEEE Transactions on Vehicular Technology*, 62(1), January 2013.

[39] Z. Chen, C. Mi, J. Xu, X. Gong, and C. You. “Energy Management for a Power-Split Plug-in Hybrid Electric Vehicle Based on Dynamic Programming and Neural Networks”. *IEEE Transactions on Vehicular Technology*, 63(4), May 2014.

[40] C. Zhang, A. Vahidi, P. Pisu, X. Li, and K. Tennant. “Role of Terrain Preview in Energy Management of Hybrid Electric Vehicles”. *IEEE Transactions on Vehicular Technology*, 59(3), March 2010.

[41] H. Yu, M. Kuang, and R. McGee. “Trip-Oriented Energy Management Control Strategy for Plug-In Hybrid Electric Vehicles”. *IEEE Transactions on Control Systems Technology*, 22(4), July 2014.

[42] Wikipedia. “Wikipedia Web Source”. `http://en.wikipedia.org/wiki/Model_predictive_control`, 2014. [Online; accessed 18-July-2014].

[43] S. Kermanil, S. Delprat, T.M. Guerra, and R. Trigui. “Predictive Control for HEV Energy Management : Experimental Results”. In *IEEE Vehicle Power and Propulsion Conference (VPPC 2009)*, September 2009.

[44] H. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, I. V. Kolmanovsky, and S. Di Cairano. “MPC-Based Energy Management of a Power-Split Hybrid Electric Vehicle”. *IEEE Transactions on Control Systems Technology*, 20(3), May 2012.

[45] X. Hu, S. J. Moura, N. Murgovski, B. Egardt, and D. Cao. “Integrated Optimization of Battery Sizing, Charging, and Power Management in Plug-In Hybrid Electric Vehicles”. *IEEE Transactions on Control Systems Technology*, 24(3), May 2016.

[46] H. Nafisi, S. M. Agah, H. A. Abyaneh, and M. Abedi. “Two-Stage Optimization Method for Energy Loss Minimization in Microgrid Based on Smart Power Management Scheme of PHEVs”. *IEEE Transactions on Smart Grid*, 7(3), May 2016.

[47] C. J. C. H. Watkins and P. Dayan. “Q-Learning”. *Machine Learning*, 8(3):279–292, May 1992.

[48] R. Bellman. “A Markovian Decision Process”. *Journal of Mathematics and Mechanics*, 6:1–11, April 1957.

[49] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010.

[50] Argonne National Laboratory. “Autonomie (Version 13.0), Computer Software”. `http://www.autonomie.net/`, 2013.

[51] M. A. Rapino and A. K. Fields. “Mega Commuting in the U.S., 2006-2010”. Technical report, U.S. Census Bureau, 2010.

[52] U.S. Census Bureau. “American Community Survey Results 2015”. Survey, The United States Census Bureau, 2015.

[53] B. McKenzie. “Who Drives to Work? Commuting by Automobile in the United States: 2013”. Technical report, U.S. Department of Commerce, Economics and Statistics Administration, U.S. Census Bureau, 2015.