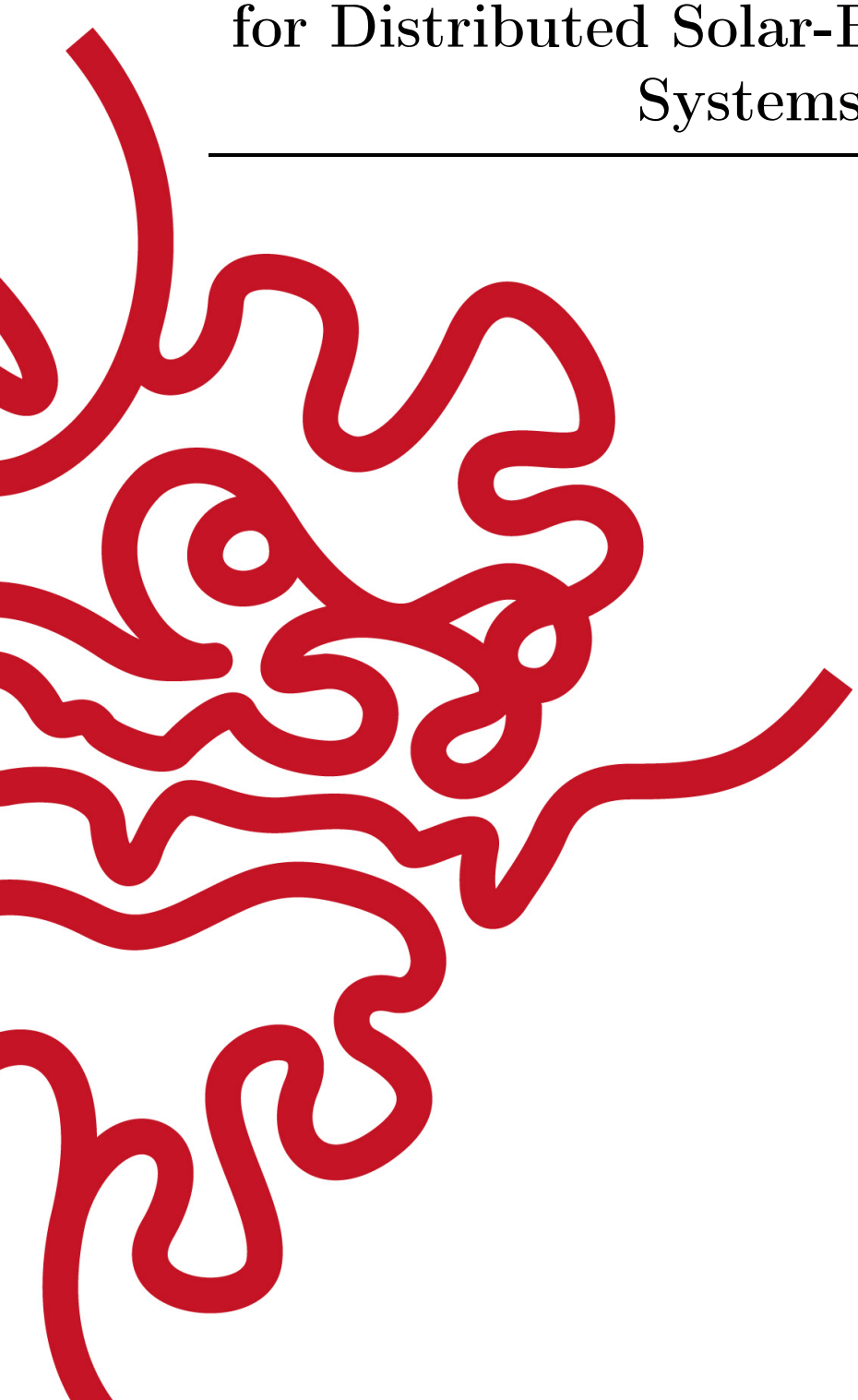
Okinawa Institute of Science and Technology
Graduate University

Thesis submitted for the degree

Doctor of Philosophy

Multi-Agent Reinforcement Learning for Distributed Solar-Battery Energy Systems



by

Qiong Huang

Supervisor: **Kenji Doya**

Jan, 2023

Declaration of Original and Sole Authorship

I, Qiong Huang, declare that this thesis entitled *Multi-Agent Reinforcement Learning for Distributed Solar-Battery Energy Systems* and the data presented in it are original and my own work.

I confirm that:

- No part of this work has previously been submitted for a degree at this or any other university.
- References to the work of others have been clearly acknowledged. Quotations from the work of others have been clearly indicated, and attributed to them.
- In cases where others have contributed to part of this work, such contribution has been clearly acknowledged and distinguished from my own work.
- None of this work has been previously published elsewhere, with the exception of the following:

  Q. Huang, E. Uchibe, and K. Doya. Emergence of communication among reinforcement learning agents under coordination environment. In *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 57-58. IEEE, 2016

  Q.Huang, K. Doya. An experimental study of emergence of communication of reinforcement learning agents. In *International Conference on Artificial General Intelligence*, pages 91-100. Springer, 2019.

Date: Jan, 2023
Signature: 

Abstract

Efficient utilization of renewable energy sources, such as solar energy, is crucial for achieving sustainable development goals. As solar energy production varies in time and space depending on weather conditions, how to combine it with distributed energy storage and exchange systems with intelligent control is an important research issue. In this thesis, I explore the use of reinforcement learning (RL) for adaptive control of energy storage in local batteries and energy sharing through energy grids. I first test multiple RL algorithms for energy storage control of single houses. I then extend the Autonomous Power Interchange System (APIS) from SONY to combine it with reinforcement learning algorithms in each house. I consider different design decisions in applying RL: whether to use centralized or distributed control, at what level of detail actions should be learned, what information is used by each agent, and how much information is shared across agents. Based on these considerations, I implemented deep Q-network (DQN) and prioritized DQN to set the parameters of real-time energy exchange protocol of APIS and tested it using the actual data collected from OIST DC-based Open Energy System (DCOES). The simulation results showed that DQN agents outperform rule-based control on energy sharing and that prioritized experience replay further improves the performance of DQN. Simulation results also suggest that sharing average energy production, storage and usage within the community helps the performance. The results contribute to future designs of distributed intelligent agents and effective operations of energy grid systems.

Acknowledgment

First, I would like to express my deep gratitude to my supervisor Prof. Kenji Doya for his invaluable patience, feedback and guidance, enthusiastic encouragement, and useful critiques of this research work. He accepted me into his unit even though I come from a different background. I also changed my research project, which was a huge challenge, but he provided me the opportunity to restart a new project and supervised me throughout the study. I have gained plenty of knowledge under his guidance that benefits me in critical thinking.

I also would like to thank my academic mentor Prof. Gail Tripp, who generously provided her knowledge and expertise. I am grateful that she can be my mentor to support my research study. Her encouragement along the way, guidelines throughout this energy management project, and continuous help in making decisions for my study plan take me to this point.

Also, I am grateful to my thesis committee member Prof. Hiroaki Kitano. With the previous DCOES project carried out in his unit as a benchmark, I can have the chance to test reinforcement learning approaches in a practical application system. And he proposed a lot of valuable questions for helping to clarify the scientific questions I can propose with the project.

I am also grateful to Mr. Kenichiro Arakaki and Mr. Daisuke Kawamoto. Mr. Arakaki provided all the previous historical data from the DCOES project and gave me a lot of practical explanations and discussions on the dataset and how the system works. Mr. Kawamoto from Sony Computer Science Laboratories, Inc. (Sony CSL) has a lot of critical discussions on the APIS simulators. Without their generous help, I would take much longer time to figure out how the system functions.

Additionally, this endeavor would not have been possible without the generous support from the OIST, Graduate School, and Neural Computational unit. Not only did they finance my research, but also they provided me with consistent help. I also want to thank my former group leader, Dr. Eiji Uchibe, for his patience, generous support, and late-night feedback sessions. I would also like to extend my thanks to all other lab members and Mr. Xianjie Zhang for their help in scientific discussions.

Lastly, I want to give special thanks to my family, especially my parents and my siblings. Their belief in me has kept my spirits and motivation high during this process. I would also like to thank my little nephew for all the entertainment and emotional support. I would like to finally thank Dr. Weiwei Qi for mailing me the hometown rice noodles which accompanied me through my study time.

Abbreviations

APIS	Autonomous Power Interchange System
BMU	Battery Management Unit
DCOES	Direct Current Open Energy System
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EMU	Energy Management Unit
ESS	Energy Storage System
GDPG	Gibbs Deep Policy Gradient
LSTM	Long-Short Term Memory
MDP	Markov Decision Process
MG	Microgrid
OES	Open Energy System
OIST	Okinawa Institute of Science and Technology
POMDP	Partial Observable Markov Decision Process
PV	Photovoltaic
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RSOC	Relative State of Charge
SSR	Self Sufficient Rate
TD	Temporal Difference

Nomenclature

γ	Discount factor
α	Learning rate
$\mathcal{A}$	Action space
$\mathcal{S}$	State space
$\mathcal{R}$	Reward function
$\mathcal{T}$	State transition probability density function
π	Policy in reinforcement learning theory
π^*	Optimal policy in reinforcement learning theory
V	State value function in reinforcement learning
V^*	Optimal state value function in reinforcement learning
Q^*	Optimal state-action value function in reinforcement learning
Q^π	State-action value function in reinforcement learning
a_t	Action taken at the step t
s_t	State at the step t
r_t	Reward received at the step t
K_c	Charing coefficient of the battery
K_d	Discharing coefficient of the battery

To my beloved parents, who have encouraged me attentively.

Contents

Declaration of Original and Sole Authorship ii

Abstract iii

Acknowledgment iv

Abbreviations v

Nomenclature vi

Contents viii

List of Figures xi

List of Tables xiii

1 Introduction 1

1.1 The DC-based Open Energy System (DCOES) at OIST 4

1.1.1 Different approaches for optimization of energy grids 7

1.1.2 Bid for markets 7

1.1.3 Machine learning method for reducing surplus energy 7

1.2 Energy grid systems and machine learning applications 8

1.2.1 Energy management approaches 8

1.2.2 Energy management by reinforcement learning 9

1.3 Thesis outline and contributions 11

2 Theoretical Background and System Description 13

2.1 Reinforcement learning 13

2.1.1 Markov decision process (MDP) 14

2.1.2 Partially observable Markov decision process (POMDP) 16

2.1.3 Different methods in RL 17

2.2 Deep reinforcement learning 18

2.3 Multi-agent reinforcement learning 20

2.4 Energy management of the DCOES 21

2.4.1 Battery charge/discharge control by RSOC 23

2.4.2 The DCOES data preprocessing 24

2.5 Autonomous Power Interchange System (APIS) . . . 26
2.5.1 Energy exchange based on scenario files . . . 27
2.5.2 APIS data flow . . . 28
2.6 Summary . . . 29

3 Single House Energy Management with Reinforcement Learning 30
3.1 PV panel and Battery Settings . . . 30
3.2 Model of battery . . . 31
3.2.1 Linear simulation model of the battery . . . 31
3.3 Single house RL with the tabular method . . . 32
3.3.1 State and Action representation . . . 33
3.3.2 Tabular Q learning . . . 34
3.4 DQN with Prioritized experience replay . . . 37
3.4.1 Algorithms . . . 37
3.4.2 States representation in DRL . . . 37
3.4.3 Simulation results . . . 38
3.5 Summary . . . 42

4 Multiple House Energy Management with Reinforcement Learning 45
4.1 Reinforcement learning setup for the APIS . . . 45
4.2 Action and state representations . . . 46
4.3 Reward and evaluation criteria . . . 47
4.4 DCOES dataset in OIST . . . 48
4.5 Choice of action time step and reward settings . . . 49
4.6 Comparison of different DRL methods . . . 49
4.7 Multiple iterations and runs . . . 52
4.8 Generalization across houses and seasons . . . 56
4.8.1 Shuffled houses ID . . . 56
4.8.2 Testing with different time periods of the year . . . 57
4.9 Summary . . . 58

5 Conclusion and Open Issues 60
5.1 Conclusion . . . 60
5.2 Open issues . . . 61

Bibliography 63

Appendices 69

A Historical raw data 69

B Scenario file 71

C Formated input data 73

D Selection of acceleration **74**

E Simulation Result Data Sample **78**

List of Figures

1.1 Global energy consumption from 1965 to 2020 (in EJs). 1
1.2 Worldwide energy sector data from BP p.l.c. [4]. 2
1.3 Global installed PV power from 1996 to 2020 (in MW). Source: statistics are taken from national statistical agencies, international organizations, and other proprietary sources. Includes data from International Renewable Energy Agency (IRENA 2021, Abu Dhabi), BNEF, IHS. 3
1.4 Different architectures of power transmission type [2]. 4
1.5 Topologies for renewable energy system, Werth et al. [66]. 5
1.6 Structure of installation in DCOES. 6
1.7 Landscape of different routes with all 19 houses in the community. . . . 6
1.8 A general microgrid system model with the integration of generations, storage, and loads components. 8

2.1 Single agent-environment interaction framework in reinforcement learning. 14
2.2 Return function. 16
2.3 Deep neural network structure diagram for the RL agent in MG. 20
2.4 A multi-agent environment interaction framework in reinforcement learning. 21
2.5 Overall architecture of the OIST DCOES. 22
2.6 Sketch of the house energy storage system. 22
2.7 The organization the Energy Storage Server (ESS) for the DCOES. . . 23
2.8 The battery operation mode control based on the RSOC. 24
2.9 Visualization of valid data of House 214 over every 30s on Jan 1, 2019. 25
2.10 Real-time battery monitoring of each residence 26
2.11 Request/Accept threshold in scenarios. 28
2.12 Data flow in APIS. 28

3.1 Relationship between battery current and change in RSOC, quarter-hour data; house 214, 2019. 31
3.2 Linear model simulation prediction in different houses. 32
3.3 Sketch of ESS data flow. 33
3.4 Discretized values of states in different bins. 35
3.5 Q learning tabular methods, mean rewards of house 214 in 2019. . . . 36
3.6 Q learning tabular methods, mean rewards of house 215 in 2019. . . . 37
3.7 Mean reward curve of House 214 in different years. 40
3.9 Average reward of house 214, with time cycle features, 2019. 41

3.10 Total reward of house 214, multiple iterations, 2019. 41
3.11 Performance in different cases. 42
3.8 Average reward of different houses in route B. 44

4.1 Statistics of PV production and consumption in different datasets. . . . 48
4.2 Results with different timestep under different conditions; the reward is set to the sum of purchased power. 50
4.3 Average values in different criteria with different reward settings, DQN. 51
4.4 Actions and scenarios for houses in different cases. 52
4.5 Performance of different states options in different methods. 53
4.6 Average values in different criteria with different iterations, prioritized DQN. 54
4.7 Cumulative total reward from all houses with regard to multiple iterations in different cases. 55
4.8 Average values in different criteria in different criteria, prioritized DQN, runs=3 iter=3. 56
4.9 Performance of testing data, training data (iter=5), prioritized DQN. . 58

D.1 Simulated data when gl.acc = 10 with sample data. 75
D.2 Simulated data when gl.acc = 30 with sample data. 76
D.3 Simulated data when gl.acc = 60 with sample data. 77

List of Tables

1.1 Smart Grid domains in conceptual model (Dileep [11]). 3

3.1 PV panel and Battery Settings for route B houses in 2018 and 2019 . . 30

4.1 Mean, Standard deviation, T-test results in shuffled ID. 56

A.1 Raw data for one day from the weather station. 69
A.2 Raw data for one day from one house. 70

C.1 Solar data for 4 houses in Jul, 2019. 73
C.2 Load data for 4 houses in Jul, 2019. 73

E.1 Individual data. 78
E.2 Summary data. 79

Chapter 1

Introduction

We live in a time of expanding energy demand due to the growth of the global population with booming economic activities. According to BP's Statistical review of world energy 2021, the global energy consumption increased 371.79% over the past 55 years, from 155.22 EJ[1] in 1965 to 577.10 EJ in 2020 (Figure 1.1) [4]. As fossil fuels dominate the world's primary energy, their mining, combustion consumption, and other aspects are extensive and cause concerning environmental impacts. Destruction of land and impact on water resources are the most typical environmental impacts of coal mining. And coal combustion produces sulfur dioxide and nitrogen dioxide, which contribute to air pollution. Moreover, fossil fuels release various gases and solid wastes during combustion and waste heat during power generation. Using fossil fuels not only produces a large amount of carbon dioxide, which intensifies the greenhouse effect, jeopardizes the global climate, and imbalances the ecological balance but also bring about thermal pollution. The "waste heat" left over from power generation by thermal power plants is discharged into rivers, lakes, the atmosphere, or oceans, causing thermal pollution in most cases.

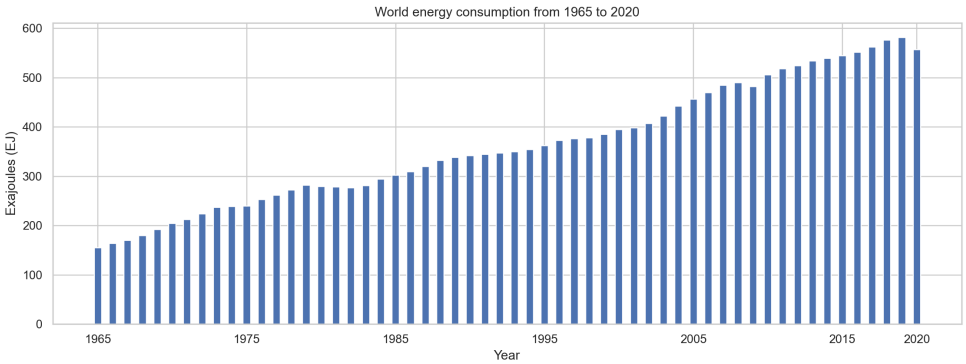


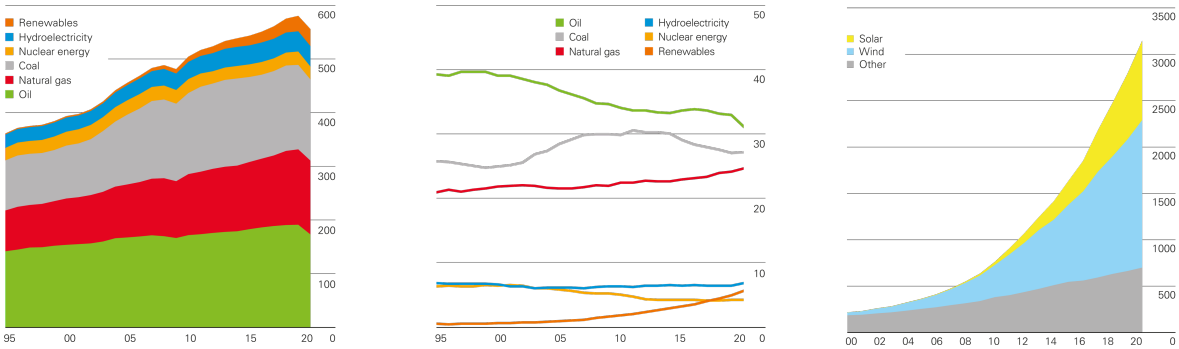
Figure 1.1: Global energy consumption from 1965 to 2020 (in EJs).

So far, the fuels used in countries around the world are almost all fossil fuels, namely oil, natural gas, and coal. Fossil fuels that have been gradually formed in nature over millions of years may all be consumed by humans within a few hundred years. Accord-

[1]EJ: energy symbol for exajoule, 1 EJ = 10^{18} J.

ing to observations and studies, there is no coal and oil forming underground today. In the 20th and 21st centuries, when stepping into the pace of global modernization, fossil fuels have a potential energy shortage crisis, especially gasoline extracted from oil, which is one of the reasons for the global oil crisis.

Currently, many countries are actively developing renewable and nuclear energy sources, which can help reducing reliance on fossil fuels, effectively delaying the consumption speed of non-renewable energy and reasonably regulating the utilization rate of resources. In the recent BP review report [4], as shown in Figure 1.2a, due to the COVID-19 pandemic, world energy consumption in 2020 fell drastically compared with 2019 since the imposition of lockdowns around the world decimated transport-related demand. However, generation from renewable energies, such as solar, wind, hydropower, biofuels, etc., continued to grow and reached its largest-ever increase. In addition, the share of renewable energy rose to record highs of 5.7% and has overtaken nuclear (4.3% of the share) in Figure 1.2b. Figure 1.2c shows the generation of the renewable by source in Terawatt-hours (TWh) from 2000 to 2020. We can observe world solar power generation has continued to grow over the past decade and accounts for 27% of total renewable energy generation.



(a) World consumption with different sources in EJ, 1995-2020.

(b) Shares of global primary energy in percentage (%), 1995-2020.

(c) Global renewable generation by source in TWh, 2000-2020.

Figure 1.2: Worldwide energy sector data from BP p.l.c. [4].

The collected statistical data indicates the trend of strong growth in renewable generation in the power sector, albeit renewable energy accounts for only a small portion (3.2%) of total power generation. In actual applications, we need to have more significant progress on energy efficiency. For instance, solar energy relies highly on weather conditions. For solar energy data alone, we could notice that its statistics worldwide capacity (installed photovoltaic (PV) power) expanded exponentially in the past 25 years (Figure 1.3).

Therefore, optimizing the allocation and using the power more efficiently and intelligently has gotten significant attention and interest recently. An intelligent energy grid, or so-called Smart Grid, has been proposed in using new communication techniques and information to better utilize electrical power [3, 5, 24, 25, 27, 38, 39, 41, 44]. Although various parties, including the European technology platform, U.S. department of energy, International Electrotechnical Commission (IEC), Institute of Electrical and

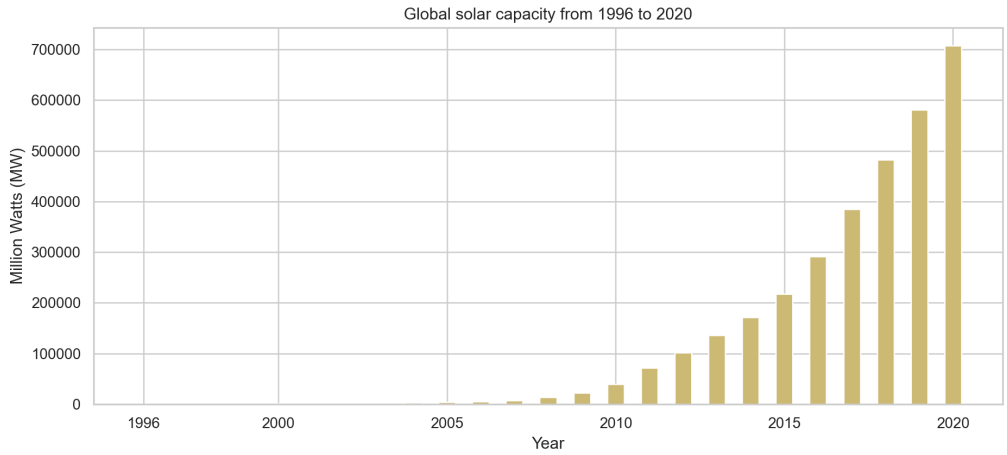


Figure 1.3: Global installed PV power from 1996 to 2020 (in MW). Source: statistics are taken from national statistical agencies, international organizations, and other proprietary sources. Includes data from International Renewable Energy Agency (IRENA 2021, Abu Dhabi), BNEF, IHS.

Electronics Engineers (IEEE), etc., provide different definitions of Smart Grids, the message is clear that efficiency should be highly concerned as survey Dileep [11] noted. In this survey, Dileep concluded the domains covered in the Smart Grid from consumer, markets, utilities, operations, generation, transmission, and distribution perspectives in Table 1.1. Each sector is interconnected with other domains in forming the Smart Grid model. Various grids and systems could seek solutions from these domains.

Table 1.1: Smart Grid domains in conceptual model (Dileep [11]).

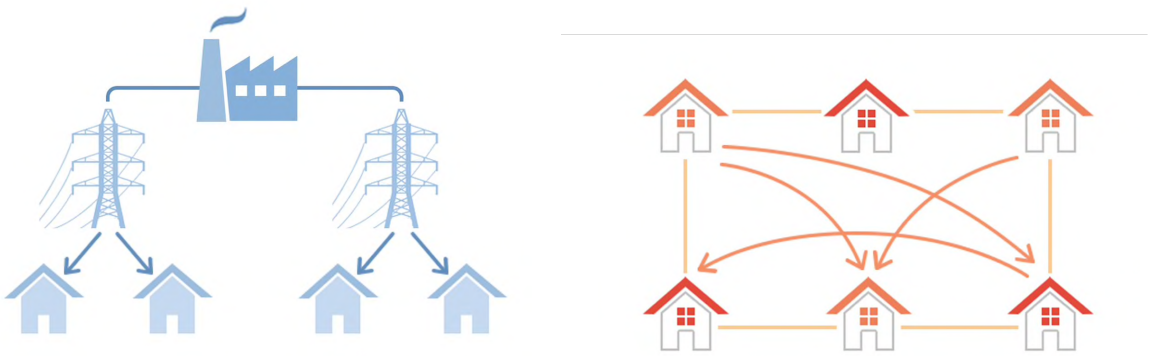
Domain	Actors in the domain
Consumer	End users of electricity, may also generate, store and manage the energy usage
Markets	The participants and operators exchange
Utilities	The organization that provides service to the consumer
Operations	The managers in movement of electricity
Bulk generation	The bulk quantity generator of electricity, can be also stored for future use
Transmission	The transporter of electricity over long distance
Distribution	The distributor of energy to consumer

Direct Current Open Energy System (DCOES) in Tokoro [56] is a local Smart Grid constructed covering all domains listed in Table 1.1. This system itself is designed to be energy efficient. A detailed description of DCOES will be listed in Section 1.1.

Furthermore, it is expected to be cost-effective to lower the purchase from the utility grid to save the bills. The detailed description of this system will be illustrated in Section 1.1. The previous data collected from the microgrid has been proven to increase self-sufficiency compared with stand-alone scenarios[2].

1.1 The DC-based Open Energy System (DCOES) at OIST

In December 2014, a local DC-based microgrid system was set up at the Okinawa Institute of Science and Technology (OIST) faculty housing area in Okinawa, Japan [42, 64]. It focuses on re-defining the conventional electricity grid systems in the form of interconnected nanogrid subsystems. It is designed to harness a mixture of renewable energy sources, including solar PV panels, and store the energy in rechargeable batteries. It is also designed to be deployed in various sorts of communities, especially to share surplus energy among different nodes. This renewable energy sharing project is organized by collaborating with three parties: Sony Computer Science Laboratories, Inc. (Sony CSL), local power supplier Okisokou Co. Ltd, and OIST. In contrast to the top-down centralized energy systems (Figure 1.4a), this DCOES proposed a novel type of distributed electric power system with bottom-up architectures (Figure 1.4b) which allows self-determined energy exchanges between residential nodes within a local DC microgrid community.



(a) Top-down architectures power transmission.

(b) Bottom-up type energy distribution.

Figure 1.4: Different architectures of power transmission type [2].

In conventional top-down energy systems of the existing electric utility grid, the energy source is usually remote, and the power flow is unidirectional. The conventional hierarchical system relies on large-scale power generation on the top and transmits power down in the grid infrastructure with long electric mileage. Power delivery to the user end of the chain fully relies on the power plant at the top of the chain. Power flow

[2]stand-alone refers to the condition where each node in the smart grid only considers stand-alone home systems with rooftop PV panels and batteries.

in these networks is synchronized, so it is challenging to have interconnections among different nodes on the user end. Furthermore, if we implement centralized system approaches to renewable energy systems, the entire power transmission is managed by a single entity (Figure 1.5 ①). One advantage of centralized renewable energy systems is that there are no lost opportunities due to the mismatch of demands and supplies [66]. However, a single failure can bring down the entire system with catastrophic blackouts, and centralized systems also come with high initial network costs.

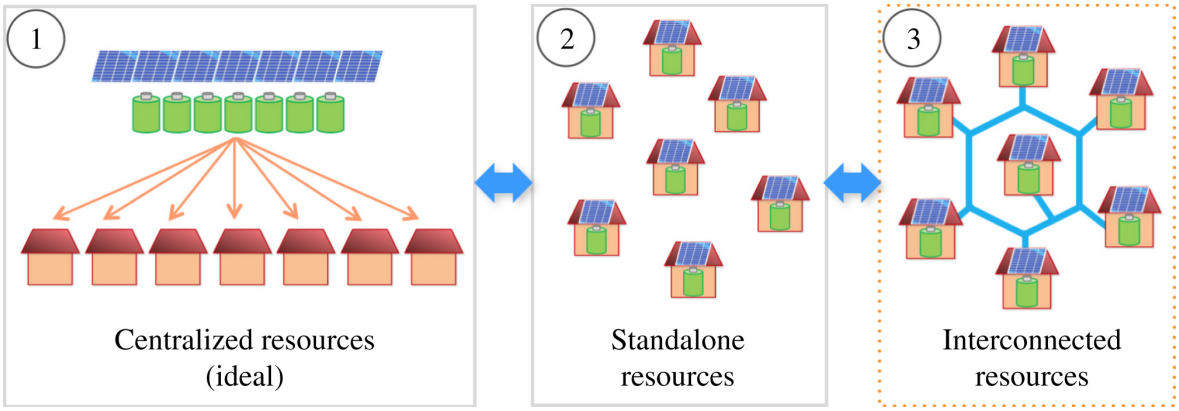


Figure 1.5: Topologies for renewable energy system, Werth et al. [66].

On the other hand, in distributed bottom-up energy systems (Figure 1.5 ②), the energy source is close to consumers with short electric mileage, and energy stored in batteries can fill the temporal gap between supply and demand [56]. For instance, commercial photovoltaic (PV) panels currently applied to homes belong to this category and are widely used to reduce electricity bills.

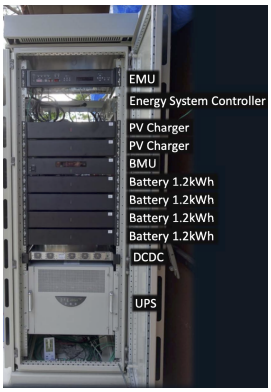
The OES further introduces interconnection among different consumers under the distributed network (Figure 1.5 ③). In this network, a single failure does not provoke entire system outages, and the initial costs could be much lower. It allows energy sharing among individual nodes and provides a more flexible way of using renewable energy.

In the early phase of the DCOES project, 19 inhabited houses and an electric room (ER, the weather station) were connected as one community in the energy grid. Each house is equipped with PV panels and an energy storage system (ESS) including lithium-ion batteries, through which it is connected to the others via DC networks [57]. Figure 1.6a and 1.6b show the PV panels installed for each house and the ESS, respectively.

Since 2020, the hillside faculty houses have been divided into three routes A, B, and C. As illustrated in Figure 1.7, each route contains 7 (route A), 6 (route B), and 6 (route C) houses for further artificial intelligence (AI) agent testbed. With its new ability to request and receive energy from the local neighbor nodes, this system has been proven in a real environment to be efficient in the use of renewable energy, flexible in size, as well as to increase dependability when used with utility electricity.

The DCOES is a distributed, heterogeneous, flexible platform that can scale up in different regions. In areas like Okinawa with tropical island weather, it is possible to broadcast the same infrastructure according to the dynamics, distributed energy

(a) PV panels equipped on the roof of each house.



(b) Energy storage system (ESS) installed in each house.

Figure 1.6: Structure of installation in DCOES.

Figure 1.7: Landscape of different routes with all 19 houses in the community.

resources, and utility grid supplies. Furthermore, a bottom-up design is more resilient against network failures and blackouts.

1.1.1 Different approaches for optimization of energy grids

So far, various approaches for optimizing the energy exchange have been applied to the DCOES. Werth et al. [65] devised a peer-to-peer control for DC microgrids. They compared different topologies with the same input data. They found that PV and battery with DC exchanges outperformed the PV-only (classic home system) cases and the standalone (without DC exchanges) cases and performed close to the theoretical limit of centralized control. They have proven that the bottom-up approach and decentralized design in a microgrid are promising under the Information Communication Technologies (ICT).

In addition, Sakagami et al. [43] also compared the standalone cases and DCOES case. They calculated the DCOES self-sufficiency rate (DSSR) to evaluate the performances in each case and found the optimal configurations of PV and installation cost. They reported that 4.6 kW PV and 3.6 kWh battery per house is the optimal configuration for the DCOES in OIST, and it could be changed with different installation costs. In addition, the cost recovery period of the DCOES can further be cut down if more PVs and batteries are installed, while the current recovery period is 15 years.

Furthermore, Werth et al. [66] proposed an evaluation model to analyze the impact of MG topologies on self-sufficiency for a given size of batteries and PV panels. They utilized three topologies: centralized resources (Figure 1.5 ①), stand-alone resources (Figure 1.5 ②), and a multi-microgrid topology with autonomous exchange (Figure 1.5 ③) for evaluation. They reported that the interconnected system showed a 10% increase in performance compared to the stand-alone design.

1.1.2 Bid for markets

Besides the controls mentioned above of the DCOES, Spasova et al. [51] presented a conceptual solution for trading in the decentralized microgrid. They merged the double-auction algorithm for the implementation of the trading platform. They reported that the results indicate that using P2P trading can reduce the electricity cost by 50%. As it is a conceptual study, they pointed out that further research should be carried out in order to be more adaptable to different power sources and respond to the individual prosumer.

1.1.3 Machine learning method for reducing surplus energy

More recently, Kawamoto and Rajendiran [28] showed that using machine learning techniques could further optimize the energy exchange by minimizing the wasting of the surplus of solar energy. They compared linear regression, simple RNN (Elman-net), and LSTM methods in predicting the future hourly consumption from the past days to maintain the charging space for the battery. Among all methods, Elman-net showed the best accuracy in demand prediction. The battery could, in advance, reserve more

charging space and reduce the time of wasting surplus energy after the battery gets fully charged.

One existing problem of the current DCOES is that energy sharing follows a fixed rule-based policy. It can be manually modified, which is demanding as each house has different usage and storage level. I want to explore other machine learning methods to increase the efficiency of energy sharing. There are many previous works using machine learning approaches in energy management, and I articulate related ones in Section 1.2.

1.2 Energy grid systems and machine learning applications

Energy management has always been one of the key issues for energy storage systems. Many previous works have explored different approaches under different conditions and systems for allocating energy. As Figure 1.8 shows, renewable energy generation system, storage system, and load are the main elements of a general microgrid system. The utility grid can include different power generation sources, such as coal, natural gas, nuclear energy, etc. And the renewable generation system can also include various types of sources, such as solar power, wind power, water power, etc. While storage system usually stands for the energy storage sectors, where various types of batteries can be applied.

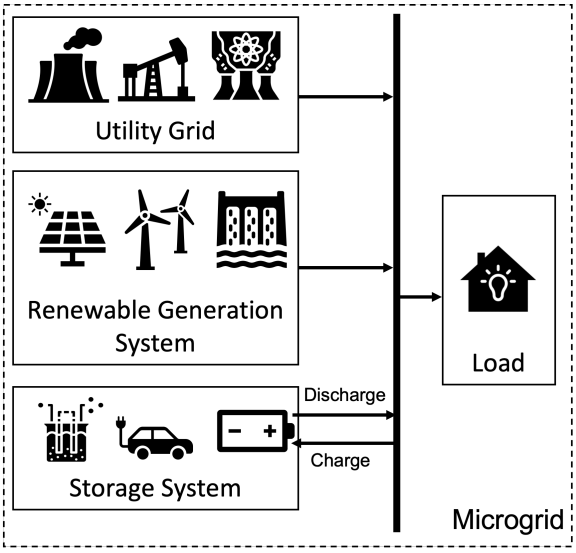


Figure 1.8: A general microgrid system model with the integration of generations, storage, and loads components.

1.2.1 Energy management approaches

Facing the challenges of balancing energy demand distribution and allocation with different capacities and demands, Trivedi et al. [59] proposed a stochastic cost-benefit framework for allocating energy storage system in the distribution network for load

leveling based on maximizing the ratio of NB (net benefit)[3] over NC (net cost), where NB is the sum of the arbitrage benefits and the benefits derived from dividing the other cost components by their differences, and NC is the total cost.

Energy management problems in power systems with renewable energy integration and/or energy storage have been extensively studied in previous publications. These problems can be classified into two main categories: on-line and off-line energy management. On-line categories refer to real-time energy management problems, while off-line tackles cases where data is collected and management is executed in an off-line manner. Under stochastic demand and/or renewable energy generation, Codemo et al. [8], and Grillo et al. [17] investigated the on-line energy management problem by assuming a stationary stochastic process with known distributions for the demand and/or renewable energy generation. While Zhang et al. [67] addressed the off-line energy management problem under the ideal presumption that the generated renewable energy and the load demand are either deterministic or known before scheduling. In addition, Rahbar et al. [40] proposed an off-line optimization approach with a "sliding-window" based sequential optimization on designing an on-line algorithm.

Furthermore, energy management in islands or confined areas also face a lot of challenges. Similarly to Okinawa, which includes a major island and some small offshore islands, many regions are island areas that are isolated or energy deficient. Hybrid renewable energy has been supplied to fulfill the electricity demands in these places. Singh et al. [48] presented a study of microgrids in rural areas consisting of PV, wind, biomass, and battery ESS. They proposed an artificial bee colony (ABC) algorithm to minimize the total net present cost (NPC) of the system to optimize the allocation of different energy sources.

1.2.2 Energy management by reinforcement learning

Reinforcement learning (RL) formalizes the idea that punishing or rewarding an agent for its behavior makes it more likely to forego or repeat that behavior in the future [54]. In addition, reinforcement learning provides a useful framework to conceptualize interaction in machine learning, and it has achieved various successes and drawn massive attention during the past recent years. Examples of successful applications of reinforcement learning are playing video games, such as the breakthrough study pertaining to the learning of playing Atari games from raw pixels in Mnih et al. [35]. By combining reinforcement learning (Q-learning) with deep learning (neural networks) to let Deep Q-network (DQN) agents play classic Atari video games, they found this combination surpasses the performances of all previous algorithms and achieves a human-level control. This is a breakthrough not only in reinforcement learning but also in the entire artificial intelligence domain, as it indicates that trained agents can play on the human level and even outperform. Other sophisticated strategy games such as Go in Silver et al. [47], Dota in OpenAI [37], and trained simulated robots to follow human instructions in Christiano et al. [7] are all notable research with reinforcement learning methods.

[3] Net benefit can be viewed as the financial objective functions. What the objective function considered is to maximize the economic gains due to energy storage system (ESS) integration.

There are plenty of existing sequential decision-making problems that would take reinforcement learning into account as a possible solution due to its properties. Many previous works also tried applying RL to the energy management sector. Guan et al. [18] presented an approach that uses TD(λ) algorithm to derive the optimal energy storage system control policy, as well as defines the state and action spaces and reward function so as to reduce residential consumers' electric bills. They confirm TD(λ) can achieve higher convergence rates and higher performance in non-Markovian environments[4]. Kim and Lim [29] also used Q-learning in learning buying, charging/discharging, and selling actions for a smart energy building. Levent et al. [31] used dynamic decision-making from training an optimal RL algorithm with past and limited data-set in an island MG. With a peer-to-peer trade pricing structure, Zhou et al. [68] used RL to represent the suggested energy trading process as an MDP and identify the best decision inside the MDP. Moreover, they applied a Fuzzy Q-learning in trading price rules.

Many reinforcement learning techniques based on deep neural network approaches have been proposed to improve and learn different tasks, which also happens in smart grid applications. François-Lavet et al. [15] applied deep RL (DRL) to MG with PV panels. They introduced a particular deep learning architecture to draw knowledge from historical time series of consumption and production as well as accessible forecasts. The value function's neural network (NN) representation effectively generalizes the policy to circumstances corresponding to previously unknown configurations of power demand and solar irradiation. Similarly, Sogabe et al. [49] utilized Deep Q-network (DQN) and Gibbs Deep Policy Gradient (GDPG) methods for both discrete and continuous action space. The outcomes demonstrate that the agent learned to select behavior that would maximize its reward and had successfully captured the energy demand and supply characteristics in the training data. Chen and Su [6] applied the DRL method in local energy trading. The prosumer's decision-making process was created as an MDP with multiple continuous variables by modeling local energy trading strategies in the proposed holistic market model. Tomin et al. [58] designed DRL for making decisions under flexible energy sources in MG when there is uncertainty about future electricity use and weather-dependent PV production at each time step. Hau et al. [19] also used a value-based reinforcement learning and DQN-based energy management algorithm for energy trading. They applied a piece-wise utility function in response to the dynamic environment under dynamic pricing.

Meanwhile, many reinforcement learning methods are typically designed for single agents (to satisfy MDP properties), which have a bad performance for multi-agent systems, neither for cooperative nor competitive environments, since the joint action space of the agents grows exponentially with the number of agents [13]. Approaches such as Iqbal and Sha [23], Lowe et al. [33] are proposed for multi-agent RL. DRL has also been explored and applied in multi-agent systems for energy sectors. Foruzan et al. [14], Vázquez-Canteli and Nagy [60] views energy suppliers and prosumers as agents in a MAS which tries to optimize their rewards. The purpose of Ahrarinouri et al. [1]'s work is to explore the multi-agent reinforcement learning approach for residential

[4]non-Markovian environment: the environment does not satisfy Markov assumptions (i.e., the probability distribution over the next state depends only on the current state).

multi-carrier energy management. Vázquez-Canteli et al. [61, 62] set up a TensorFlow-based building energy simulator *CitySim* to monitor the building energy management problems. After modeling the single building's heating and cooling issues, they further broadcasted the work to a multi-agent scale in their CityLearn environment for building energy coordination and demand response [62, 63].

Previous works have explored various approaches to optimizing energy grids. However, energy management with RL in a DCOES-like system is not discussed yet. Whether RL could work in this system needs to be verified. In addition, which reinforcement learning methods and components are needed to be considered? As Section 1.1 explains, one problem of current methods is improving the efficiency of the grids. I need to ascertain how to combine the controller with reinforcement learning. I also need to verify what state-action representations could influence the performances.

1.3 Thesis outline and contributions

With the above DCOES as a test bed, I build intelligent learners (charging/discharging in batteries and requesting/accepting energy from other agents) and compare them with the current rule-based controller. I deploy reinforcement learning methods to the DCOES as reinforcement learning is a promising candidate approach with the capability to adapt with regard to the available energy. Section 1.2.2 list some existing energy management with reinforcement learning methods, and only some of them mentioned applications to multiple agent systems and could be applied to embedded systems. I first explored the reinforcement learning method in single-house experiments and further extended it to multiple houses cases. The organization of this thesis is structured as follows:

Chapter 1 Introduction This chapter surveys the background and necessity of leveling up efficiency in energy usage. In this chapter, the previous study on DCOES is covered, as well as other related work in energy management with RL methods. The problem statements and challenges of the thesis work are explained.

Chapter 2 Theoretical Background and System Description This chapter provides theoretical knowledge in reinforcement learning. Data arrangement and energy management unit is described, as well as the APIS emulator structures of DCOES.

Chapter 3 Single House Energy management with Reinforcement Learning This chapter tests the performance of reinforcement learning in single houses with a PV panel and a battery, without energy exchanges across houses. I developed a battery simulator and applied different RL methods to the data from different houses with different input states.

Chapter 4 Multiple houses Energy management with Reinforcement Learning This chapter presents multiple houses RL using the APIS emulator. Different options of states with Deep Q-network and with prioritized replay methods are proposed and tested.

Chapter 5 Conclusions, Discussion and Perspective In this chapter, I summarize the overall works presented in this thesis and discuss future perspectives.

Contributions of this study

The main contributions of this study come in several folds:

- **Data management:** historical data collected from all houses and weather stations are massive and sometimes contains many missing data. Therefore, cleaning up the data and selecting the related ones as our input states are crucial. Furthermore, I re-arranged the data format to apply to the multiple-house emulator for multi-agent cases. I made this data public which allows other researchers to benefit from this for further research.

- **Apply reinforcement learning to single houses:** whether it is possible to apply reinforcement learning methods to the houses for energy consumption opens the first page of my experiments. I first set up a battery simulator for a single house and verified its usability. Then I compared the tabular method and deep reinforcement learning method in the single-agent case and found tabular method is possible to control the current of the energy storages. But it has limitations in structuring the state representations. The DRL method converges faster in learning than the tabular method. And having time-of-day information in the state can reach smoother learning in the early days, and this cyclic value slower the performance. The learned actions also changed more frequently.

- **Compare different options of states influencing the performances for single houses:** I designed for charging and discharging the battery to explore the reward function of each agent. In addition, I compared different input state options and compared the performances.

- **Setting up multiple agent reinforcement learning to the DCOES:** I applied multi-agent reinforcement learning methods to the energy exchange system. I set up the action selections for each node (agent) in the DCOES to change the actions for requesting and accepting power from other nodes according to the state-action value and found that learned actions can adapt to houses' battery status automatically. It infers that manually modifying the exchange rules can be replaced.

- **Compare different options of states influencing the performances in multiple houses:** different options of states and how reinforcement learning improves the shared energy among each node to reduce the energy consumption from the external power lines.

Chapter 2

Theoretical Background and System Description

This chapter introduces the essential knowledge of Reinforcement Learning (RL) and Multi-agent Reinforcement Learning (MARL). In addition, concepts of the Markov decision process (MDP), as well as the Partially Observable Markov Decision Process (POMDP), are introduced. Deep Reinforcement Learning (DRL), multi-agent reinforcement learning, DCOES system, and APIS emulator are further explained.

2.1 Reinforcement learning

Reinforcement learning is one subcategory of the machine learning approach [53]. Unlike supervised learning, which uses labeled datasets to train algorithms to classify data or predict outcomes accurately, or unsupervised learning, which learns patterns from untagged data, reinforcement learning is trying to figure out how the agent or AI could learn an optimal policy over time to maximize its reward value or goal-achieving. In supervised learning, we know the correct output for every example input. For instance, for identifying an animal in a picture, we have plenty of training data and must know the true answer for each training example. While for reinforcement learning, there are no training examples. It solves problems that never have been solved before. We know the inputs (states), but correct (i.e., optimal) actions are not provided in contrast to supervised learning. We can only observe the consequences of actions taken at states (some effects of that action) after taking some actions. Some actions produce good outputs, while others don't. For example, for an agent or a person who performs self-driving in snow, which has never been done before, they have to learn as they go in the actual situation. Actions could be chosen from turning the wheel, accelerating, or breaking. While observing the changes in the environment, such as position or speed, the agent can get a sense of whether the situation goes good or bad, i.e., the car stays in the lane or is varying off the road. First, the agent could try random changes like steering a little left or right or breaking, but once the agent starts to observe the outcomes, it starts to learn that some actions are more appropriate in certain situations than others. In reinforcement learning, the system associate actions with positive and negative rewards (negative values can be a penalty), so the goal is to reinforce actions

with higher rewards over time. Actions associated with higher rewards will be more likely to be repeated in the future, which essentially solves the problem. A typical framework of single agent-environment interaction in reinforcement learning can be explained with Figure 2.1.

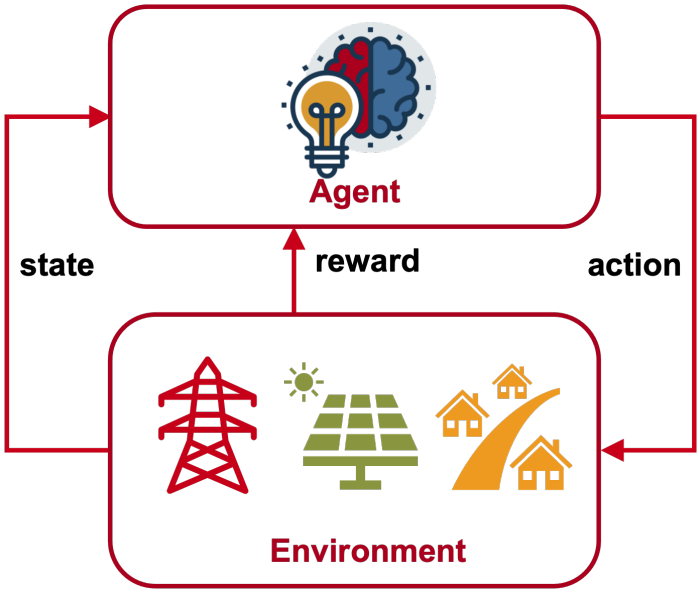


Figure 2.1: Single agent-environment interaction framework in reinforcement learning.

RL problems are often formulated as Markov decision processes as RL is a set of methods that learn an optimal policy in an environment, whereas MDP is a formal representation of such an environment. MDP description is introduced in the following section.

2.1.1 Markov decision process (MDP)

A Markov Decision Process (MDP) is defined as a stochastic discrete time control process. It refers to processes that make decisions based on some amount of uncertainty. MDP uses only the current state to evaluate the next actions without any dependencies on past states or actions. MDPs provide a mathematical form to reinforcement learning problems as their nice property in sequential decision-making. An MDP model comprises several elements as follows [46]:

- a (finite) set of states $\mathcal{S}$ which describes the current situation of the agent
- a (finite) set of actions $\mathcal{A}$ which affects the dynamics of the process, and $\mathcal{A}(s)$ is the admissible action set when state $s \in \mathcal{S}$
- a set of reward $\mathcal{R}$ or reward function $R_a(s, s')$ which is given to the agent when it takes an action $a \in \mathcal{A}_s$ at a state $s \in \mathcal{S}$, and the state transitions to $s' \in \mathcal{S}$
- one-step dynamics function (the state transition probability function) of the environment T: $p(s'|s, a) = p(S_{t+1} = s'|S_t = s, A_t = a)$ at time t

- discounting factor $\gamma \in [0, 1]$

For simplicity, time is divided into discrete steps. At each time step, the agent observes the state of the environment $s_t \in \mathcal{S}$ and decides an action $a_t \in \mathcal{A}$. For each state, action, and the next state, there is a transition function $T(s_t, a_t, s_{t+1}) \to [0, 1]$. After taking action, the agent receives an immediate reward $r_{t+1} \in \mathcal{R}$, reflecting how good the action is, and observes a new state of the environment $s_{t+1} \in S$. MDP provides a mathematical framework for modeling decision-making in circumstances where outcomes are partially determined at random and partially controlled by the decision-maker.

The goal in RL uses reward assumption and tries to maximize the (expected) cumulative reward via taking a sequence of actions while visiting a sequence of states. The interaction between the agent and the environment proceeds as follows: initially, the agent is placed at an initial state $s_0 \in \mathcal{S}$, which can be predetermined or sampled from a distribution over $\mathcal{S}$; then, the agent selects and execute an action a_0; the reward r_0 and the next state s_1 can be obtained based on the transitions; then the agent would make the next decision of action based on received reward and execute the next step of action a_1, and the act till it meets the terminal conditions. Its discounted cumulative return at time step t can be formulated as

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots. \tag{2.1}$$

With the above definition of the expected return, we try to maximize this value with the discounting rate γ ($0 \leq \gamma \leq 1$, also called *discount factor*), which determines long-term rewards with the discounted rate. The larger the value of γ is, the more the agent cares about the future reward; the smaller the value of γ is, the greater the discount would be, so the agent cares more about the immediate reward. In general, γ is set close to 1 in most applications. Furthermore, from the above description that the agent tries to take a sequence of decisions when calculating the return, and this sequence of actions is called policy π. There exists an action (for a deterministic strategy) or an action distribution (for a randomness strategy) for any $s \in S$ maps to $\pi(s)$. Then, based on the policy π, a state value v_π could be calculated. For each state $s \in \mathcal{S}$, the mapping value can be written as

$$v_\pi(s) \doteq E_\pi[G_t | S_t = s]. \tag{2.2}$$

Where E_π denotes the expected value of policy π at timestep t. When the agent chooses an action at a state according to the policy π, V_π could be written in the form of expectation of this distribution as above. In addition, due to the definition of MDP, G_t could be written as $G_t = R_{t+1} + \gamma v_\pi(S_{t+1})$ as the **Bellman equation** in Equation 2.3.

$$v_\pi(s) \doteq E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1} | S_t = s)]. \tag{2.3}$$

Figure 2.2 shows how this value is been calculated.

The agent's main objective is to learn the optimal policy π^* from possible policies π, which produces the highest possible cumulative long-term payoff. If a policy π' is said to excel or equal to a policy π, it is an optimal policy (the optimal policy may not be unique). The value function of the state v_π reveals the accumulative reward the

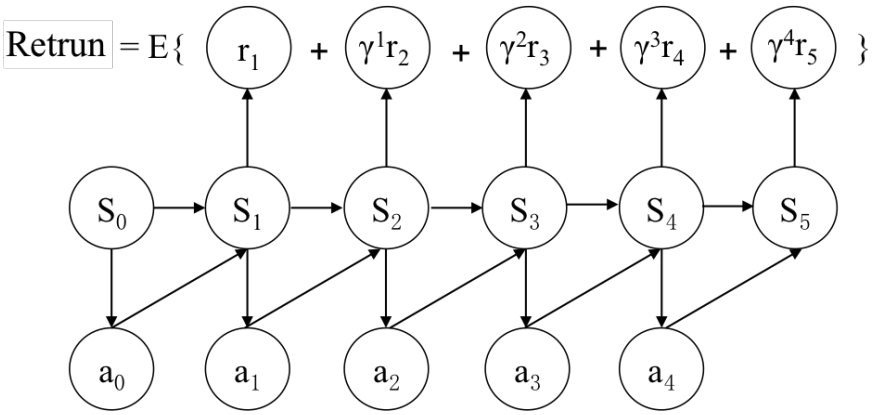


Figure 2.2: Return function.

agent could get when it acts according to policy π at state s. An action-value function q_π could be introduced to evaluate the action agent takes at state s. Following the above explanation of v_π, q_π could be written as

$$q_\pi(s,a) \doteq E_\pi[G_t|S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]. \quad (2.4)$$

The agent can use approaches such as dynamic programming (DP) to calculate $q^*(s,a)$ of the optimal policy and then update the policy in Equation 2.5.

$$q_*(s,a) = E_\pi[R_{t+1} + \gamma max q_*(S_{t+1}, a')|S_t = s, A_t = a]. \quad (2.5)$$

The optimal policy has the highest action value at state s, which could be written as $\pi^*(s) = \arg\max_{a\in\mathcal{A}} \quad q^*(s,a)$.

There are some typical ways for action selection based on the current action-value function, which we also call state-action value (Q-value, $Q(s,a)$), such as ϵ-greedy and *softmax* (Boltzmann action selection [53]). Softmax action selection is implemented as Equation 2.6 with a hyperbolic lowering of the temperature parameter τ,

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_b e^{Q(s,b)/\tau}}, \quad (2.6)$$

where τ is a positive scalar that determines the “greediness" of the agent. For high temperatures, the agent is inclined to select all actions with the same probability while for low temperatures, the agent tends to behave close to greedy action selection, namely, the agent is apt to take the action with the highest expected reward.

2.1.2 Partially observable Markov decision process (POMDP)

One feature of the MDP is that if the process fulfills the markovian property, where the future state is independent of the past with given the present, an optimal policy's existence $\pi^* : \mathcal{S} \to \mathcal{A}$ could be guaranteed.

Unfortunately, this conclusion could not be applied to the partially observable Markov decision process (POMDP). The POMDP is a combination of a regular MDP

to model system dynamics with a hidden Markov model that connects partially observable system states probabilistically to observations. Similar to the MDP in the above section, a POMDP can be described as a tuple as $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, \Omega, \mathcal{O}, \gamma)$ as follows [26]:

- a (finite) set of states $\mathcal{S}$ which describes the current situation of the agent
- a (finite) set of actions $\mathcal{A}$ which affects the dynamics of the process, and $\mathcal{A}(s)$ is the admissible action set when state $s \in \mathcal{S}$
- a set of reward $\mathcal{R}$ or reward function $R_a(s, s')$ which is given to the agent when it takes an action $a \in \mathcal{A}_s$ at a state $s \in \mathcal{S}$, and the state transitions to $s' \in \mathcal{S}$
- one-step dynamics function (the state transition probability function) of the environment T: $p(s'|s, a) = p(S_{t+1} = s'|S_t = s, A_t = a)$ at time t
- a set of observations Ω
- a set of observation probabilities $\mathcal{O} = \mathcal{O}(o|s', a)$, it is conditioned on the reached state and the taken action
- discounting factor $\gamma \in [0, 1]$

In this case, the agent could not always acquire the current state s_t. Finding an optimal policy in POMDP is significantly difficult, and Lusena et al. [34] illustrates an overview of many variations of POMDP and difficulties in finding optimal policies. Even if the agent knows Q^*, it still cannot behave optimally as it cannot always know the current state s. In POMDPs, the agent cannot directly observe the complete system state, but the agent makes observations that depend on the state. It uses these observations to form a belief about what state the system currently is in. This belief is called a belief state and is expressed as a probability distribution over all possible states. The solution of the POMDP is a policy prescribing which action to take in each belief state.

For a single agent, POMDP is already difficult as the belief states continuously cause an infinite state set. While in multi-agent cases, due to the dynamic changes of the environment and other agents, it would be much more difficult to solve compared to MDPs. Furthermore, the time complexity of solving POMDP iterations is exponential in both states and observations spaces, and the dimensionality of the belief space also grows with the number of states. In fact, in most applications, the size of real-world problems is outside the scope of tractable exact solutions.

2.1.3 Different methods in RL

Suppose we have a given task, and state-action transition probability and reward function is known. We can calculate each state under a certain policy. With this calculation, we can further update the policy for each state to find the optimal policy. This is the typical "policy iteration" method.

On the other hand, in the case where the real purpose of updating the state-action estimation is to compare the difference in the return of each action in a certain state

to choose the optimal operation for different states according to the Bellman equation, the estimation of each state may not need to be absolutely accurate. This is considered as the "value iteration" method.

The policy iteration method of evaluating and improving policies is unified into generalized policy iterations (GPI) [54]. It can be considered that in RL, the MDP framework gives the agent's perspective on the task (environment), while the GPI tells the agent how to analyze and learn the basic idea of the task. And this kind of method to "dynamically plan" the policy based on the state and transition probability is called the dynamic programming (DP) method.

DP method can be applied to calculate the state-action value $Q(s, a)$, both the transition and the reward function are required in using DP. For small state-action spaces, it is an efficient approach, but a lot of environments are accompanied with large state-action spaces, and the transition probability and rewards of each state-action pair are not determined in advance in the environment and their dynamics. To tackle this problem, the temporal difference (TD) approaches such as *Q-learning* and *SARSA* can be applied, by which RL learners can find an optimal policy through interactions with the environment without knowing the dynamic model of the environment. The incremental evaluation equation in TD methods can be written as: $Q_{t+1}(s, a) = Q_t(s, a) + \alpha[G_t - Q_t(s, a)]$, where $G_t = R_t + \gamma Q(s_{t+1}, a_{t+1})$ and a_{t+1} is the action under s_{t+1} in the next step. The difference between G_t and Q_t is called the *TD error*. TD methods are reliable and theoretically proven to be convergent if the learning rate is small enough [55].

Either DP or TD method uses a look-up table by saving each state value (or Q value) and then continuously updating it. Many tasks in reality would encounter a very large state and/or state-action space, so it is obviously unrealistic to evaluate each state according to the previous description. We can use function approximation to give a more reasonable estimate of the state that has been seen.

2.2 Deep reinforcement learning

Deep reinforcement learning (DRL) combines deep learning and reinforcement learning. In tabular RL, each state and the Q value of each action in this state are stored in a look-up table. However, many environments can have a large number of states, and the computer will not have enough memory to store them as well as it is time-consuming to search for the corresponding state in a large table under tabular methods. The neural network (NN) can handle this problem in which we can take the state and action as the input and obtain the Q value of the action to avoid recording the Q value in a table. Because deep learning is incorporated, agents can decide what actions to take with massive inputs, such as every pixel presented on screen in a video game illustrated in Mnih et al. [35, 36], Silver et al. [47]. In video games environments, pixels of the screen can be modeled as the input of states in the form of NNs. While all data passes through the NN with hidden layers and the activation function, the output layer can be the objective in the RL framework. In Deep Q-Network (DQN), the value function is approximated with the deep neural network. The Q value function

is updated iteratively as Equation 2.7

$$Q(s,a) \leftarrow Q(s,a) + \alpha(\boxed{r + \gamma \max_{a'} Q(s',a';\theta_i)} - \boxed{Q(s,a;\theta_i)}), \tag{2.7}$$

where θ_i is the weight of the network at i-th iteration. In DQN, the target Q value (*Q-target*) is shown in a grey box and the prediction (*Q-predict*) is highlighted in a red box in Equ 2.7. The loss of the network at i-th iteration is calculated with the mean-squared error between the target value and the prediction in Equation 2.8

$$Loss = \frac{1}{m}\sum_{j=1}^{m}(y_j - Q(\phi(S_j), A_j, \omega))^2, \tag{2.8}$$

where y_i denotes the target Q values, and $Q(\phi(S_j), A_j, \omega)$ is the predicted Q value. However, small updates to Q may significantly change the policy in the sequence of observations and therefore change the data distribution, so having *experience replay* in the memory bank can make the NN updates more efficient. With experience replay, some previous experiences can be randomly sampled for learning. Furthermore, Mnih et al. [35] also used *fixed Q-targets* for disrupting correlations between the action and the target values. When fixed Q-targets are applied, two NNs with the same structure but different parameters are used. The NN that predicts Q-predict has up-to-date parameters, while the NN that predicts Q-target only updates parameters periodically. With these two improvements, DQN could outperform humans in some environments, such as Atari games.

The main differences between Q-learning and DQN are in three-folds:

- Replay buffer (for repetitive learning)
- Neural network to calculate Q-value
- Temporarily freeze the Q_target parameter.

DQN uses random sampling from the batch memory and the differences between the prediction and the target network. The agent would take a very long time to learn from the memory bank when the reward is scarce, and the agent tends to forget the previous experience. For example, in Montezuma's Revenge in Atari games, the reward is sparse, and almost all experiences have no useful information. Schaul et al. [45] modified the DQN method with Prioritized experience replay (PER) to tackle this problem. In prioritized DQN, agent samples experience according to the sample priority in memory rather than random sampling. Therefore it can find the learning samples we need more effectively. To define sampling priority, they utilize *TD-error* to decide the learning order. The larger TD-error is, the more space we have for prediction, which means this sample needs to be learned more, and the higher priority this sample is. The priority of experience p_i is proportional to the TD-error where $p_i = |\delta_i| + \varepsilon$, and ε is a small constant ensuring that no transition has zero priority. Adding a hyperparameter α, the priority is calculated from $p = p_i^\alpha$, where $\alpha \in [0,1]$ controls the difference between high and low error. To effectively sample according to p_i, a 'sum-tree' method is applied to the memory (A SumTree refers to a Binary Tree where the value of a node is equal to

the sum of the nodes present in its left subtree and right subtree). Applying DQN and prioritized DQN methods to energy management systems will be discussed in detail in later chapters.

Figure 2.3 shows a diagram design in the DRL of MG scenario. For an energy system with storage, The possible options of state variables can be power production, load, weather information, time information, and so on. While environment dynamics, power production, load, battery dynamics, and more can be considered. Action, in this case, can be discharging/charging action of the battery, sending/receiving energy, and others.

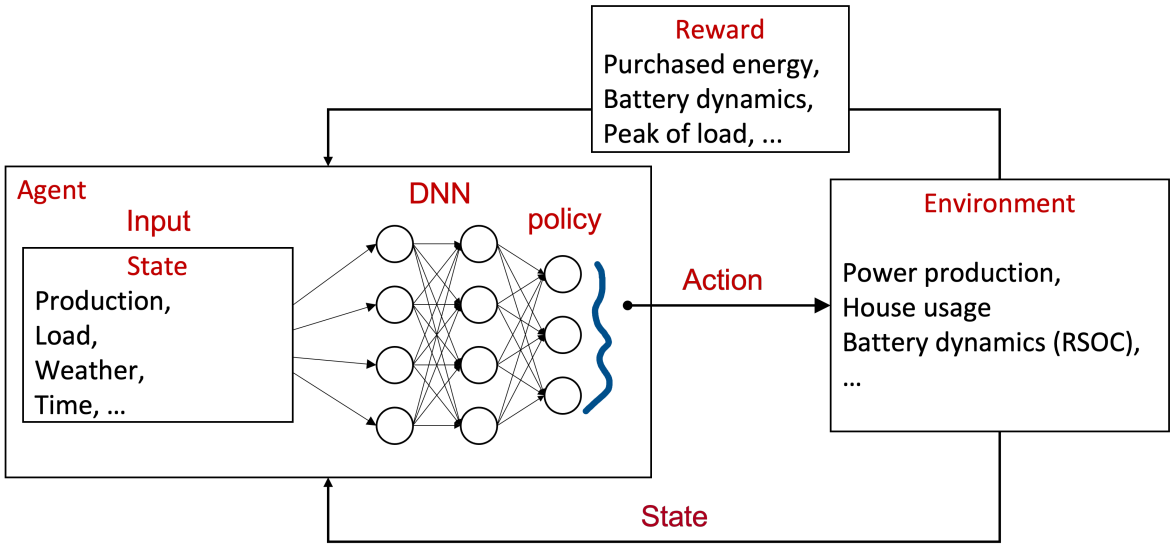


Figure 2.3: Deep neural network structure diagram for the RL agent in MG.

2.3 Multi-agent reinforcement learning

If there is more than one agent in the environment, i.e., a multi-agent system, all agents need to interact with the environment, as well as with other agents. Figure 2.4 shows a framework of the multi-agent system for agent-environment interaction in reinforcement learning.

In real applications, it is also more reasonable and natural to build models with multiple agents. For multi-agent systems, in general, could not fulfill the MDP properties. And the reason is quite apparent: one agent's action will change the environment, but usually this agent's behavior and intention are not known to others, and for other agents, vice versa. They could only observe partial information, and due to the situation in such a Markov game, reinforcement learning faces many difficulties in multi-agent system (MAS) studies. Following the self-driving example, apparently, there would be more than just one automobile on the road, and we have to take other vehicles into account, such as how to deal with the situation when one car suddenly breaks, or how to plan the route to avoid traffic jam, or how to manage the situation when cars keep leaving the lane while others keep joining temporarily. Interactions

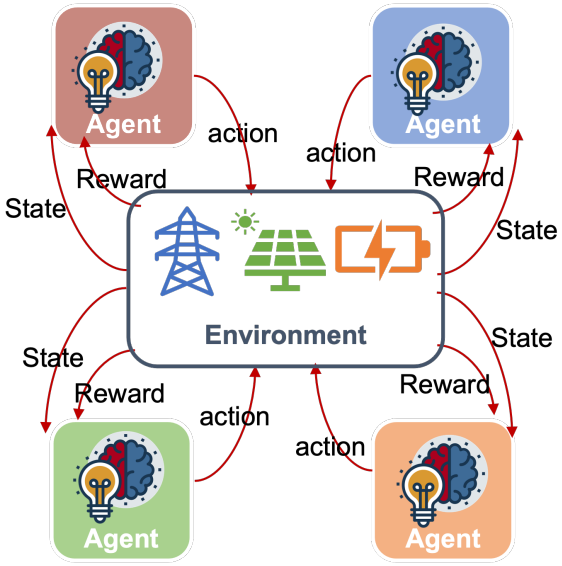


Figure 2.4: A multi-agent environment interaction framework in reinforcement learning.

among different vehicles are quite important in this multi-agent system. It might be possible to consider this problem from a single agent point of view as a super-centralized brain controls the whole system, but it will bring a plurality of problems [20]. However, it would be much more natural to consider each car as one agent. Many algorithms centrally train the strategies of all agents, and after the training, agents can have the ability to make distributed decisions, such as COMA in Foerster et al. [13] and MADDPG method in Lowe et al. [33]. However, due to the centralized training process, it also has a certain scalability problem. When the number of agents is large, it may face the risk of not being able to learn. Besides, there are also methods that use communication to alleviate the impact of environmental non-stationarity and local observability by increasing the transfer of information between agents [12, 21, 22].

2.4 Energy management of the DCOES

The Open Energy System on the OIST campus site includes 19 inhabited houses that are equipped with photovoltaic panels and lithium-ion batteries and interconnected by a DC power bus line. Figure 2.5 shows the overall architecture of the OIST DCOES. Charging/discharging and power exchange decisions are made by the energy storage server (ESS) in each house.

Figure 2.6 illustrates the basic energy inputs and outputs of each house and how they are connected in a network.

Figure 2.7 shows a detailed organization of the energy storage server (ESS). There are three input energy sources to the ESS: solar power generated from photovoltaic panels (pvc charge power), power purchased from the external power supply line (Powermeter p2), and exchanged power from other houses in the community (DC grid). The output is AC 100V power consumption in the household through an uninterrupted

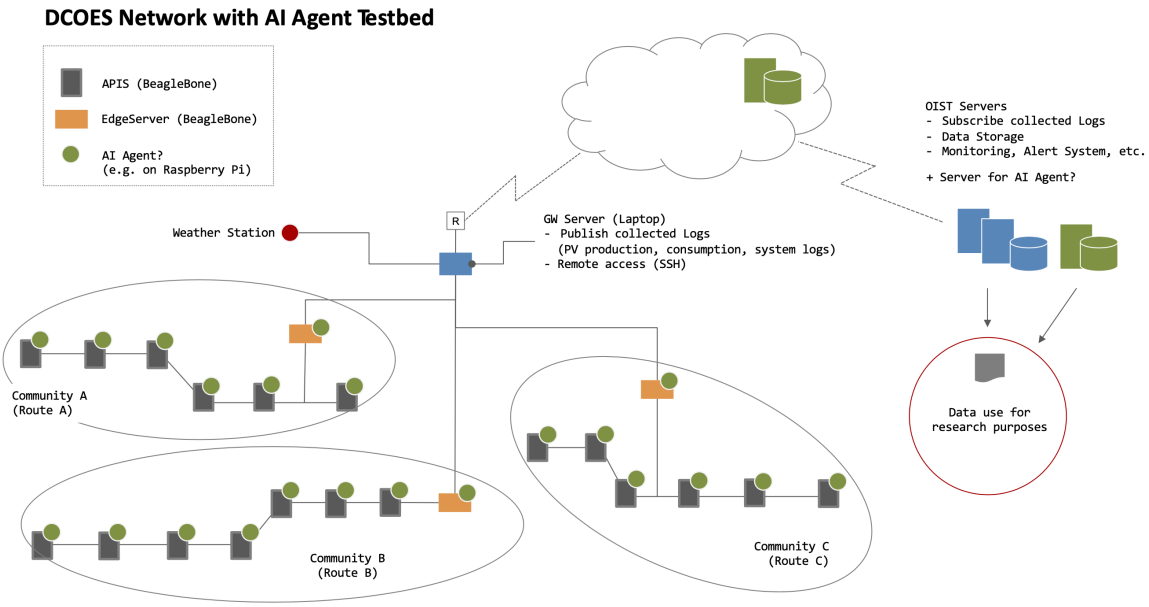


Figure 2.5: Overall architecture of the OIST DCOES.

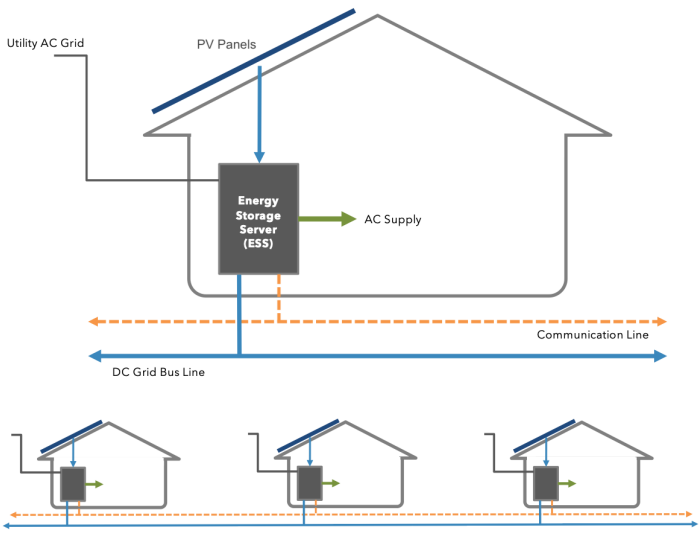


Figure 2.6: Sketch of the house energy storage system.

power supply (UPS) that performs AC-DC conversion (ups output power). DC power can be stored locally in the battery through the battery managing unit (BMU) (charge-discharge power) and exchanged across houses through the DC grid (grid power) In addition, 200V AC appliances, such as air conditioning, are additionally provided from the external power line without the control of the ESS.

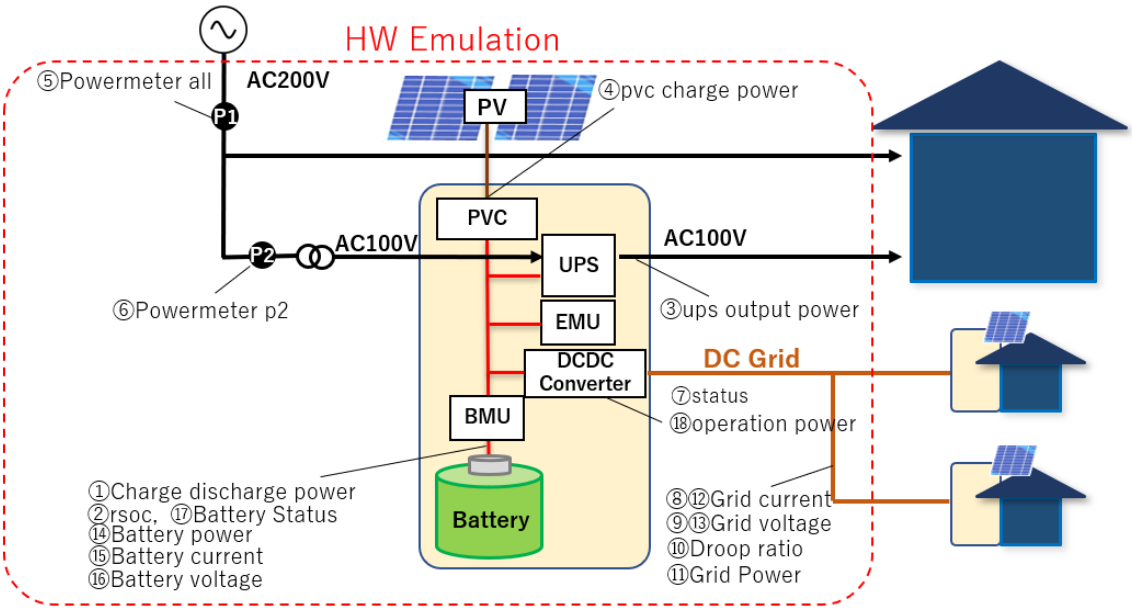


Figure 2.7: The organization the Energy Storage Server (ESS) for the DCOES.

2.4.1 Battery charge/discharge control by RSOC

In the standard operation of the DCOES, whether to use the local battery power or commercial utility AC power is decided based on the relative state of charge (RSOC) of the battery, as illustrated in Figure 2.8. The battery operation mode shows the relationship between the RSOC and UPS modes. UPS modes have two modes, when the RSOC remains higher than a threshold (30%) the battery operation mode is "BATTERY Mode" (also denoted as UPS mode = 2), in which the household power is supplied through the DC-AC converter and no utility AC power is used. When the RSOC drops below the threshold, the batter operation mode turns into the "BYPASS Mode" (UPS mode = 5), and the utility AC power is used directly. As the RSOC recovers above another threshold (35%) through recharging from the PV power, DC grid exchange, or the utility AC power, the battery operation model turns to the "BATTERY Mode" (UPS mode = 2).

Under this control, the power flow to the battery is determined by the following

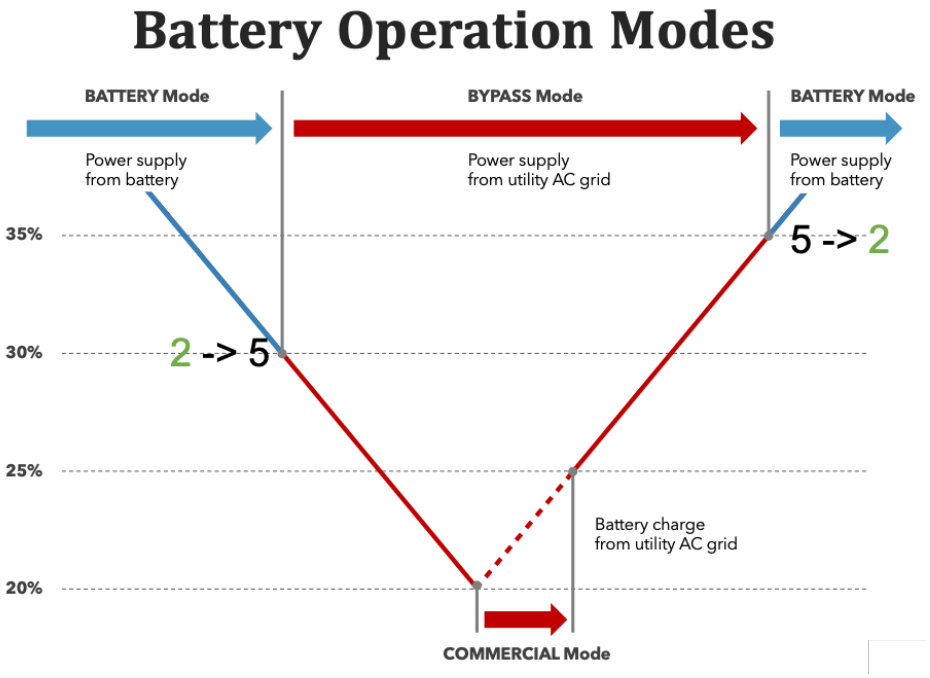


Figure 2.8: The battery operation mode control based on the RSOC.

equation (numbers are those in Figure 2.7):

$$\begin{aligned}\textbf{Power Flow to Battery} = {} & ⑪\text{DC Grid power} \\ & + ④\text{pvc charge power} \\ & + ⑥\text{Powermeter p2 (power consumption to the ESS)} \\ & - ③\text{ups output power} \\ & - \text{ac\_loss (Transition loss)} \\ & - \text{dc\_loss (ESS loss and DCDC loss).}\end{aligned} \tag{2.9}$$

2.4.2 The DCOES data preprocessing

The original data file and logs are given in *.csv* format, including 31 data points for each house and 14 data points for weather stations (See Appendix A). A preprocessing of data files is necessary for further uses of the data sets. In addition, we currently have housing data from the year 2014 to 2019, and weather station data from 2015 to 2019. We also have to remove any unreliable or incomplete data from the data because it is necessary for our testing and learning processes. We interpolate the forward value (last previous valid) of the missing points. Figure 2.9 shows data visualization on the first day of 2019 of house 214 over every 30s. The red line denotes the photovoltaic charge power, the magenta dashed line denotes the charge-discharge power to the battery, the black dot line denotes the consumption power, the green dotted curve denotes the relative state of charge (RSOC) of the battery, and the blue star curve denotes the

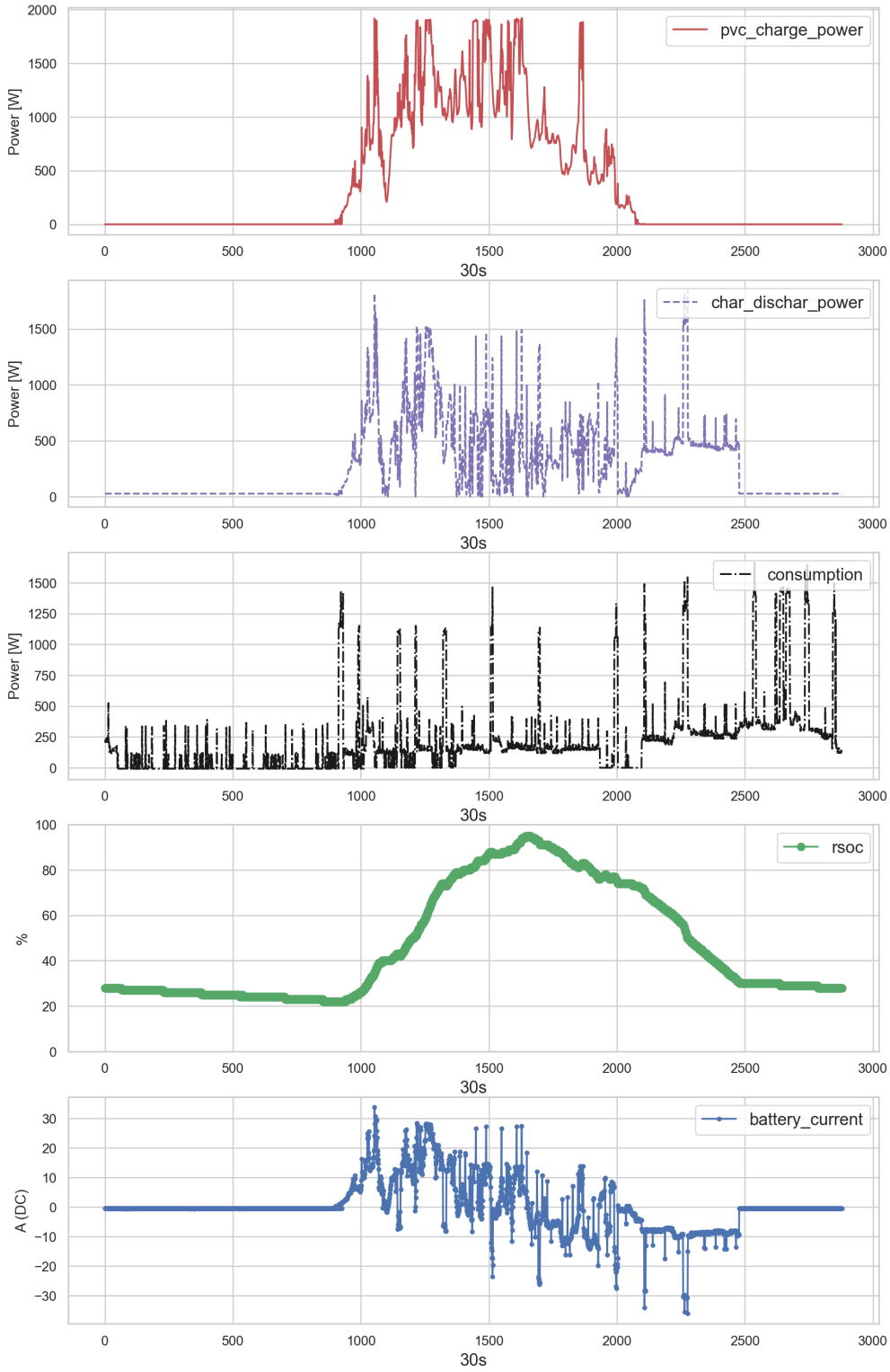


Figure 2.9: Visualization of valid data of House 214 over every 30s on Jan 1, 2019.

charge/discharge current of the battery of the day. We can tell from the figures that the RSOC level is changing according to the battery current values, which increase during the daytime with solar power, and decreases when the solar sources decline with sunset. Appendix A shows an example of historical log data of one house from the original file with all data points.

2.5 Autonomous Power Interchange System (APIS)

The Autonomous Power Interchange System (APIS) is an open-source power interchange management software that comprises the node software for P2P power interchange, the main controller for monitoring and visualization, and emulators of the DCOES hardware, including ESS [50]. Figure 2.10 shows a real-time battery monitoring of multiple houses. The dotted line represents that there is an energy exchange happening among 19 nodes. The battery state of charge (0% to 100%) is also depicted in the green part with different levels.

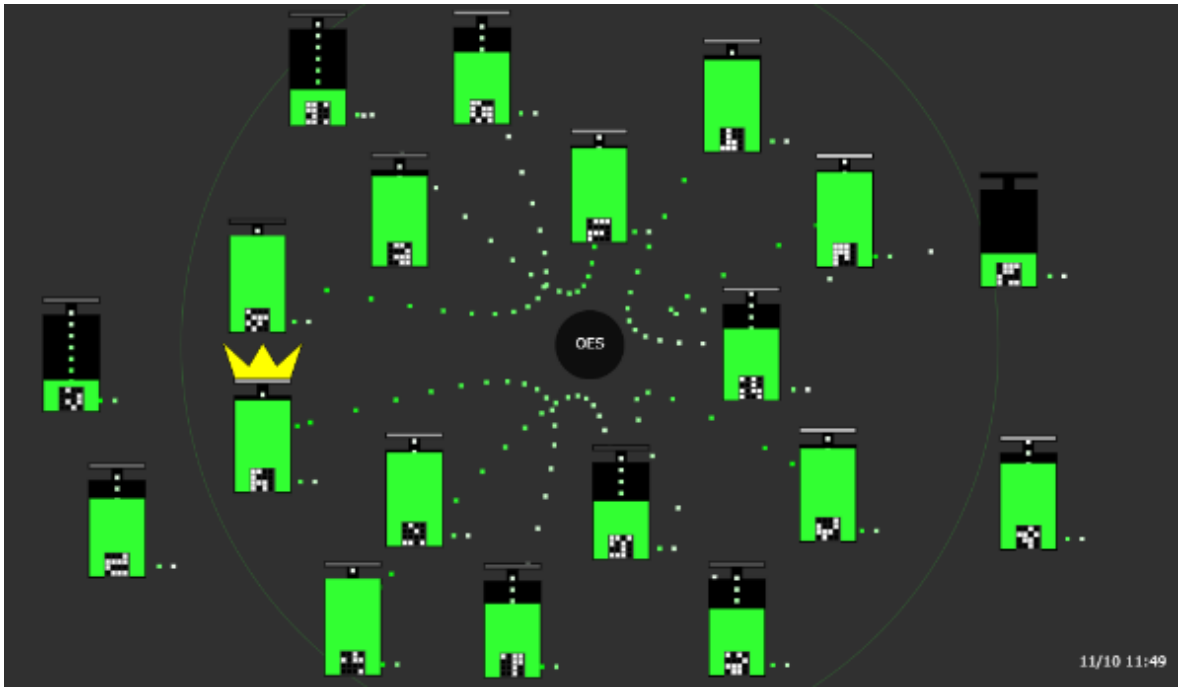


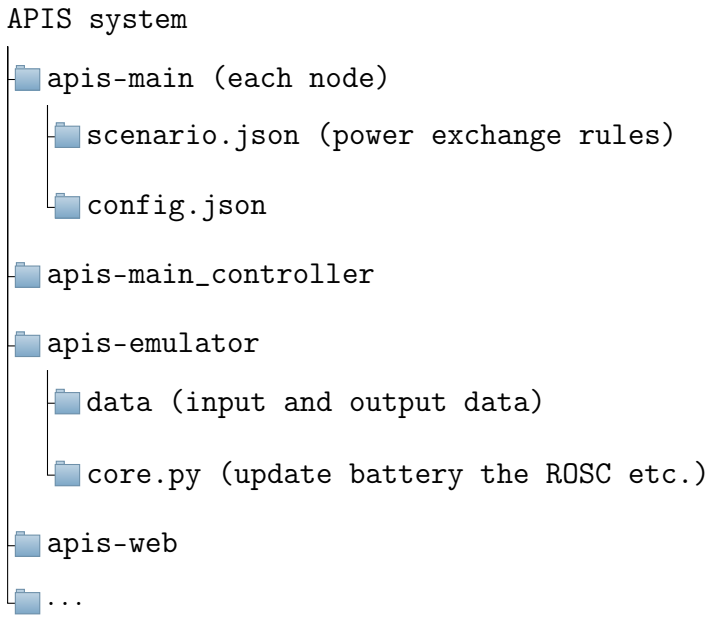
Figure 2.10: Real-time battery monitoring of each residence

I will use the APIS for energy-sharing experiments, i.e. multi-agent system energy management. A detailed description of how the APIS works is illustrated below. I will present the work done with this emulator with RL methods in Chapter 4.

The APIS simulator contains several sub-modules for the overall simulation. It can realize physical peer-to-peer (PP2P) energy sharing as well as autonomous distributed control (Werth et al. [65]). It includes an *apis-main* folder, which is installed for each node to provide a bi-direction energy exchange. The power exchange policy file (**scenario.json**) and each node *config* file are under this software. In addition, APIS has an emulator thread called *apis-emulator*, and we can read all the real-time data

of nodes from the web-API of this emulator. Input and output data from the nodes are also under this thread, and the discharge/charge rules for the battery are also implemented with this emulator.

The software that realizes these technologies is organized in the following structure.



2.5.1 Energy exchange based on scenario files

The energy exchange rule for each house is defined by **scenario.json** files. The structure of an example of the scenario file from 0 to 1 o'clock is shown as follows. A sample *.json* file is given in Appendix B.

A scenario file is defined independently for each node and re-read periodically. In each time period, the battery status is classified into one of four levels: *Excess*, *Sufficient*, *Scare*, and *Short* based on the thresholds specified in the scenario file. The deal negotiation is carried out after each node compares its own battery RSOC with the scenario information. A node in *Excess* status sends a Discharge-REQUEST, while a node in *Scare* status sends a Charge-REQUEST to all nodes in the cluster. A node in *Excess* or *Sufficient* status sends Discharge-ACCEPT, while a node in *Sufficient* or *Scare* status sends a Charge-ACCEPT. Once the deal is made, energy exchange starts between the subsystems for deal execution. Figure 2.11 provides an example of the scenario settings for a node with the maximum battery capacity of 4.8kWh (the same as the real local battery system). In this example scenario, the status is *Excess* when the RSOC level is over 80% (3840Wh - 4800Wh), *Sufficient* when the RSOC level is 60%-80% (2880Wh - 3840Wh), *Scare* when the RSOC level is 40%-60% (1920Wh - 2880Wh), and *Short* when the RSOC level is lower than 40% (0 - 1920Wh).

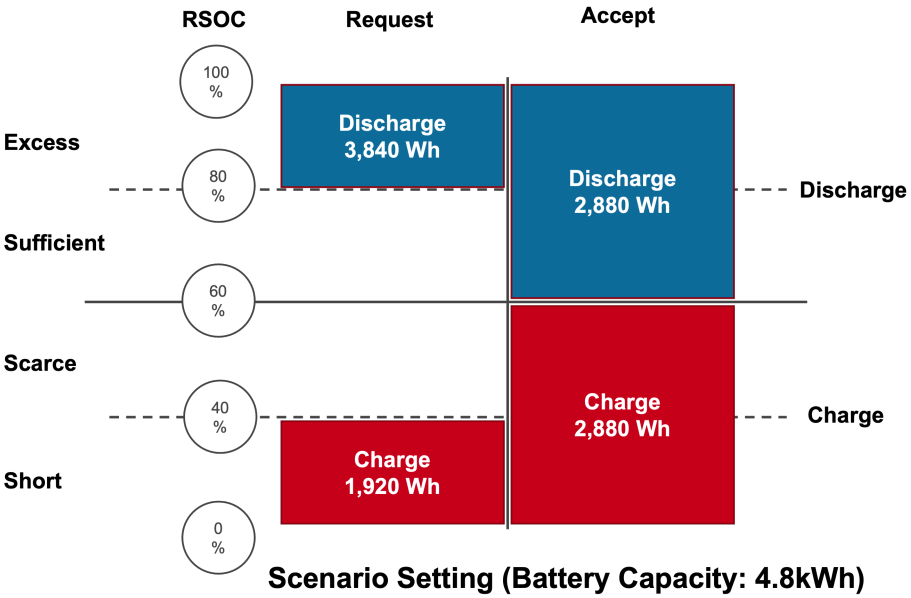


Figure 2.11: Request/Accept threshold in scenarios.

2.5.2 APIS data flow

The simulation data flow of APIS is summarized in Figure 2.12.

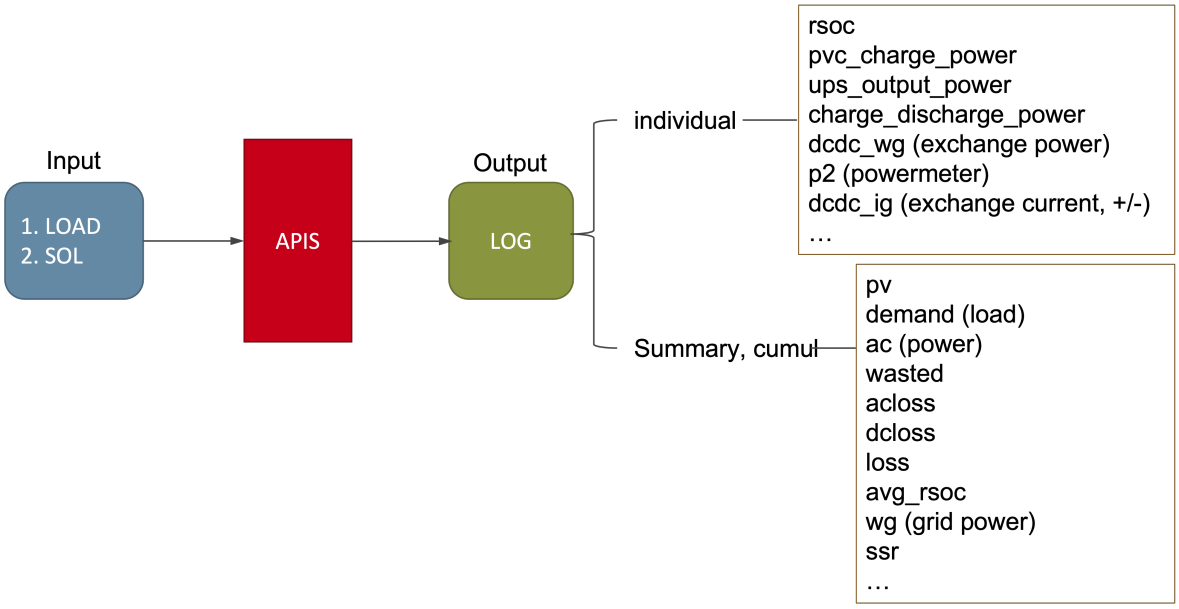


Figure 2.12: Data flow in APIS.

The input data to the APIS is the load and the solar generation power. The solar radiation data is hourly data for 24 hours (unit: W/m^2), and the load data is the consumption power every 30 minutes for 24 hours (unit: kW). We reorganize the OIST houses data in a specific format to feed into the APIS. An example of input data is listed in Appendix C.

The outputs are stored in log files for evaluation as *.csv* files. Figure 2.12 shows a sample output variables we could save. Appendix E lists a result of each house's individual performance data file and a summarized result data file for all houses.

Choice of the acceleration parameter The APIS is a real-time simulator and has an acceleration function. The acceleration rate can be set between 1 and 200, which is used to change the progression of time. This restriction (factor *gl.acc*) results in a limitation of speeding up. If *gl.acc* is set to 30, time in the emulator progresses 30 seconds for each second in the real world. Setting reasonable acceleration parameters is important to speed up all training experiments as the simulator is a real-time simulator. If we do not use acceleration, the time used in the simulator would be the same as the real-time. A detailed acceleration setting and implementation can be found in Appendix D.

2.6 Summary

I explain the essential theoretical knowledge of RL and MARL in this chapter. The DCOES battery component and APIS simulator are also introduced. These provide the groundwork for the following RL implementations.

Chapter 3

Single House Energy Management with Reinforcement Learning

In this chapter, I test the applicability of reinforcement learning to energy optimization of single houses without energy exchanges across houses. I first developed a battery simulator based on a linear battery model for each residence in the community of faculty houses at OIST. The model is verified with a testing data set. I then use this model to test different RL methods for learning with different houses and input states. The results of different cases are discussed.

3.1 PV panel and Battery Settings

There are three routes (route A, B, and C) in the overall DCOES on the OIST campus (Figure 1.7). For the experiments in this chapter, I select route B for all simulations. There are seven houses on this route. PV panels and battery information, such as the PV size and battery capacity, are listed in Table 3.1.

Table 3.1: PV panel and Battery Settings for route B houses in 2018 and 2019

Route	House ID	Total number of PV modules	Total PV Size (kW)	PVC Size (kW)	Battery Size (kWh)
B	205	12	2.88	4	4.8
	206	12	2.88	2	
	208	12	2.88	2	
	212	12	2.796	4	
	213	12	2.88	2	
	214	18	4.32	2	
	215	12	2.88	2	

3.2 Model of battery

Battery models describe the relationship between RSOC, battery voltage, and battery current [9, 10]. I assume a linear relationship between the change in RSOC and battery current. Positive battery current happens when the battery is charged (RSOC rises), negative battery current means the battery is discharged (RSOC declines), and zero battery current means no charge/discharge happens. A decay in RSOC even when there is no charging or discharging should also be considered. Although each house has slightly different battery charge and discharge coefficients, the battery model could be simulated as a simple linear model. A more detailed explanation is given in the following section.

3.2.1 Linear simulation model of the battery

To find the relationship between battery current and RSOC, I visualized the first 30 days of data for house 214 in 2019. Figure 3.1 is the plot of battery current and the change in RSOC (d_rsoc) per minute. By performing linear regression on the data points, we can get the estimated coefficients for the linear regression as the charge/discharge coefficients and the independent term as the decay value in the linear model.

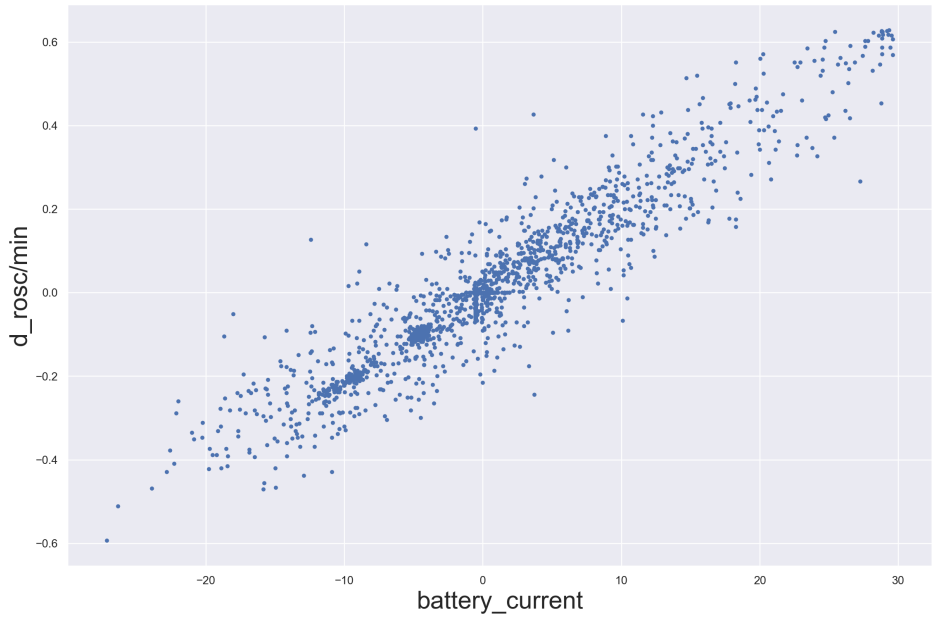


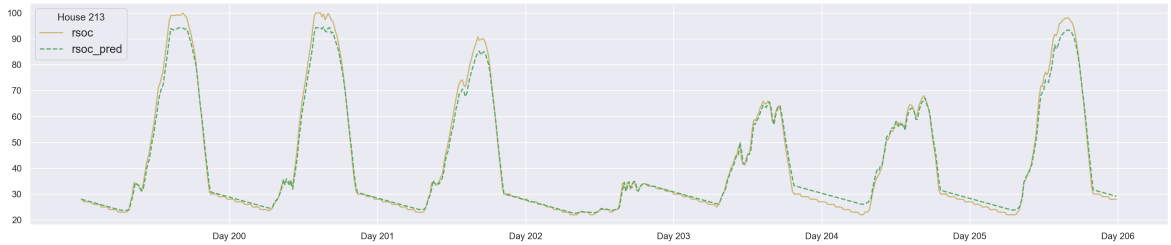
Figure 3.1: Relationship between battery current and change in RSOC, quarter-hour data; house 214, 2019.

Thus, a linear model simulator of the system can be generated from the charge/discharge current of the battery and the relative state of charge (RSOC) of the battery for each

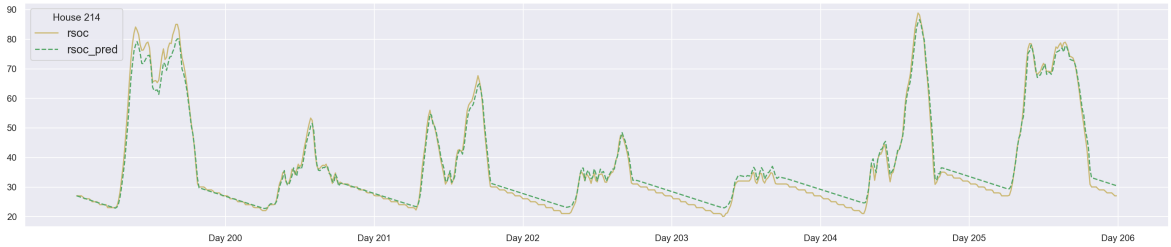
house. The relationship between these variables follows Equation 3.1:

$$RSOC_{t+1} = \begin{cases} RSOC_t + (K_d \times a_t - Decay) \times timestep & \text{if } a_t \text{ is discharge} \\ RSOC_t + (K_c \times a_t - Decay) \times timestep & \text{if } a_t \text{ is charge} \\ RSOC_t - Decay \times timestep & \text{if } a_t \text{ is idle,} \end{cases} \tag{3.1}$$

where a_t is the charge/discharge current and K_d and K_c represent discharge/charge coefficients, respectively. *Decay* is the decaying factor of the battery. *timestep* denotes the time step of the input data points. Training data uses data from the first 200 days, and testing data uses data on different days (after 200 days). Figure 3.2 shows the prediction in house 213 and house 214 by the linear simulator of RSOC. The green dashed line denotes the simulated RSOC from the linear model, while the yellow curve is the real RSOC value. The root mean square error (RMSE) between the predicted RSOC and the real RSOC is 2.11 for house 214 when *timestep* is 15 minutes, while the RMSE is 2.41 for house 213 when the *timestep* is 15 minutes. It indicates that in different houses, the linear model provides a good estimation of the RSOC.



(a) Linear simulator for quarter-hour data sets, house 213, 2019.



(b) Linear simulator for quarter-hour data sets, house 214, 2019.

Figure 3.2: Linear model simulation prediction in different houses.

3.3 Single house RL with the tabular method

Figure 3.3 shows the energy flow in a single house controlled by the ESS. Here I only consider two input energy sources, solar power (PV Charger) and external power (p2). And I am going to use simulated p2 value (purchased power) for all further experiments. After passing the input data to the ESS, the RSOC value could be calculated. Here the load is the usage in the house.

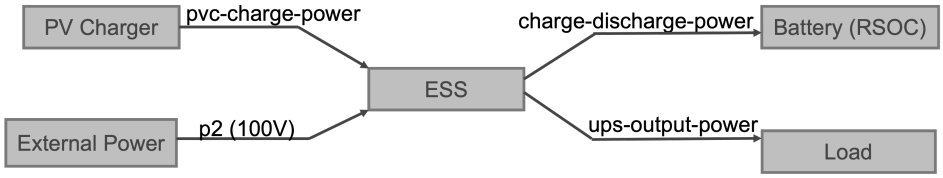


Figure 3.3: Sketch of ESS data flow.

3.3.1 State and Action representation

(1) State representation Based on Figure 3.3, I consider the following representations for **state** variables:

1) how much production is currently received from the PV charger, which is **pvc_charge_power**
2) how much electricity is currently used in the house, which is **ups_output_power**
3) relative status of charge of the battery, which is Battery **rsoc**
4) input power from the utility grid to ESS, which is **p2**

There could be other options of state, such as solar radiation, outside temperature, wind speed, etc. For single-house, **dcdc_grid_power** can be ignored (no exchange of power from other agents). Depending on the charge/discharge decision, each agent ends up buying/not buying electricity (**p2**). State representation for each agent can be formulated from the above variables depending on the model design.

(2) Action selection The action setting in single-agent RL for energy management varies according to different designs. There could be selections from different input energy sources in hybrid MG or charge/discharge the battery of the storage system. To increase the ESS system's self-sufficiency rate (SSR), I can adjust the *charge/discharge* of the battery.

For a reinforcement learning agent, actions can be charge/discharge power to the battery. For Q learning that assumes discrete actions, an action can be chosen directly from a set of battery currents:

$$\mathcal{A} = \{-35, -28, -21, -14, -7, 0, 7, 14, 21, 28, 35\}(\text{unit: Ampere}),$$

where negative value denotes discharging action to the battery, positive value denotes charging action to the battery, and 0 denotes idle action to the battery. The range of action is between an interval of [-35, 35], and different levels of charge/discharge can also be considered.

(3) Reward design The basic reward is set as the power bought from the commercial power line. In addition, avoiding over-charging, over-discharging, or heating of the battery is an important goal. I also set the upper and lower bounds to the reward.

Accordingly, a reward function based on the battery status and charge/discharge/idle operation can be considered, such as

$$r_t^*(a_t) = \begin{cases} k_d p_t^B \Delta t & \text{if } a_t \text{ is discharge} \quad rsoc_{min} < rsoc < rsoc_{max} \\ -nk_d & \text{if } a_t \text{ is discharge} \quad rsoc \leq rsoc_{min} \\ k_c p_t^B \Delta t & \text{if } a_t \text{ is charge} \quad rsoc_{min} < rsoc < rsoc_{max} \\ -nk_c & \text{if } a_t \text{ is charge} \quad rsoc \geq rsoc_{min} \\ 0 & \text{if } a_t \text{ is idle} \quad otherwise \end{cases}$$

$$r_t(a_t) = \begin{cases} r_t^*(a_t) - \eta_t^{buy} p_t^{grid} & \text{if } p_t^{grid} > 0 \\ r_t^*(a_t) + \eta_t^{sell} p_t^{grid} & \text{if } p_t^{grid} < 0, \end{cases}$$

where k_d is the discharging reward factor; k_c is the charging reward factor; $rsoc_{min}$ and $rsoc_{max}$ are the minimum and maximum relative state of charge, respectively; n is the plenty factor. $p_t^{grid} = p_t^{load} - p_t^{PV} + p_t^B$, when $p_t^{grid} > 0$, purchasing electricity from the power line ; when $p_t^{grid} < 0$, agent sells electricity to the power line. η_t^{buy} and η_t^{sell} are buying and selling factors. Currently, buying and selling factors are not considered. Note that reward settings could be different according to different targets.

For simplicity, I set the reward function as the negative value of the net cost, where cost is the house usage and $cost = p2$, and $p2$ is calculated from the battery simulator with $p2 = battery\_charge\_power + load - pvc\_charge\_power$. The overall goal is then to minimize the purchasing power from the power supply line ($r = -p2$).

3.3.2 Tabular Q learning

I first implemented tabular methods with the existing data sets. Similar to the model from [49] and the above design guidelines, I consider the following setup for the model for each agent:

State selection Each agent can use sensory information available from its own house. The most basic sensory state for each house can be the following: $\mathcal{S} = \{s_{pv}, s_{rsoc}, s_{load}\}$, where
s_{pv} denotes photovoltaic power production;
s_{rsoc} denotes the Relative State Of Charge (RSOC) of the battery;
s_{load} denotes power consumption in the house.

The state is formulated from {pv, rsoc, load}, and each input is divided into 4 discrete levels. Thus, I will have $|S_{pv}| \times |S_{rsoc}| \times |S_{load}| = 64$ states. I bin values into discrete intervals. Figure 3.4 shows the discretized values with the histogram bins of states.

$$\text{pv}(W): \begin{cases} 0 & \text{when } pv \in [0., 500.] \\ 1 & \text{when } pv \in (500., 1000.] \\ 2 & \text{when } pv \in (1000., 1500.] \\ 3 & \text{when } pv \in (1500., 2000.] \end{cases}$$

$$\text{load}(W): \begin{cases} 0 & \text{when } consume \in [0., 590.] \\ 1 & \text{when } consume \in (590., 1200.] \\ 2 & \text{when } consume \in (1200., 1800.] \\ 3 & \text{when } consume \in (1800., 2400.] \end{cases}$$

$$\text{rsoc}(\%): \begin{cases} 0 & \text{when } rsoc \in [0., 25.] \\ 1 & \text{when } rsoc \in (25., 50.] \\ 2 & \text{when } rsoc \in (50., 75.] \\ 3 & \text{when } rsoc \in (75., 100.] \end{cases}$$

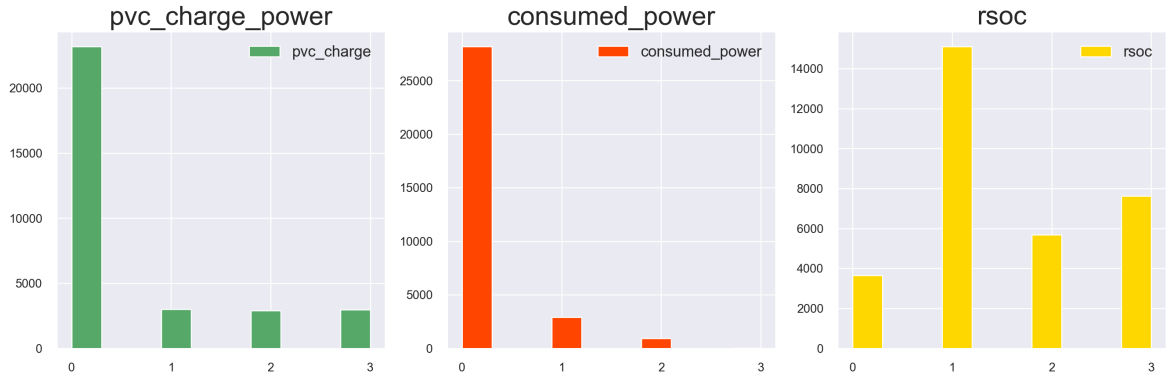


Figure 3.4: Discretized values of states in different bins.

Action design As explained above, I discretize the actions into 11 different levels from [-35, 35] (unit: A), which denotes the battery current. Therefore, action values can be selected from the list: [-35, -28, -21, -14, -7, 0, 7, 14, 21, 28, 35]. Actions are selected with ϵ-greedy method by $a_t \doteq \arg\max_a Q_t(a)$, where $\arg\max_a$ denotes taking action a which maximizes the Q value (referring to Sutton and Barto [54]).

Reward As the goal is to minimize **p2** overtime, where I want to be independent of the commercial energy supply as much as possible, I set the reward function to the

negative value of the purchased power from the AC grid as follows:

$$\begin{aligned} reward &= -cost \\ &= -(-p2_{sim}) \\ &= battery\_charge\_power + load - pvc\_output\_power \\ &= battery\_voltage * action + load - pvc\_output\_power, \end{aligned}$$

where $p2_{sim}$ denotes the simulated p2 value.

I then implement *Q-learning* method with Algorithm 1.

Algorithm 1: Q-learnig (off-policy TD control) for estimating $\pi \approx \pi_*$ (Sutton and Barto [54])

Initialize $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily
set α, γ, where $\alpha \in [0, 1]$ and $\gamma \in [0, 1)$
for *each episode* **do**
  Initialize S
  for *each step of episode* **do**
    select A from S using policy derived from Q (e.g., ϵ-greedy, softmax)
    take action A, observe new states S', and reward R
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
    until Termination Condition
  end
end

I implemented Q-learning method with data from house 214 and 215 in 2019 where $\alpha = 0.2$ and $\gamma = 0.95$. The mean reward of the days of these houses is shown in Figure 3.5 and Figure 3.6, respectively.

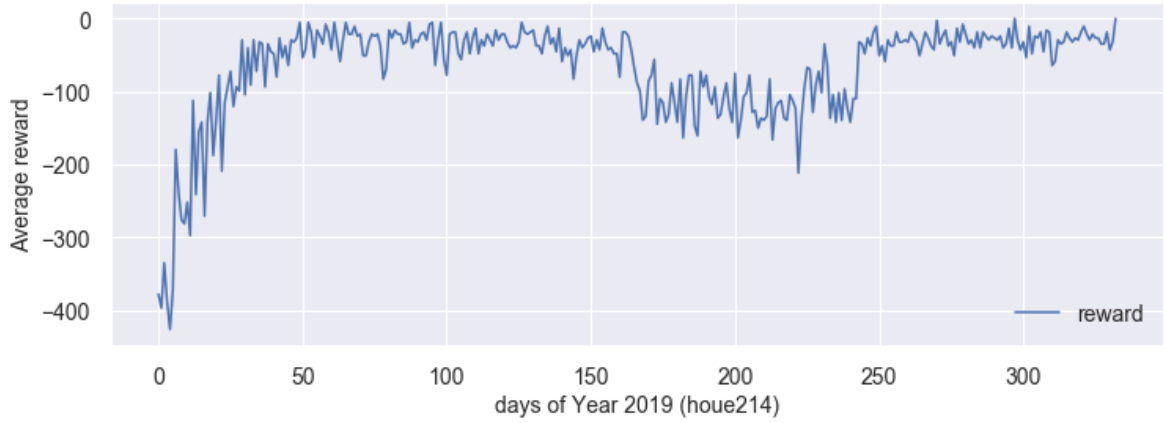


Figure 3.5: Q learning tabular methods, mean rewards of house 214 in 2019.

As inferred from the reward curve, the average reward increases and gradually approaches 0, which means it tries to minimize the purchased power from the external

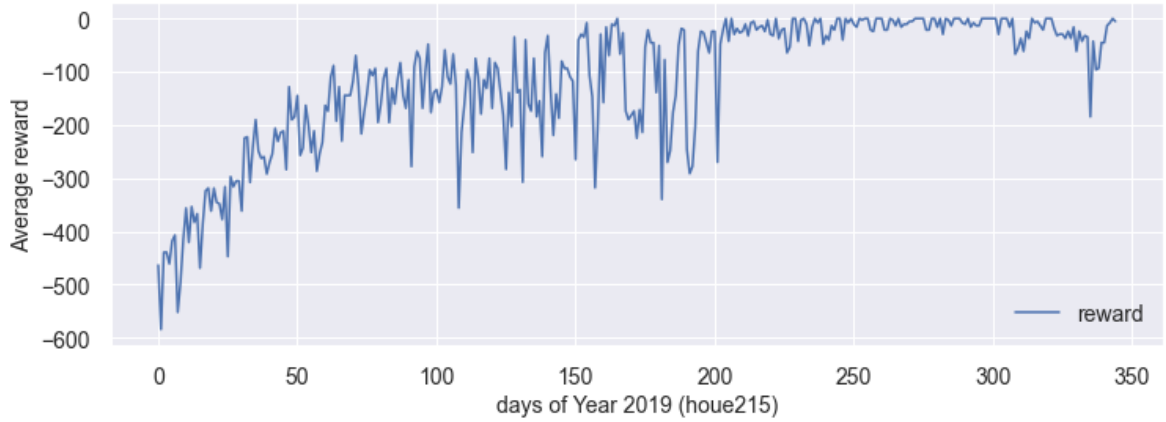


Figure 3.6: Q learning tabular methods, mean rewards of house 215 in 2019.

power supply. I also notice that there are some declines in the reward curve approximately from day 150 to day 240, which is the summer season when energy demand is greater than in other seasons.

3.4 DQN with Prioritized experience replay

3.4.1 Algorithms

The simple tabular method result above indicates that it is feasible to apply RL to single-house energy management. As the data collected in the DCOES have many variables, they are hard to handle by the above look-up table approach. Thus I considered applying deep RL methods. DQN approach follows Algorithm 2 below. As DQN uses random sampling, agents would take a very long time to learn from the replay memory when the reward is scarce. I applied the prioritized DQN method for further learning. As shown in Algorithm 3, in prioritized DQN, agent samples, according to the sample priority in memory instead of random sampling, it can find the learning samples more effectively. p_i denotes the priority of transition i, and importance-sampling (IS) is a Monte Carlo method for evaluating properties of a particular distribution while only having samples generated from a different distribution than the distribution of interest (Kloek and Van Dijk [30]).

3.4.2 States representation in DRL

Following the above state representation, I set the state set from $\mathcal{S} = \{s_{pv}, s_{rsoc}, s_{load}, s_{p2}\}$, where s_{pv} denotes photovoltaic power production, s_{rsoc} denotes RSOC of the battery, s_{load} denotes power consumption in the house and s_{p2} is input power to ESS. I further consider the option of the time of the day information as part of the state.

Encoding cyclic features for states The time of the day information is a cyclic value, and I consider a sin/cos function, which is a standard method for representing continuous cyclic variables. The most common time attributes are months, days, weeks,

Algorithm 2: Deep Q-learning with Experience Replay (Mnih et al. [35, 36])

Initialize replay memory $\mathcal{D}$ to capacity $\mathcal{N}$
Initialize action-value function $\mathcal{Q}$ with random weights θ
Initialize target action-value function $\hat{\mathcal{Q}}$ with weights $\theta^- = \theta$
for *episode = 1,* $\mathcal{M}$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, \mathcal{T}$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a \mathcal{Q}^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \theta_{j+1} \\ r_j + \gamma \max_{a'} \hat{\mathcal{Q}}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - \mathcal{Q}(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Every $\mathcal{C}$ steps reset $\hat{\mathcal{Q}} = \mathcal{Q}$
  end
end

hours, minutes, and seconds which all occur in specific cycles. Other examples might include features such as seasonal, tidal, or astrological data.

I performed a sine and cosine transformation to encode a cyclical feature:

$$x_{sin} = \sin\left(\frac{2 * \pi * x}{\max(x)}\right)$$

$$x_{cos} = \cos\left(\frac{2 * \pi * x}{\max(x)}\right),$$

where x denotes time t. For instance, a quarter-hourly recorded data (96 points per day) could be transformed as following code in python:

```
data['time_sin'] = np.sin(2 * np.pi * data['time']/96.0)
data['time_cos'] = np.cos(2 * np.pi * data['time']/96.0)
```

3.4.3 Simulation results

I implement prioritized DQN for the single-house case under the following conditions:
i) one year data for each house in route B;
ii) two year data for house 214;
iii) one-year data with cyclic features in the state for house 214;
iv) multiple iterations of learning for house 214.

Algorithm 3: DQN with proportional prioritization (Schaul et al. [45])

Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.

Initialize replay memory $\mathcal{H} = \varnothing$, $\Delta = 0$, $p1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ ***to*** T **do**
    Observe S_t, R_t
    Store transition $(S_{t-1}, A_{t-1}, R_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
    if $t \equiv 0$ *mod* K **then**
        for $j = 1$ ***to*** k **do**
            Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
            Compute importance-sampling weight $\omega_j = (N \cdot P(j))^{-\beta} / \max_i \omega_i$
            Compute TD-error $\delta_j = R_j + \gamma_j Q_{target}(S_j, \arg\max_a Q(S_{j-1}, A_{j-1}))$
            Update transition priority $p_j \leftarrow |\delta_j|$
            Accumulate weight-change $\Delta \leftarrow \Delta + \omega_j \cdot \delta_j \cdot \bigtriangledown_\theta Q(S_{j-1}, A_{j-1})$
        end
        Update weights $\theta \leftarrow +\eta \cdot \Delta$, reset $\Delta = 0$
        From time to time copy weights into target network $\theta_{target} \leftarrow \theta$
    end
    Choose action $\mathcal{A}_t \sim \pi_\theta(S_t)$
end

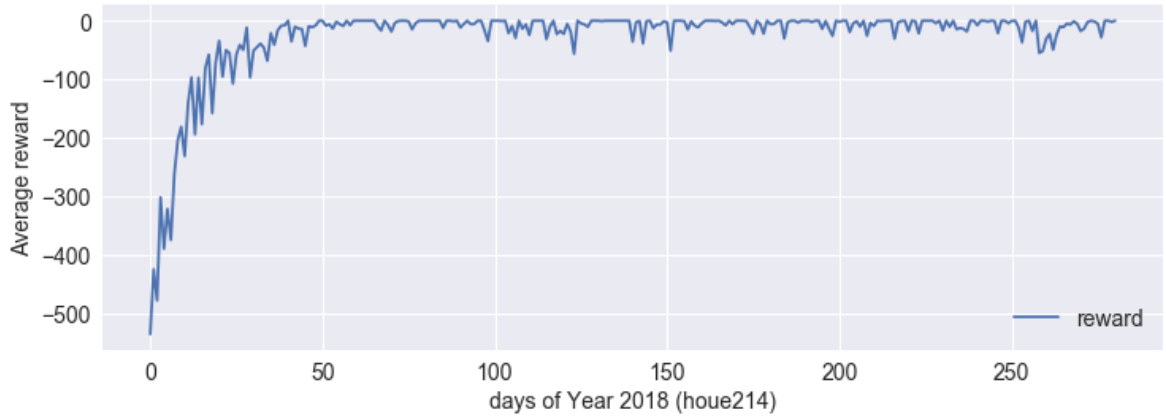
Note that all data are averaged quarter-hour sampled data from the raw data files. The average reward curve in each condition is shown in the following figures.

Yearly data simulation I implement the yearly data for conditions **i)**and **ii)** for houses with prioritized DQN method, where state representation is $\mathcal{S} = \{s_{pv}, s_{rsoc}, s_{load}, s_{p2}\}$. I first implemented data from house 214 in 2018. Figure 3.7a shows the learned average daily reward of house 214 in 2018. As there are only 280 days of data are valid, I further implement data from the year 2019 (Figure 3.7b), as well as a longer time period by concatenating data from both years (Figure 3.7c).

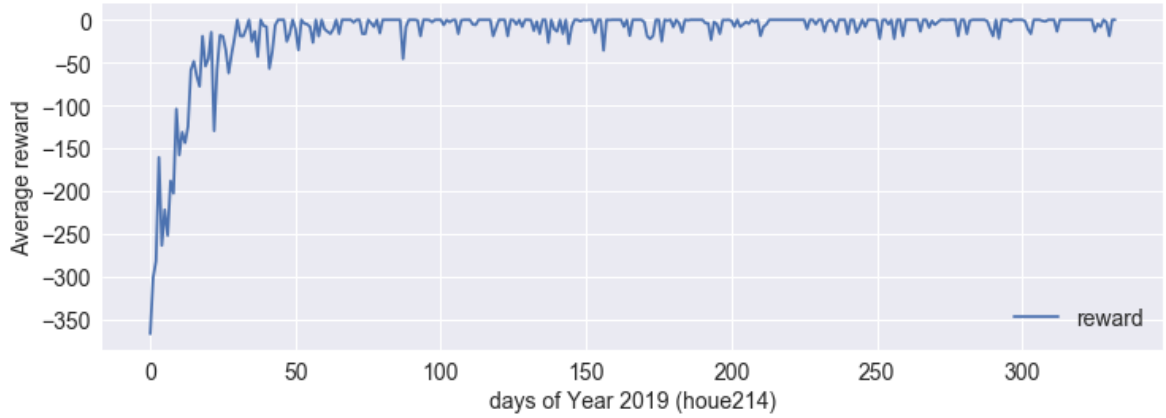
It can be seen from the training results that the daily average reward for houses 214 in 2018 and 2019 with the prioritized DQN method increased and reached 0.

I then implement the same method for all other houses in the same route. The average reward of house 205, house 206, house 208, house 212, house 213, and house 215 are shown in Figure 3.8a, Figure 3.8b, Figure 3.8c, Figure 3.8d, Figure 3.8e, Figure 3.8f, respectively.

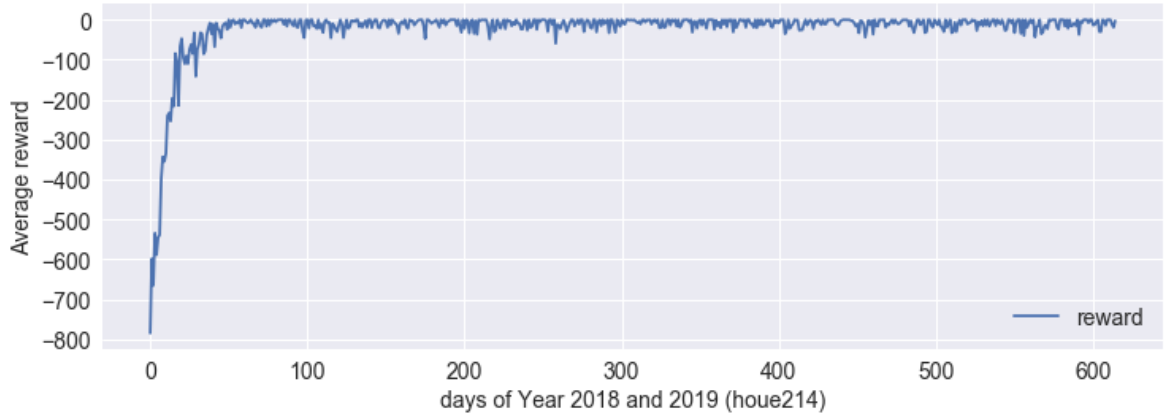
Figure 3.8 indicates that the daily average reward for most houses with the prioritized DQN method increases and reaches 0. Since different houses have different inputs of generated PV power and different usages, performances are also different. In addition, as shown in Figure 3.6 and Figure 3.8f, the prioritized DQN method outperformed the Q-learning method in learning, where the reward reaches 0 earlier and has a higher value with the same dataset.



(a) House 214, 2018



(b) House 214, 2019



(c) House 214, 2018∼2019

Figure 3.7: Mean reward curve of House 214 in different years.

Cyclic features in state I further implement the case **iii)** with cyclic time information. In this situation, state representation is $\mathcal{S} = \{s_{pv}, s_{rsoc}, s_{load}, s_{p2}, s_{time}\}$, where s_{time} uses the sine and cosine transformation I encoded above. The reward in this condition is shown in Figure 3.9.

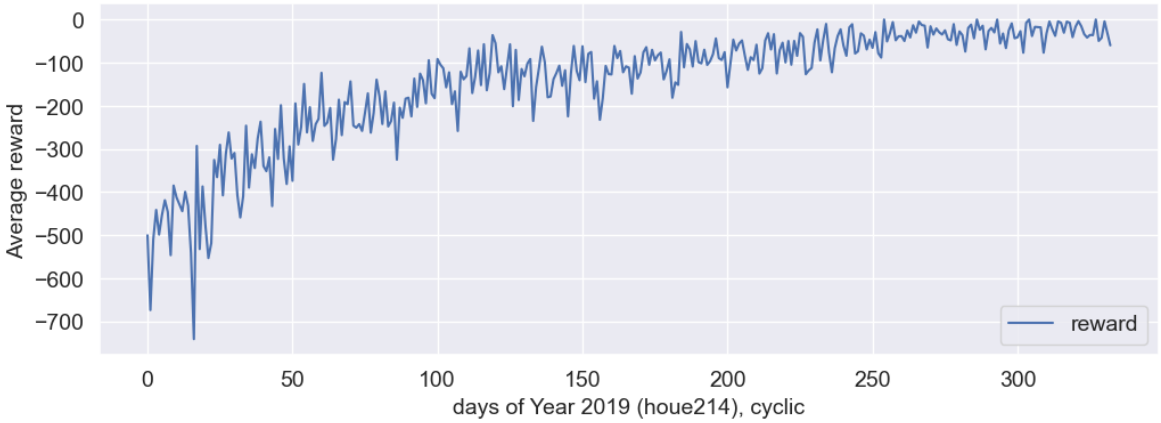


Figure 3.9: Average reward of house 214, with time cycle features, 2019.

Adding cyclic time information to the state has smoother learning in the early days but no significant improvement over the latter days compared with cases without cyclic information.

Multiple iterations Finally, I implement multiple iterations learning (condition **iv)**) in house 214 with $\#EPI = 10$. The total reward over multiple epochs is shown in Figure 3.10.

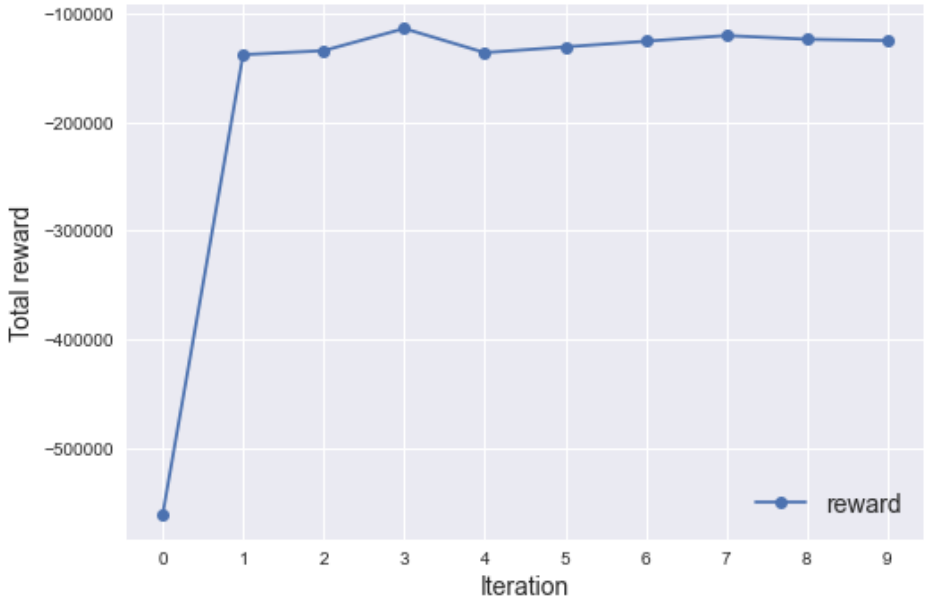
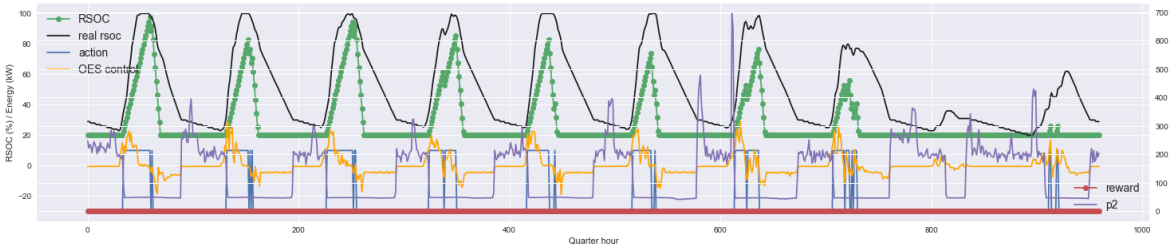


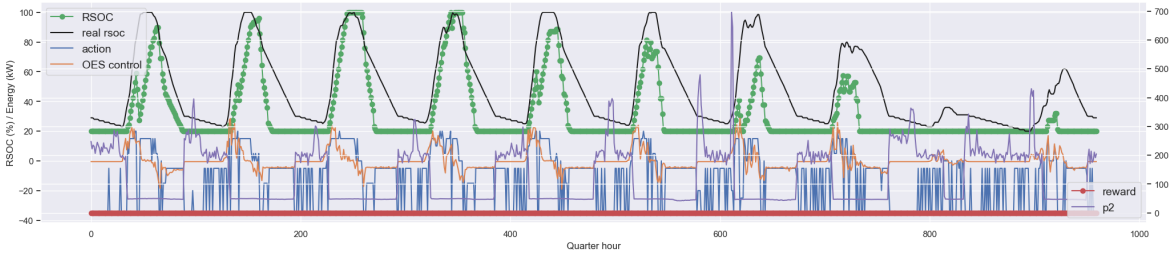
Figure 3.10: Total reward of house 214, multiple iterations, 2019.

The total reward in multiple iterations also indicates that the prioritized DQN approach can improve performance with increased experience.

To illustrate the actual operation of the RL agents, I plot the performance of house 214 in the last 10 days. One run is shown in Figure 3.11a. The cyclical feature's performance for house 214 can be found in Figure 3.11b. The blue line indicates the action of the battery (positive value: charge; negative value: discharge; 0: idle).



(a) Prioritized DQN method, Last 10 days of 2019, house 214.

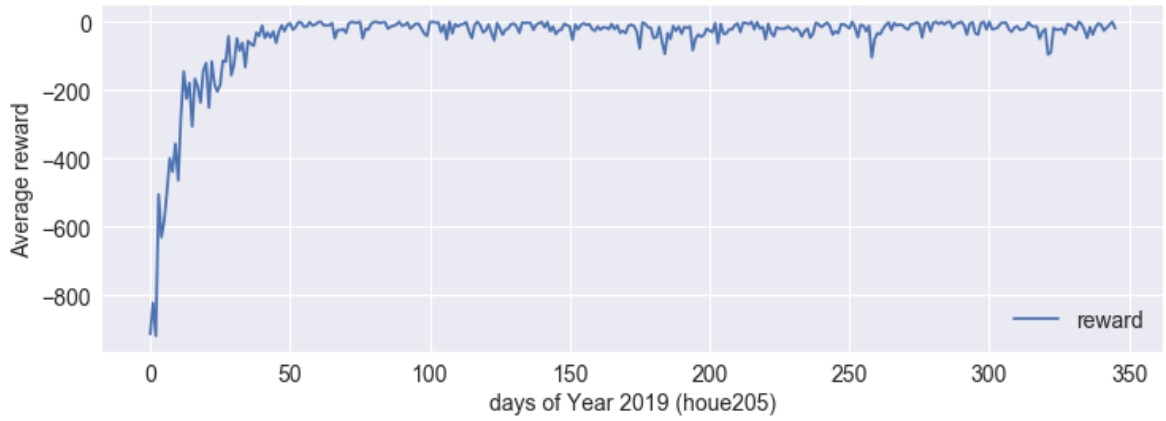


(b) With the cyclical feature, Last 10 days of 2019, house 214.

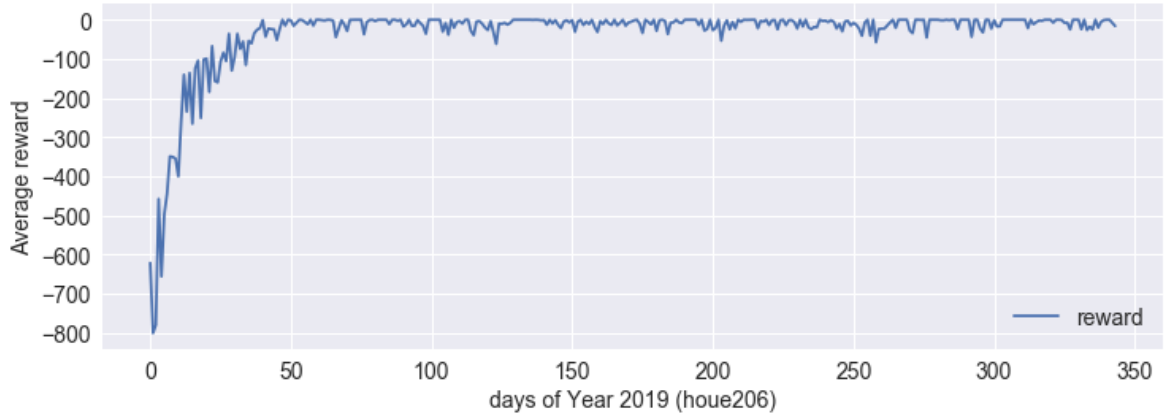
Figure 3.11: Performance in different cases.

3.5 Summary

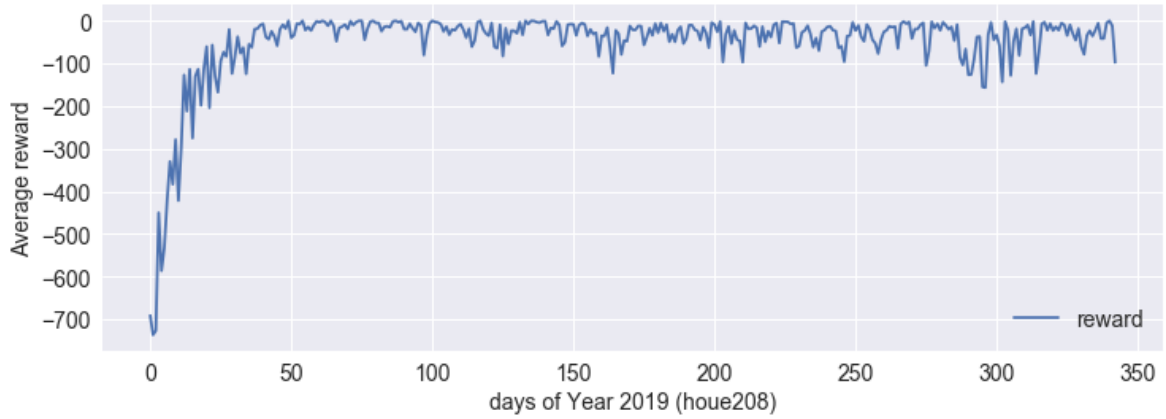
In this chapter, I first proposed a battery model using historical data with linear regression (LR). I confirmed that charge and discharge coefficients and the decay value can be obtained with LR for individual houses with the battery model. Then I evaluated the tabular method, the Q-learning algorithm, for controlling the current to charge/discharge the battery. I discovered that the tabular method is feasible for controlling the current of energy storage. However, using a look-up table purely to store the knowledge has limitations in structuring the state representations. Therefore, I further applied the DRL algorithm to the same problem. By applying the prioritized DQN method in single-house learning, it is verified that learning converges faster than the tabular method. In addition, having time-of-day information in the state is able to reach smoother learning in the early days, but this cyclic value degraded the performance. One possible reason is that daily solar power has a cyclic pattern with respect to solar radiation, while time-of-day is redundant and makes it longer to learn.



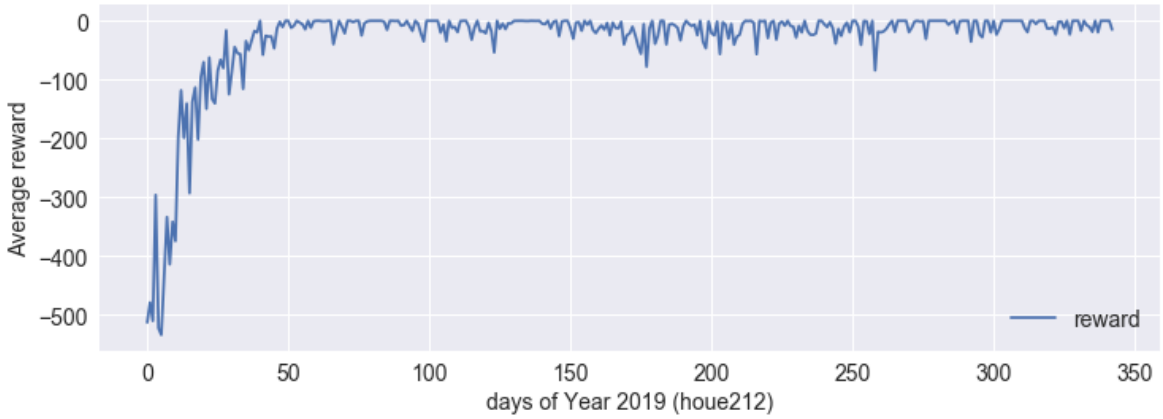
(a) House 205, 2019



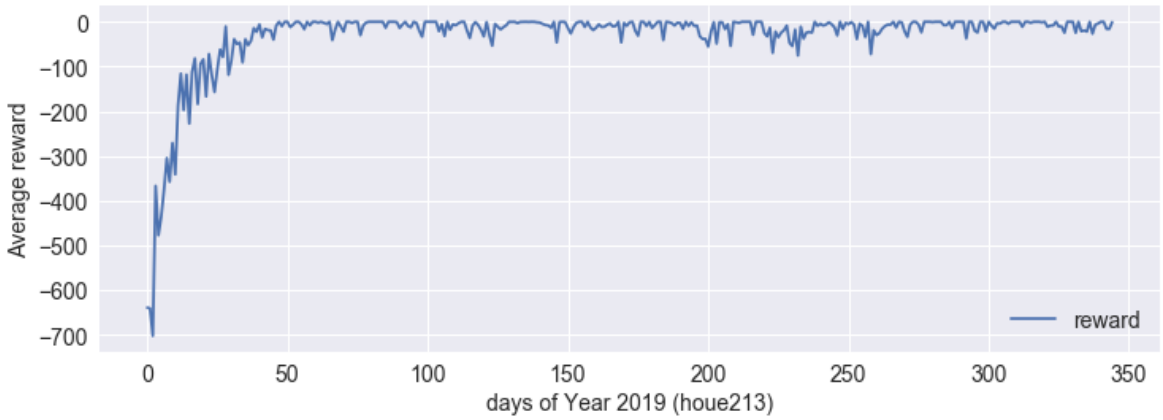
(b) House 206, 2019



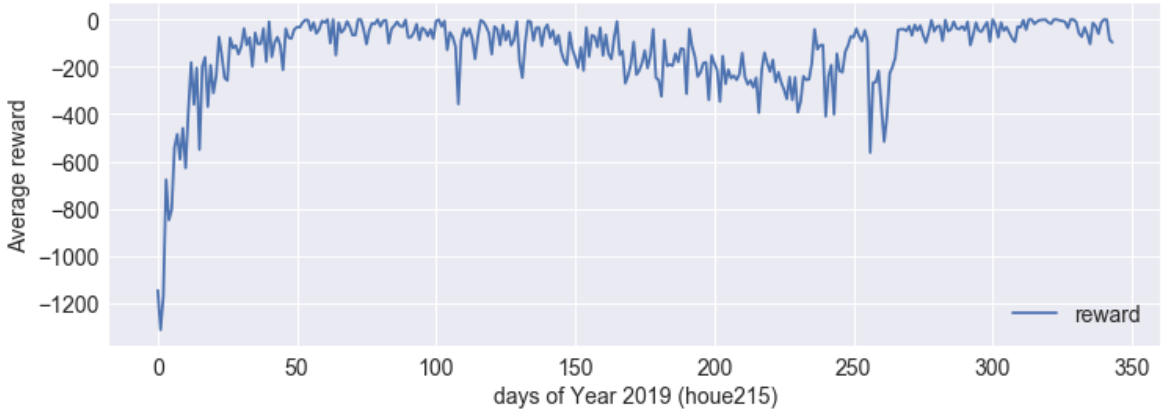
(c) House 208, 2019



(d) House 212, 2019



(e) House 213, 2019



(f) House 215

Figure 3.8: Average reward of different houses in route B.

Chapter 4

Multiple House Energy Management with Reinforcement Learning

This chapter considers how to apply reinforcement learning for the optimization of energy exchanges across multiple houses using the APIS emulator. I first consider possible state and action representations for the energy exchange policies. Different options of state representation with deep Q-network and prioritize-replay methods are proposed and presented.

For multiple houses, rather than adding energy exchanges between different houses to my own linear battery simulator used in the previous chapter, I decided to utilize the open-source APIS software developed by SonyCSL [50].

4.1 Reinforcement learning setup for the APIS

As the APIS is an open-source software simulating real-time energy exchange and battery charge/discharge control, it opens up the possibility of adding whatever higher-level control and optimization rules as desired. Thus, I produced the multi-agent RL system on top of the APIS so that each agent works on each of the node-level controllers.

As discussed in Section 2.5.1, scenario files determine the timing for the energy exchange. Depending on the battery status, it triggers the energy exchange for deal negotiation. However, the default scenario files use fixed values for controlling the exchanges between different nodes. Therefore, it is not flexible enough to adapt to different energy production and usage profiles of different nodes.

The policy for RL agents for the APIS can have various forms. Similar to the single house case, actions can be direct real-time control of charge/discharge current to the battery as well as exchanging current to the DC grid. However, coordinating charge/discharge decisions across multiple agents requires a real-time arbitration mechanism to avoid overloading or under-loading. As a negotiation mechanism is already implemented using the scenario files in the APIS, I decided to take an indirect, higher-level action of rewriting the scenario files to control the energy exchanges among multiple agents. This allows me to focus on optimizing energy exchanges while charge/discharge control of the battery is based on the RSOC as explained in 2.4.1.

Based on the APIS structure, I created a separate thread **apis-rl** in the main APIS

simulator for implementing reinforcement learning algorithms. The overall structure is then formulated in the following forest structure.

```
APIS system with RL thread
├── apis-main (each node)
│   ├── scenario.json (power exchange rules)
│   └── config.json
├── apis-main_controller
├── apis-emulator
│   ├── data (input and output data)
│   └── core.py (update battery rsoc etc.)
└── apis-rl (read data and send back the learned policy)
```

To produce charge/discharge requests and acceptances adaptively, agents need to select the status of the battery (excess, sufficient, scarce, and short) by changing the threshold values. As each node controller has its own scenario file, each RL agent can dynamically update the rules in its scenario file. In this case, the state for an agent is read from the emulator's log web API, and the action is updated by refreshing the scenario file.

4.2 Action and state representations

Action representation For creating a scenario file, three threshold RSOC values need to be specified for the borders between *Excess*, *Sufficient*, *Scare* and *Short* states, as illustrated in Figure 2.11. Because deep Q-learning assumes discrete actions, I use RSOC thresholds in discrete values in every 10% step from 20% to 90%. A state-action value (Q-value) is updated for each action represented by a set of three RSOC thresholds, namely, action[0] for *Excess* lower bound, action[1] for *Sufficient* lower bound, and action[2] for *Scarse* lower bound, with a constraint of $action[0] > action[1] > action[2]$.

States representation In the multi-house setting, in addition to the state representations similar to the single-house cases, it may also be helpful to take into account the states of other houses and the DC grid. I test three settings of state variables.

1) **Stand alone** $\mathcal{S} = \{s_{pv}, s_{load}, s_{rsoc}, s_{p2}\}$

2) **With Community average** $\mathcal{S} = \{s_{pv}, s_{load}, s_{rsoc}, s_{p2}, s_{rsoc_{ave}}, s_{ig}\}$

3) **With time of the day information** $\mathcal{S} = \{s_{pv}, s_{load}, s_{rsoc}, s_{p2}, s_{rsoc_{ave}}, s_{ig}, s_{time}\}$,

where s_{pv} denotes photovoltaic power production, s_{load} denotes power consumption in the house, s_{rsoc} denotes the RSOC of the battery, s_{p2} is input power purchased from the utility grid, $s_{rsoc_{ave}}$ denotes the community average RSOC, s_{ig} is the exchange grid current in DCDC, and s_{time} is a two-dimensional encoding of daily cyclic with sine and cosine components.

4.3 Reward and evaluation criteria

Reward The reward setting is similar to the single house RL, which is the consumption power of the house. The goal of this learning is to minimize the external purchase power as much as possible. And here, I tried two different reward settings, one using the sum of purchased power of the community and the other one using individually purchased power ($p2_i$ where i denotes agent i). The reward for each agent i is calculated from

$$reward_i = -p2_i,$$

and

$$reward_i = -\sum_i p2_i,$$

respectively.

Exchanged power Another key evaluation criterion is the exchanged power within the community. More exchanging power implies that more energy is shared among different nodes, which allows utilizing any surplus power across houses.

Self-sufficiency rate (SSR) For evaluating the performance of different operation policies, one important indicator is the self-sufficiency rate (SSR), which is the proportion of the locally supplied energy (by PV panels) in the total energy consumption in each house or in the community.

In the DCOES, the SSR value is calculated as [42]

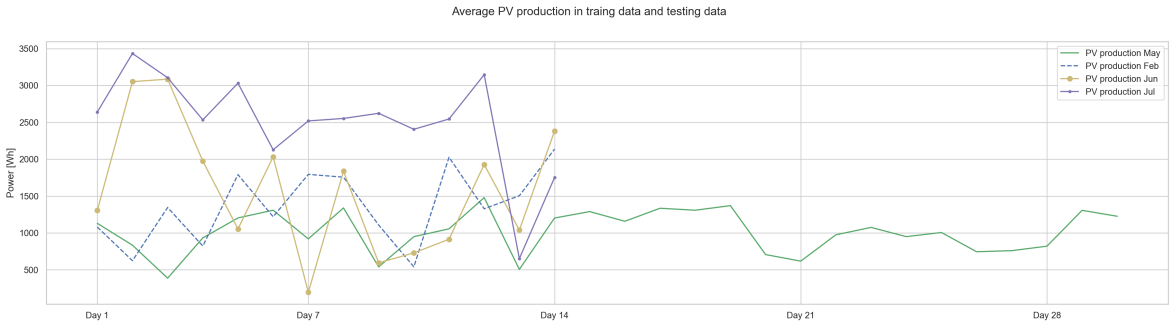
$$SSR = \frac{E_{battery}}{E_{consumption}} \tag{4.1}$$

Here $E_{battery}$ and $E_{consumption}$ are, respectively, the electric power supplied [kWh] during **battery mode**, and the total power consumed [kWh] in both battery mode and bypass mode, as described in Section 2.4.1. The SSR is 100% when all the electric power consumed in the community is supplied by solar power. On the contrary, the rate is 0% when electric power is only supplied by the external power supplier. Here I calculate the SSR value (ssr_pv) without considering the AC loss.

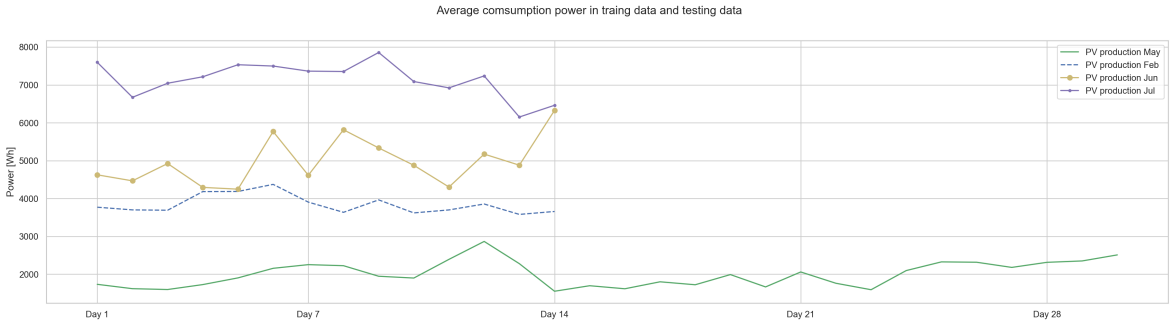
I compared the performances with these three evaluation criteria in the following experiments.

4.4 DCOES dataset in OIST

I prepared both the training and the testing dataset for the different settings for the cases listed above. The training dataset is sliced from May 8 to Jun 6 (30 days) in 2019 for 4 houses (house 212, house 213, house 214, and house 215). While the testing data are from 3 different time periods of the same year. The testing input data are from Feb 2 to Feb 21 (winter season), Jun 14 to Jun 27 (rainy season), and July 21 to Aug 3 (summer season), namely, every two weeks' data is tested with the saved trained model. One reason for only having 30 days of data for training is that the APIS is a real-time simulator with very limited acceleration functions. As I use an acceleration $gl.acc = 60$ (see section 2.5.1 for more details), applying yearly data would take $\tilde{6}.08$ days for one iteration. Therefore, I used monthly data to shorten training hours. Another reason for picking this dataset is that there are some days with missing data in some houses, and on some days, solar generation data is not recorded in some houses in the raw data files. It is necessary to guarantee the selected data have near-complete data points. Figure 4.1 shows a statistics plot of generated power from the PV panel and energy demand from the users in different datasets.



(a) Average PV production power [Wh] per day for 4 houses.



(b) Average consumption power [Wh] per day for 4 houses

Figure 4.1: Statistics of PV production and consumption in different datasets.

Moreover, house IDs are transferred into E001, E002, E003, and E004 for further formatting. A detailed formatted input *sol* (PV generation) and *load* data can be found in Appendix C.

4.5 Choice of action time step and reward settings

The time step is set to decide the update frequency of the learning. I first implement the DQN method with different time steps. I set the time step to 1 hour and 3 hours for the case 1), case 2), and case 3).

Figure 4.2a shows the average purchased power per day with different time steps. Figure 4.2b shows the average exchanged power per day with different time steps. And Figure 4.2c depicts the average SSR value with different timestep. The baseline case with default scenario control is also shown in each condition.

It is clear that with DQN learners, purchased power from the external power line is reduced in both *time step = 1 hour* and *time step = 3* hours cases. Houses have more exchanged power compared with the default scenarios in both cases.

There are no significant differences between *time step* = 1 hour and *time step* = 3 hours cases in the performance. Therefore, I consider using *time step* = 3 hours for all future experiments.

Then I implemented the experiments with reward setting using individual purchased power.

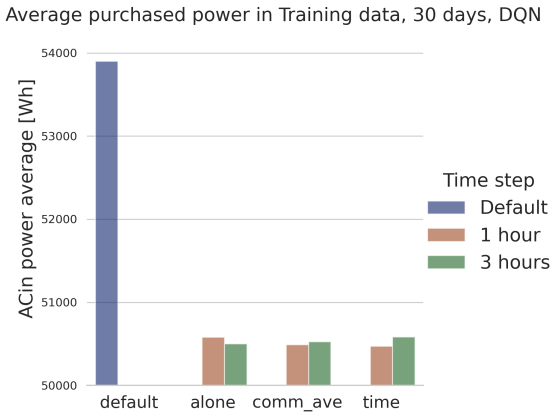
Figure 4.3a shows the average purchased power per day with different reward settings. Figure 4.3b shows the average exchanged power per day with different reward settings. And Figure 4.3c depicts the average SSR value with reward settings. It can be inferred that the case with the sum of the reward in the community slightly outperforms the case with individual reward settings (note that the y-axis in purchased power is not starting from 0). And I am focusing on using the sum reward for the following experiments.

The default fixed scenario for all houses is shown in Figure 4.4a. Meanwhile, actions in the last week of house E001 and E003 learned with the DQN method are further visualized in Figure 4.4b and Figure 4.4c, respectively. Request-CHARGE action is colored in green, Request-DISCHARGE action is colored in the dark blue, and Accept-CHARGE/DISCHARGE are the orange and light blue areas, respectively. RSOC value is shown in the black dot curve. The learned action policy of DQN adjusts with respect to the RSOC level, while the default rule-based control can not realize that.

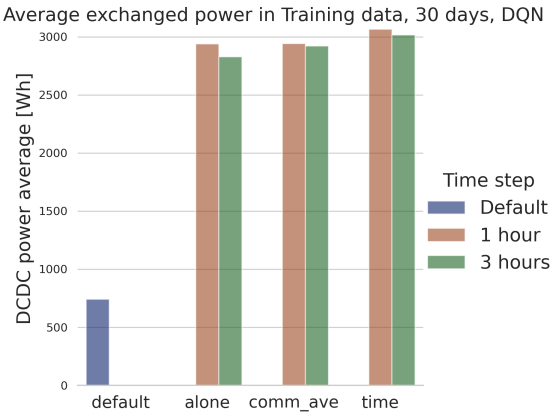
4.6 Comparison of different DRL methods

DQN agents outperformed the fixed rule-based case in all criteria. I further applied prioritized DQN (prior-DQN in short form) with different options of input state in section 4.1. Figure 4.5 shows the performance in comparison of the DQN and the prioritized DQN method, with the default scenario as the baseline. Three conditions are stand-alone, with community average and with time-of-day information, respectively.

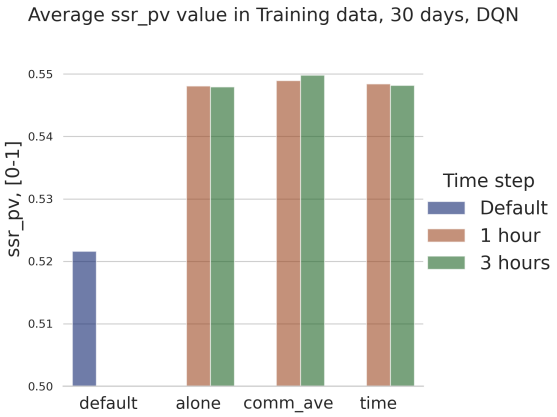
It can be inferred from Figure 4.5 that DRL methods can outperform the default energy exchange control. Purchased energy from the external power line decreases. DRL learners also invoke more shared energy compared with the baseline cases. The SSR values in different conditions are also increased. Moreover, the prioritized DQN agents tend to have less purchased power, greater exchanged power and larger SSR value in all three conditions than the DQN method.



(a) Average purchased power per day, different timestep.

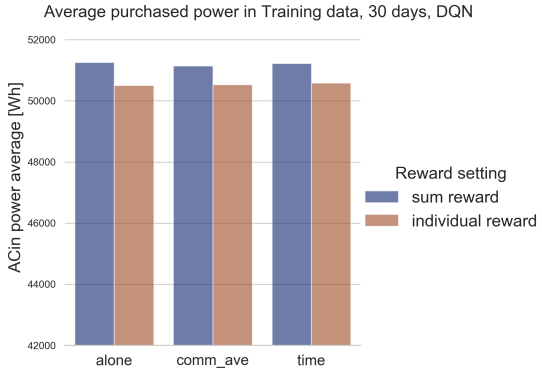


(b) Average exchanged power per day, different timestep.

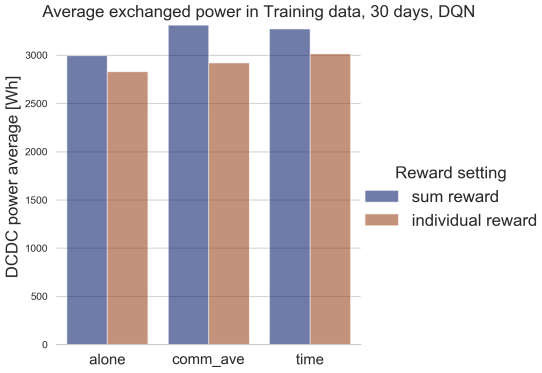


(c) Average SSR, different time step.

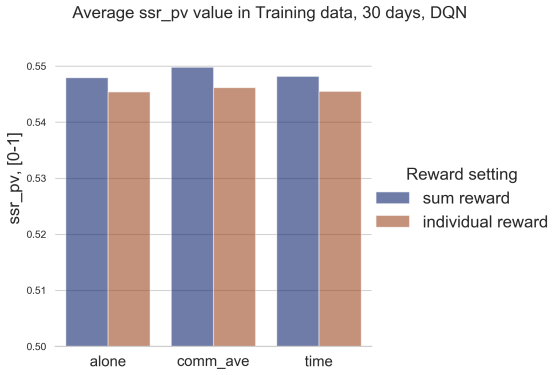
Figure 4.2: Results with different timestep under different conditions; the reward is set to the sum of purchased power.



(a) Purchased power, with different reward settings.

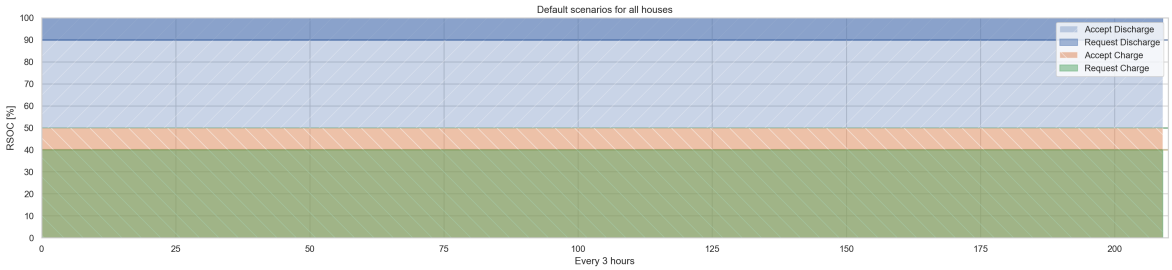


(b) Exchanged power, with different reward settings.

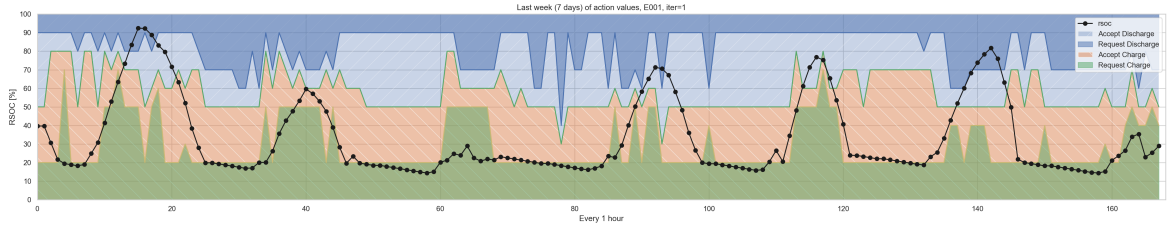


(c) SSR value with different reward settings.

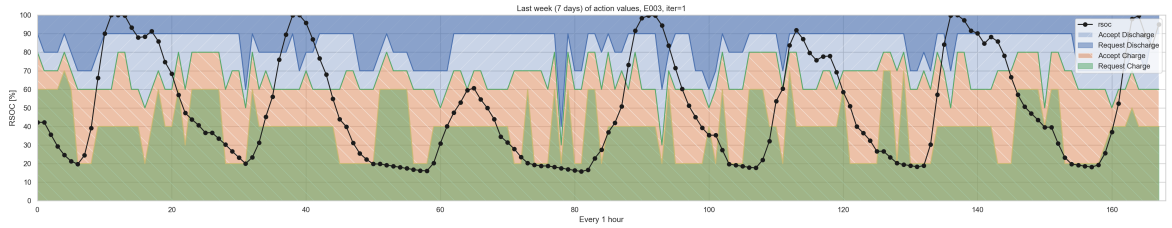
Figure 4.3: Average values in different criteria with different reward settings, DQN.



(a) Default scenarios for houses E001 - E004.



(b) Actions in last 7 days, house E001, timestep=1hr.



(c) Actions in last 7 days, house E003, timestep=1hr.

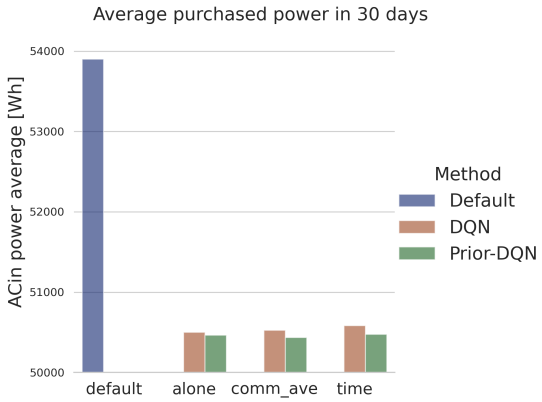
Figure 4.4: Actions and scenarios for houses in different cases.

4.7 Multiple iterations and runs

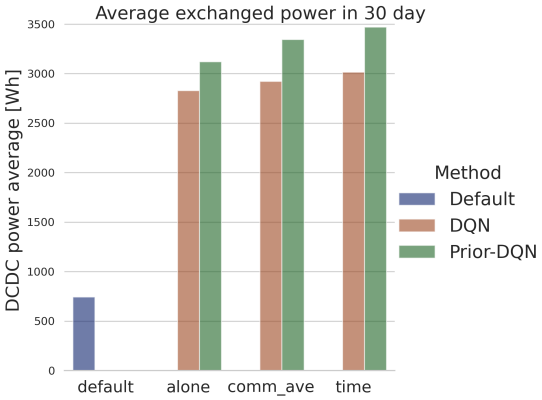
In addition, I repeat multiple iterations under multiple runs with the prioritized DQN method. Figure 4.8a, Figure 4.8b, and Figure 4.8c shows the average purchased power, exchange power, and ssr_pv value with standard deviation in different cases for $run = 3$ and $iter = 3$, respectively.

In multiple iterations experiments (Figure 4.6), agents with community average RSOC information have the least amount of purchased power and the most shared energy on average. While agents with time-of-day information have the largest amount of purchased power. Note that in cases when houses share more energy, it would lead to an increase in purchasing some extra energy as well. For performance on average, cases with community average information performed the best. While the case with time-of-day information has more exchanged power and a larger SSR value than the stand-alone case.

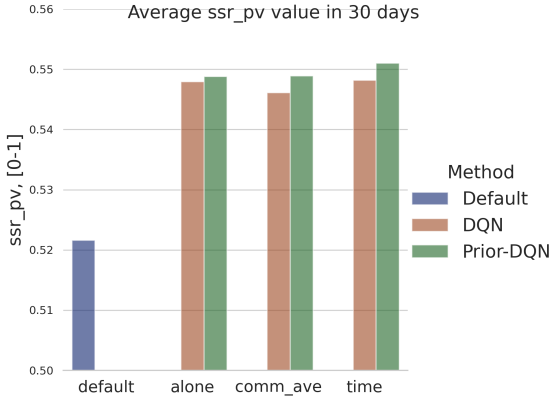
By iterative training with multiple runs and iterations, the case using the community average RSOC information performs best. In this case, the community purchases the least external energy and exchanges the most shared energy, and the SSR value is also the highest of all cases. It can be inferred that the improvements in the performance of multiple runs and iterations in the prioritized DQN method are indeed



(a) Average daily purchased power, different methods.

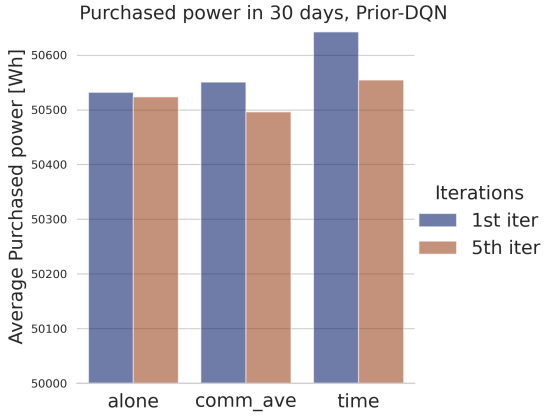


(b) Average daily exchanged power, different methods

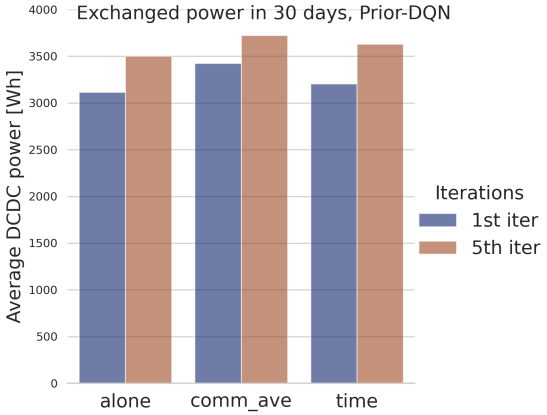


(c) Average SSR, different methods.

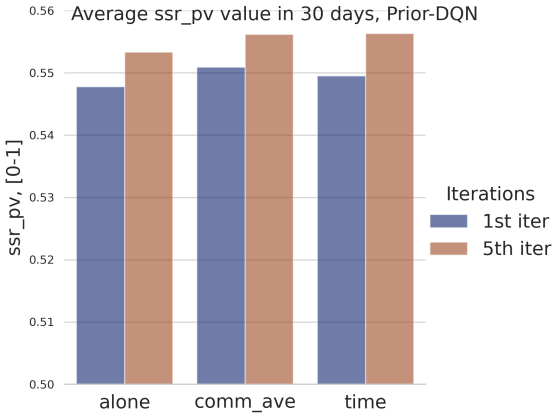
Figure 4.5: Performance of different states options in different methods.



(a) Purchased power, in the 1st and 5th iteration.



(b) Exchanged power, in the 1st and 5th iteration.



(c) SSR value in the 1st and 5th iteration.

Figure 4.6: Average values in different criteria with different iterations, prioritized DQN.

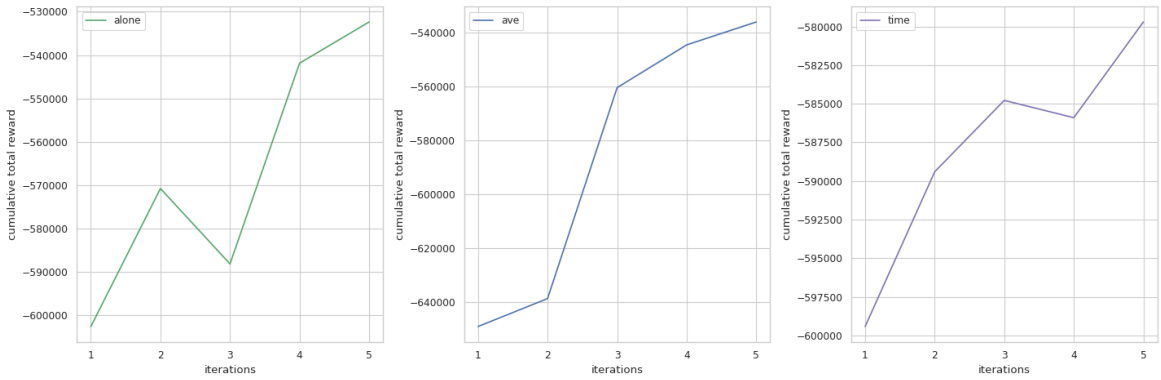
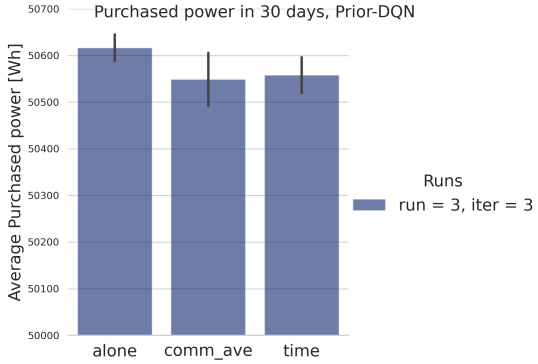
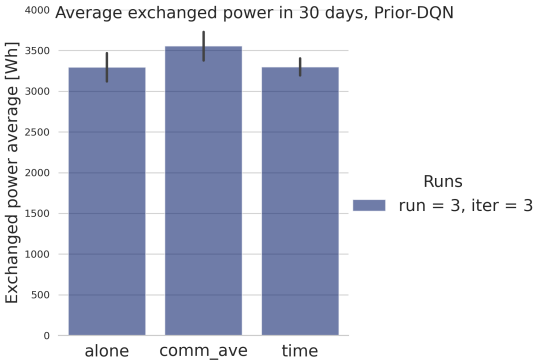


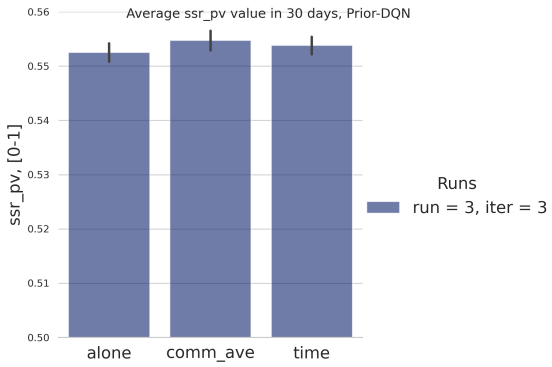
Figure 4.7: Cumulative total reward from all houses with regard to multiple iterations in different cases.



(a) Average purchased power.



(b) Average exchanged power.



(c) Average SSR value.

Figure 4.8: Average values in different criteria in different criteria, prioritized DQN, runs=3 iter=3.

obtained by learning. Increasing the number of iterations further outperforms single-iteration training. The results from multiple runs also indicate a robust improvement in saving energy.

4.8 Generalization across houses and seasons

4.8.1 Shuffled houses ID

For the selected four houses, energy exchange mostly occurs between houses E001 and E003, while house E004 did not participate in exchanging power much compared with other houses. To make sure the input data order will not influence the performance and make a generalized conclusion, I also shuffled house ID order (exchange E001 and E004) to compare the performance.

Table 4.1: Mean, Standard deviation, T-test results in shuffled ID.

Criteria	Purchased power		Exchanged power		SSR	
	normal	shuffled	normal	shuffled	normal	shuffled
mean	50527.13	50489.03	2921.93	2941.63	0.5498	0.5490
standard deviation	11974.59	12080.59	1007.85	1011.76	0.1753	0.1727
t-statistic	0.0121		-0.0743		0.0173	
p-value	0.9904		0.9410		0.9863	

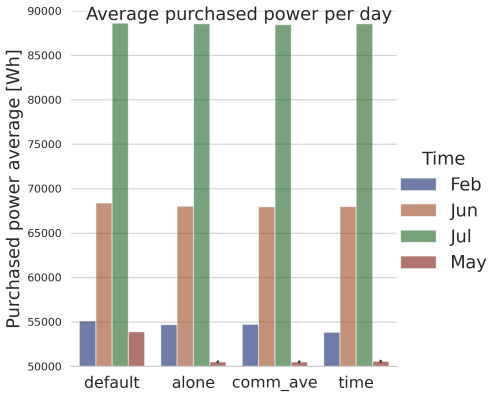
Table 4.1 listed the mean and standard deviation values over 30 days in different criteria in normal order and shuffled the order of the input data set. I also ran paired a simple t-test to compare the performances in these two cases [52]. The t-value measures the size of the difference relative to the variation in the data, and the p-value

is the probability of observing the data with a null hypothesis that they come from the same distribution. The t-values in these two cases of the daily purchased power, the exchanged power and the SSR value are very close to 0, while p-values in the three criteria are close to 1. Therefore, there is no significant difference in the performances when the input order is changed.

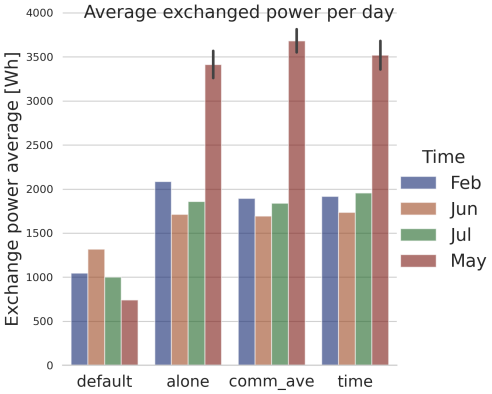
4.8.2 Testing with different time periods of the year

I further use the trained model to test the performance with input data from different time periods of the year. The testing input data are from Feb 2 to Feb 21, Jun 14 to Jun 27, and July 21 to Aug 3. Figure 4.9 shows the average purchased power, the average exchanged power, and the average SSR value per day in all test datasets compared with the trained agent in May.

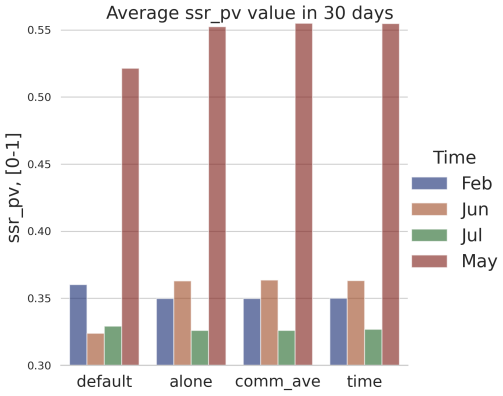
The plots, as shown in Figure 4.9, indicate that the trained model can primarily reduce the purchased power in February. Also, daily exchanged power and SSR values increased the most in February. As the weather and solar generation data and consumption are similar in early May and the test data in February (Figure 4.1), the learned model performs the best among other test datasets. While in July, the testing data had a huge difference compared with the training data (both solar and consumption). In June, most of the dates are rainy, and the daily purchased power decreases slightly compared with their default cases. Although testing data only has two weeks of data points for each duration, the above results suggest that sharing average energy production, storage, and usage within the community helps the performance.



(a) Average purchased power.



(b) Average exchanged power.



(c) Average SSR.

Figure 4.9: Performance of testing data, training data (iter=5), prioritized DQN.

4.9 Summary

In this chapter, I implemented different DRL methods in the APIS simulator for multi-house energy sharing. The physical feature of the APIS makes it impossible to speed up the learning, so it is challenging to apply yearly data to the system. Therefore, I used

monthly data in May for training the agents. Since I put more emphasis on sharing energy among different nodes, I took an indirect, higher-level action to control the exchanging energy performance. I considered both DQN and prioritized DQN algorithms to the APIS. Different updating frequencies and different options of state representations are tested. Training data are tested with stand-alone, community RSOC average, and time-of-day information conditions.

The results indicate that DRL agents significantly outperform fixed rule-based agents. The overall consumption from the utility grid is decreased in all DRL methods, and the total exchanged power gets increased as well. An interesting finding is that in cases when exchanged energy increases, the overall purchased power may also increase (the whole network still outperforms baseline cases). The SSR values of the active agents are also increased compared with default cases. It is important to note that among the above three options of state representations, sharing average energy production, storage, and usage within the community helps the performance. In addition, by shuffling the input data order, I further confirmed that changing order does not impact the improved performance across the houses. This is because energy sharing is carried out sequentially based on the deal information registered in the shared memory of the APIS.

In addition, the results from applying the learned model from training data to different testing datasets in different seasons further indicate that the learned model can also outperform the baseline cases. Another finding is that the learned model performs the best with similar feature inputs. Meanwhile, data from the summer season with considerable differences have less good performance compared with other datasets.

Chapter 5

Conclusion and Open Issues

5.1 Conclusion

An increase in energy consumption is arising with the development of modern technologies and the growing population. How to optimize allocation and to use energy more efficiently is critical for energy management. Previous works discussed some specific aspects of energy allocation, first in a single-agent system with demand-independent electricity prices and a stationary environment. Meanwhile, the multi-agent system in energy management has gained increasing attention in recent years. How to combine efficient use of renewable energy sources with distributed energy storage with intelligent control is an important research concern. The present study was designed to apply RL methods in exploring better utilization of renewable energy sources. As shown in this dissertation, RL methods provide a feasible solution for controlling energy storage. RL learners could learn policies in decision-making to modify their actions with respect to the available energy in both single-agent and multi-agent systems.

In Chapter 1, I illustrated an introduction to the need and necessity for applying artificial intelligence to save energy from a global point of view. As shown in energy statistical review reports, there is a clear trend in increasing renewable energy consumption. Several systematic reviews of energy grid systems and machine learning applications have been undertaken. I also introduce the DCOES system in detail and the algorithms proposed by SonyCSL.

In Chapter 2, I provided the theoretical background of MDP and POMDP to explain how RL works. In addition, I introduced how the deep-Q network is updated with neural networks and experience replay. This method is applied in both single-agent and multi-agent systems in our energy management control. I also explained the APIS, the essential software for energy sharing among houses in the local community. Moreover, I outlined the UPS design in the APIS to verify how the battery operates under different RSOC levels.

In Chapter 3, I first devised a linear battery model for each house. I tested the Q-learning algorithm for battery current control using yearly data in different selected houses. In addition, I applied the prioritized DQN method to the same problem. I also implemented the learning for the battery control by introducing extra time information in the states. Learning curves in different experiments indicate RL methods can reduce the purchased power in a single-agent testing environment.

In Chapter 4, I articulated how I applied DRL methods to the APIS simulator in the multi-agent system. I implemented both DQN and prioritized DQN approaches with different time steps, different reward settings, and multiple iterations, as well as multiple runs. I also performed shuffled house ID in input datasets. Finally, I carried out experiments using testing datasets with the learned training model.

Note since different houses (agents) have different usages (some are heavy electricity users while some are not), the energy sharing among different houses would also vary within different nodes. In the chosen four houses, house 214 is self-sufficient most of the time and outputs extra energy to other nodes when the exchanging policy requirements meet. While house 212 and house 215 are requesting charging from other houses in most deals. I only listed four agents for training while in the physical environment, but there are 19 nodes in total and 7 nodes in sub-route B. In addition, as the APIS does not have a speed-up function, I could currently only test the DQN and prioritized DQN methods. Other MARL methods are not trained. Also, the reward function is designed directly from $p2$, while in a more general case, I could also consider other components in the reward, such as introducing the initial cost of the system or adding the system loss.

In the present thesis, I used historical data collected from the DCOES in the OIST faculty houses' community to explore energy management with RL methods. I investigated the application of RL algorithms in both single-agent battery control problems and multi-agent energy-sharing control problems. I mainly applied Q-learning, DQN, and prioritized DQN approaches in different cases. RL algorithms provide promising methods in the energy management sector. Agents can learn how to adapt in the face of energy source uncertainty attributable to RL techniques. For the single node situation in Chapter 3, the agent can learn to adjust its control of the battery to minimize the purchased power. The simple tabular method is possible to learn charge/discharge actions in minimizing the purchasing power. Since deep Q-learning uses the neural network to update its Q value function. Agents developed the ability to optimize the weights of connections between neurons by introducing experience replay and fixed Q-targets. The prioritized DQN method can outperform the Q-learning method in the battery current control. For the multi-agent system in Chapter 4, agents can also learn to make decisions to use different sharing thresholds to further minimize the overall purchased energy from the AC grid and exchange more energy. One finding from this study is that all DRL methods can reduce the consumed power and increase the SSR value in AC grids. Another finding is that prioritized experience replay can further improve the performance of DQN. Moreover, simulation studies imply that sharing the community's average energy production, storage, and consumption improves performance.

5.2 Open issues

Currently, this thesis work has several open issues for further development.

Different MARL techniques A natural progression of this work is to apply other general MARL methods for comparing the performance. Currently, MARL algorithms

can be mainly divided into two categories. The first category is cooperation-based algorithms, which mainly study how multi-agents learn independently executable strategies through centralized collaborative training. COMA by Foerster et al. [13], MADDPG by Lowe et al. [33] (developed from DDPG in Lillicrap et al. [32] for single agent) are notable methods in cooperation-based type with the centralized critic. The second category uses algorithms based on communication, which further promotes collaboration between agents by establishing communication between them. MAAC by Iqbal and Sha [23], RIAL/DIAL by Foerster et al. [12], etc., are candidate methods.

In addition, we can consider applying other machine learning/deep learning methods for preprocessing the data, such as having a predictive model for further usage and solar generation to help delay the surplus hour the next day.

Hardware connections Another important practical implication is transferring the RL controller to the physical power exchange system. The APIS provides different software which allows the control of actual DC/DC converters and batteries. Current RL algorithms have been widely employed in video games and other game settings (Go, majiang, etc.). It would be very inspiring to apply RL methods to the physical energy management system.

Expanding in local community The distributed topology of the DCOES makes it possible to scale up and propagate in other local communities. In Japan, smart communities such as Fujisawa SST smart town (see its webpage link in [16]) are equipped with solar panels and smart network management. Machine learning controllers can be exported to other urban contexts.

Bibliography

[1] M. Ahrarinouri, M. Rastegar, and A. R. Seifi. Multiagent reinforcement learning for energy management in residential buildings. *IEEE Transactions on Industrial Informatics*, 17(1):659–666, 2020.

[2] K. Arakaki, K. Kuwae, Y. Shimizu, and H. Kitano. A microgrid simulation software for dc-based open energy systems–the dcoes simulator. In *2018 International Conference on Smart Grid (icSmartGrid)*, pages 136–141. IEEE, 2018.

[3] R. F. Arritt and R. C. Dugan. Distribution system analysis and the future smart grid. *IEEE Transactions on Industry Applications*, 47(6):2343–2350, 2011.

[4] BP p.l.c. Statistical review of world energy. `http://bp.com/statisticalreview`, 2021.

[5] S. Bruno, S. Lamonaca, M. La Scala, G. Rotondo, and U. Stecchi. Load control through smart-metering on distribution networks. In *2009 IEEE Bucharest PowerTech*, pages 1–8. IEEE, 2009.

[6] T. Chen and W. Su. Local energy trading behavior modeling with deep reinforcement learning. *IEEE access*, 6:62806–62814, 2018.

[7] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[8] C. G. Codemo, T. Erseghe, and A. Zanella. Energy storage optimization strategies for smart grids. In *2013 IEEE International Conference on Communications (ICC)*, pages 4089–4093. IEEE, 2013.

[9] B. K. Das, N. Hoque, S. Mandal, T. K. Pal, and M. A. Raihan. A techno-economic feasibility of a stand-alone hybrid power generation for remote area application in bangladesh. *Energy*, 134:775–788, 2017.

[10] S. Dhundhara, Y. P. Verma, and A. Williams. Techno-economic analysis of the lithium-ion and lead-acid battery in microgrid systems. *Energy Conversion and Management*, 177:122–142, 2018.

[11] G. Dileep. A survey on smart grid technologies and applications. *Renewable energy*, 146:2589–2625, 2020.

[12] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, pages 2137–2145, 2016.

[13] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] E. Foruzan, L.-K. Soh, and S. Asgarpoor. Reinforcement learning approach for optimal distributed energy management in a microgrid. *IEEE Transactions on Power Systems*, 33(5):5749–5758, 2018.

[15] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau. Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning (EWRL 2016)*, 2016.

[16] Fujisawa SST. Fujisawa sustainable smart town. `https://fujisawasst.com/EN/project/`, 2014.

[17] S. Grillo, M. Marinelli, S. Massucco, and F. Silvestro. Optimal management strategy of a battery-based storage system to improve renewable energy integration in distribution networks. *IEEE Transactions on Smart Grid*, 3(2):950–958, 2012.

[18] C. Guan, Y. Wang, X. Lin, S. Nazarian, and M. Pedram. Reinforcement learning-based control of residential energy storage systems for electric bill minimization. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 637–642. IEEE, 2015.

[19] C. Hau, K. K. Radhakrishnan, J. Siu, and S. K. Panda. Reinforcement learning based energy management algorithm for energy trading and contingency reserve application in a microgrid. In *2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, pages 1005–1009. IEEE, 2020.

[20] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[21] Q. Huang and K. Doya. An experimental study of emergence of communication of reinforcement learning agents. In *International Conference on Artificial General Intelligence*, pages 91–100. Springer, 2019.

[22] Q. Huang, E. Uchibe, and K. Doya. Emergence of communication among reinforcement learning agents under coordination environment. In *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 57–58. IEEE, 2016.

[23] S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912*, 2018.

[24] S. Javadi and S. Javadi. Steps to smart grid realization. In *Proceedings of the 4th WSEAS international conference on Computer engineering and applications*, pages 223–228, 2010.

[25] J. J. Justo, F. Mwasilu, J. Lee, and J.-W. Jung. Ac-microgrids versus dc-microgrids with distributed energy resources: A review. *Renewable and sustainable energy reviews*, 24:387–405, 2013.

[26] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[27] A. Karabiber, C. Keles, A. Kaygusuz, and B. B. Alagoz. An approach for the integration of renewable distributed generation in hybrid dc/ac microgrids. *Renewable energy*, 52:251–259, 2013.

[28] D. Kawamoto and G. Rajendiran. A study of battery soc scheduling using machine learning with renewable sources. In *ICML 2021 Workshop on Tackling Climate Change with Machine Learning*, 2021. URL `https://www.climatechange.ai/papers/icml2021/58`.

[29] S. Kim and H. Lim. Reinforcement learning based energy management algorithm for smart energy buildings. *Energies*, 11(8):2010, 2018.

[30] T. Kloek and H. K. Van Dijk. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.

[31] T. Levent, P. Preux, E. Le Pennec, J. Badosa, G. Henri, and Y. Bonnassieux. Energy management for microgrids: a reinforcement learning approach. In *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, pages 1–5. IEEE, 2019.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[33] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[34] C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable markov decision processes. *Journal of artificial intelligence research*, 14:83–103, 2001.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[37] OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

[38] S. Paul, M. S. Rabbani, R. K. Kundu, and S. M. R. Zaman. A review of smart technology (smart grid) and its features. In *2014 1st International Conference on Non Conventional Energy (ICONCE 2014)*, pages 200–203. IEEE, 2014.

[39] Z. Qu, S. Hou, L. Zhu, J. Yan, and S. Xu. The study of smart grid knowledge visualization key technologies. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 12(1):323–333, 2014.

[40] K. Rahbar, J. Xu, and R. Zhang. Real-time energy storage management for renewable integration in microgrid: An off-line optimization approach. *IEEE Transactions on Smart Grid*, 6(1):124–134, 2014.

[41] J. R. Roncero. Integration is key to smart grid management. In *CIRED Seminar 2008: SmartGrids for Distribution*, pages 1–4. IET, 2008.

[42] T. Sakagami, A. Werth, M. Tokoro, Y. Asai, D. Kawamoto, and H. Kitano. Performance of a dc-based microgrid system in okinawa. In *2015 International Conference on Renewable Energy Research and Applications (ICRERA)*, pages 311–316. IEEE, 2015.

[43] T. Sakagami, Y. Asai, and H. Kitano. Simulation to optimize a dc microgrid in okinawa. In *2016 IEEE International Conference on Sustainable Energy Technologies (ICSET)*, pages 214–219. IEEE, 2016.

[44] N. Saxena, B. J. Choi, and R. Lu. Authentication and authorization scheme for various user roles and devices in smart grid. *IEEE transactions on Information forensics and security*, 11(5):907–921, 2015.

[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[46] O. Sigaud and O. Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[47] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587): 484–489, 2016.

[48] S. Singh, M. Singh, and S. C. Kaushik. Feasibility study of an islanded microgrid in rural area consisting of pv, wind, biomass and battery energy storage system. *Energy Conversion and Management*, 128:178–190, 2016.

[49] T. Sogabe, D. B. Malla, S. Takayama, S. Shin, K. Sakamoto, K. Yamaguchi, T. P. Singh, M. Sogabe, T. Hirata, and Y. Okada. Smart grid optimization by deep reinforcement learning over discrete and continuous action space. In *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC)(A Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU PVSEC)*, pages 3794–3796. IEEE, 2018.

[50] Sony Computer Science Laboratories, Inc. Autonomous power interchange system. `https://github.com/SonyCSL/APIS`, 2021.

[51] B. Spasova, D. Kawamoto, and Y. Takefuji. Energy exchange strategy for local energy markets with heterogenous renewable sources. In *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pages 1–6. IEEE, 2018.

[52] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

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

[54] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[55] C. Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[56] M. Tokoro. Sony csl-oist dc-based open energy system (dcoes). In *Proc. 1st Int. Symp. Open Energy Syst*, pages 64–67, 2014.

[57] M. Tokoro. Dcoes: Dc-based bottom-up energy exchange system for community grid. In *2nd International Symposium on Open Energy Systems*, pages 22–29, 2015.

[58] N. Tomin, A. Zhukov, and A. Domyshev. Deep reinforcement learning for energy microgrids management considering flexible energy sources. In *EPJ Web of Conferences*, volume 217, page 01016. EDP Sciences, 2019.

[59] A. Trivedi, H. C. Aih, and D. Srinivasan. A stochastic cost–benefit analysis framework for allocating energy storage system in distribution network for load leveling. *Applied Energy*, 280:115944, 2020.

[60] J. R. Vázquez-Canteli and Z. Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied energy*, 235:1072–1089, 2019.

[61] J. R. Vázquez-Canteli, J. Kämpf, G. Henze, and Z. Nagy. Citylearn v1. 0: An openai gym environment for demand response with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 356–357, 2019.

[62] J. R. Vázquez-Canteli, S. Dey, G. Henze, and Z. Nagy. Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*, 2020.

[63] J. R. Vazquez-Canteli, G. Henze, and Z. Nagy. Marlisa: Multi-agent reinforcement learning with iterative sequential action selection for load shaping of grid-interactive connected buildings. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 170–179, 2020.

[64] A. Werth, N. Kitamura, and K. Tanaka. Conceptual study for open energy systems: distributed energy network using interconnected dc nanogrids. *IEEE Transactions on Smart Grid*, 6(4):1621–1630, 2015.

[65] A. Werth, A. André, D. Kawamoto, T. Morita, S. Tajima, M. Tokoro, D. Yanagidaira, and K. Tanaka. Peer-to-peer control system for dc microgrids. *IEEE Transactions on Smart Grid*, 9(4):3667–3675, 2016.

[66] A. Werth, N. Kitamura, M. Tokoro, and K. Tanaka. Evaluation model for multi-microgrid with autonomous dc energy exchange. *IEEJ Transactions on Electrical and Electronic Engineering*, 12(5):676–682, 2017.

[67] Y. Zhang, N. Gatsis, and G. B. Giannakis. Robust energy management for microgrids with high-penetration renewables. *IEEE transactions on sustainable energy*, 4(4):944–953, 2013.

[68] S. Zhou, Z. Hu, W. Gu, M. Jiang, and X.-P. Zhang. Artificial intelligence based smart energy community management: A reinforcement learning approach. *CSEE Journal of Power and Energy Systems*, 5(1):1–10, 2019.

Appendix A

Historical raw data

Table A.1 shows an example of historical log data of the weather station in transposed format from the original file (14 data points).
Filename: weather_20190501.csv

Table A.1: Raw data for one day from the weather station.

timestamp	2019/5/1 00:00:21	2019/5/1 00:00:51	
barometer	1008.5	1008.6	
inside_temperature	25.3	25.3	
outside_temperature	25.8	25.7	
wind_speed	2.2	2.7	
wind_direction	194	216	
rain_rate	0	0	...
storm_rain	0	0	
storm_start_date	-1	-1	
solar_radiation	0	0	
inside_humidity	52	52	
outside_humidity	91	91	
forecast_icons	3	3	
forecast_rule_number	192	192	

Table A.2 shows an example of historical log data of one house in transposed format from the original file (31 data points).
Filename: houseID_20190501.csv

Table A.2: Raw data for one day from one house.

timestamp	2019/5/1 00:00:21	2019/5/1 00:00:51	
charge_discharge_power	29	29	
rsoc	25	25	
pvc_charge_power	0	0	
pvc_charge_voltage	0.95	0.95	
pvc_charge_current	0	0	
pvc_alarm	0	0	
battery_rsoc	31	31	
battery_voltage	51.9	51.8	
battery_current	-0.56	-0.56	
ups_input_voltage	103.6	103.7	
ups_output_power	1709	1716	
ups_output_voltage	103.6	103.7	
ups_output_current	17.1	15.9	
ups_output_frequency	59	60	
ups_parameter	80	80	...
ups_mode	5	5	
ups_stop_mode	2	2	
ups_operation_schedule	1	1	
alarm	0x4010 0x0080 0x0000	0x4010 0x0080 0x0000	
dischargeable_time	0	0	
dcdc_grid_power	-1.22	-1.22	
dcdc_grid_voltage	16.1	16.12	
dcdc_grid_current	-0.02	-0.02	
dcdc_battery_power	-18.31	-17.09	
dcdc_battery_voltage	51.83	51.83	
dcdc_battery_current	-0.33	-0.32	
p1	2513.3	2302.6	
p2	1910.7	1831.5	
dcdc_temperature	37.62	37.65	
dcdc_status	0x0000	0x0000	

Appendix B

Scenario file

An example of the power interchange scenario file (JSON) to be refreshed in the APIS simulator is shown below.

Filename: scenario.json

```
{
"refreshingPeriodMsec": 5000,
"acceptSelection": {
    "strategy": "pointAndAmount"
},
"00:00:00-24:00:00": {
    "batteryStatus": {
        "4320.0-": "excess",
        "3840.0-4320.0": "sufficient",
        "3360.0-3840.0": "scarce",
        "-3360.0": "short"
    },
    "request": {
        "excess": {
            "discharge": {
                "limitWh": 4320.0,
                "pointPerWh": 10
            }
        },
        "sufficient": {},
        "scarce": {},
        "short": {
            "charge": {
                "limitWh": 3360.0,
                "pointPerWh": 10
            }
        }
    },
    "accept": {
        "excess": {
            "discharge": {
                "limitWh": 3840.0,
                "pointPerWh": 10
            }
        },
        "sufficient": {
            "discharge": {
                "limitWh": 3840.0,
                "pointPerWh": 10
            }
```

```
        },
        "scarce": {
            "charge": {
                "limitWh": 3840.0,
                "pointPerWh": 10
            }
        },
        "short": {
            "charge": {
                "limitWh": 3840.0,
                "pointPerWh": 10
            }
        }
    }
}
}
```

Appendix C

Formated input data

Input data for DCOES in APIS (OIST data) in the corresponding format.

Solar data: For each row, the value is the solar power for every 30 seconds, beginning at 0:00 for 24 hours (a set of 2880 data points). [unit: W]

Filename: fourhouses_2019_apis_sol_Jul.csv

Table C.1: Solar data for 4 houses in Jul, 2019.

houseID	date	00:00:00	00:00:30	...	12:00:00	12:00:30	...	23:59:00	23:59:30
1	2019/7/21	0	0		1178	400		0	0
1	2019/7/22	0	0		832	831		0	0
...					...			...	
4	2019/7/21	0	0		1187	1169		0	0
4	2019/7/22	0	0		1852	1866		0	0

Load data: For each row, the values denote residence power consumption every 30 seconds for 24 hours, beginning at 0:00 (a set of 2880 data points). Data sets for n (n=4) households continue in the column direction. [unit: W]

Filename: fourhouses_2019_apis_load_Jul.csv

Table C.2: Load data for 4 houses in Jul, 2019.

houseID	date	00:00:00	00:00:30	...	12:00:00	12:00:30	...	23:59:00	23:59:30
1	2019/7/21	913	899		1108	1125		757	736
1	2019/7/22	949	925		1020	1043		1113	1138
...					...			...	
4	2019/7/21	1994	2036		1494	1569		2273	2252
4	2019/7/22	2288	2320		1236	1187		2030	2022

Appendix D

Selection of acceleration

Factor *gl.acc* is used to change the progression of time in the emulation according to the number we set. Figure D.1, Figure D.2, and Figure D.3 show the simulated result of four houses with sample data under different acceleration factors. We select *gl.acc* = *60* for all experiments.

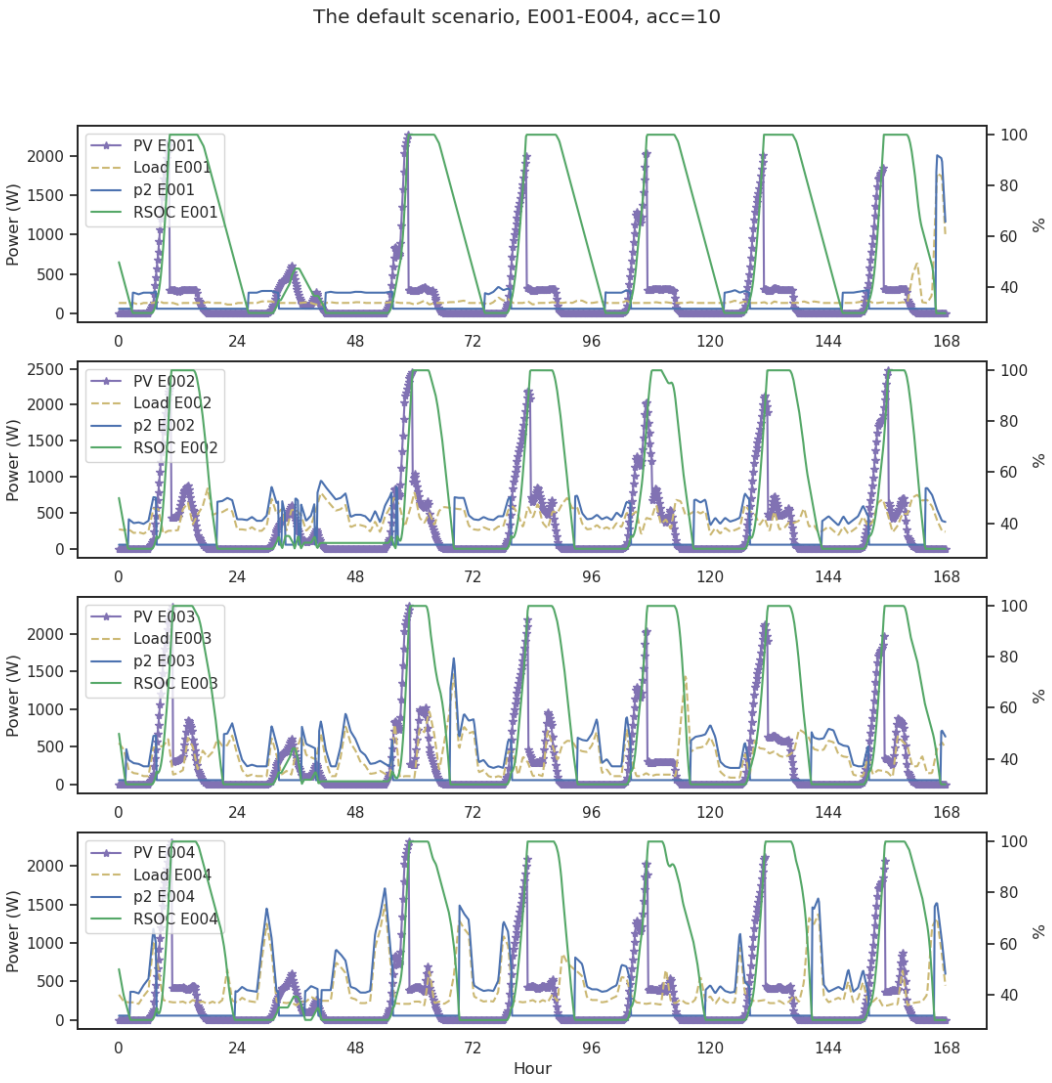


Figure D.1: Simulated data when gl.acc = 10 with sample data.

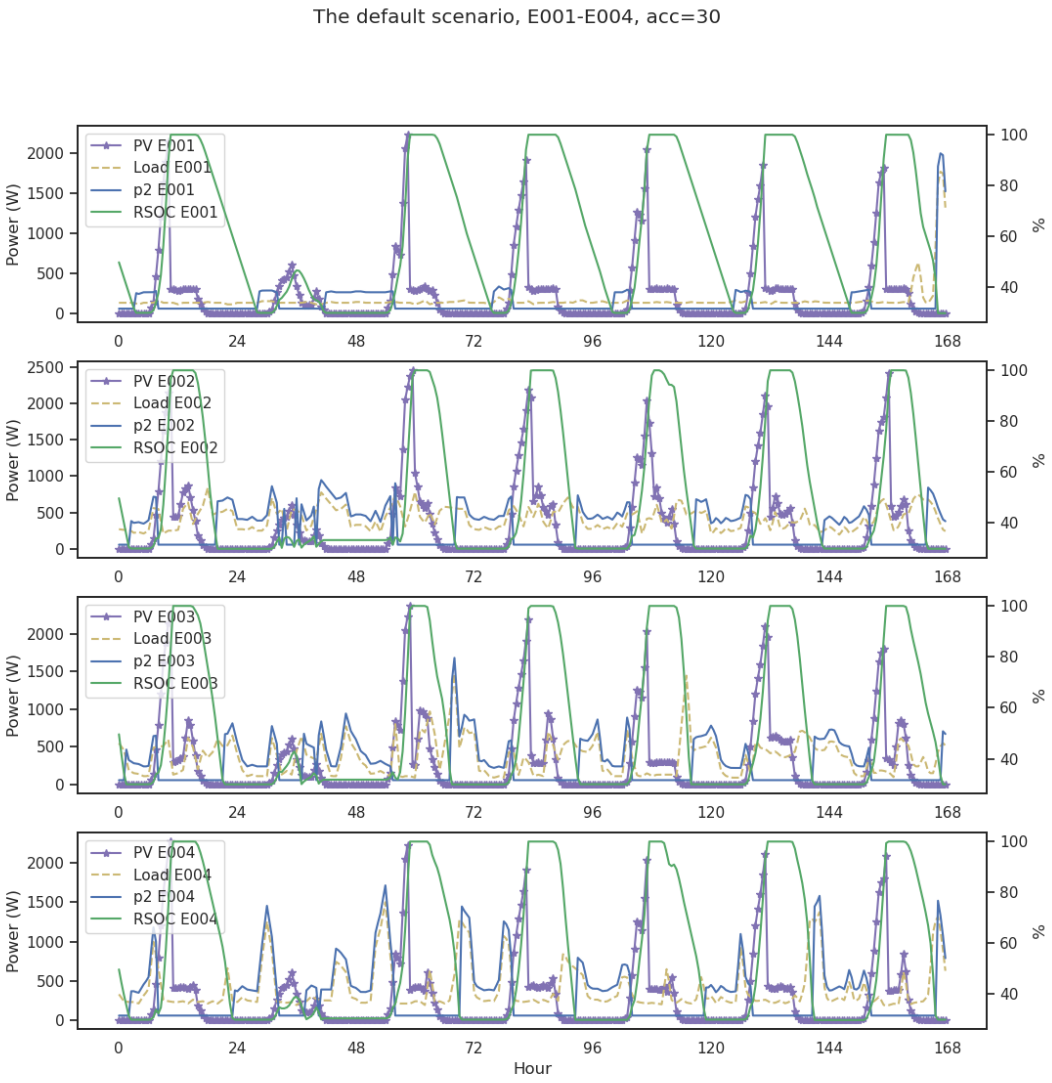


Figure D.2: Simulated data when gl.acc = 30 with sample data.

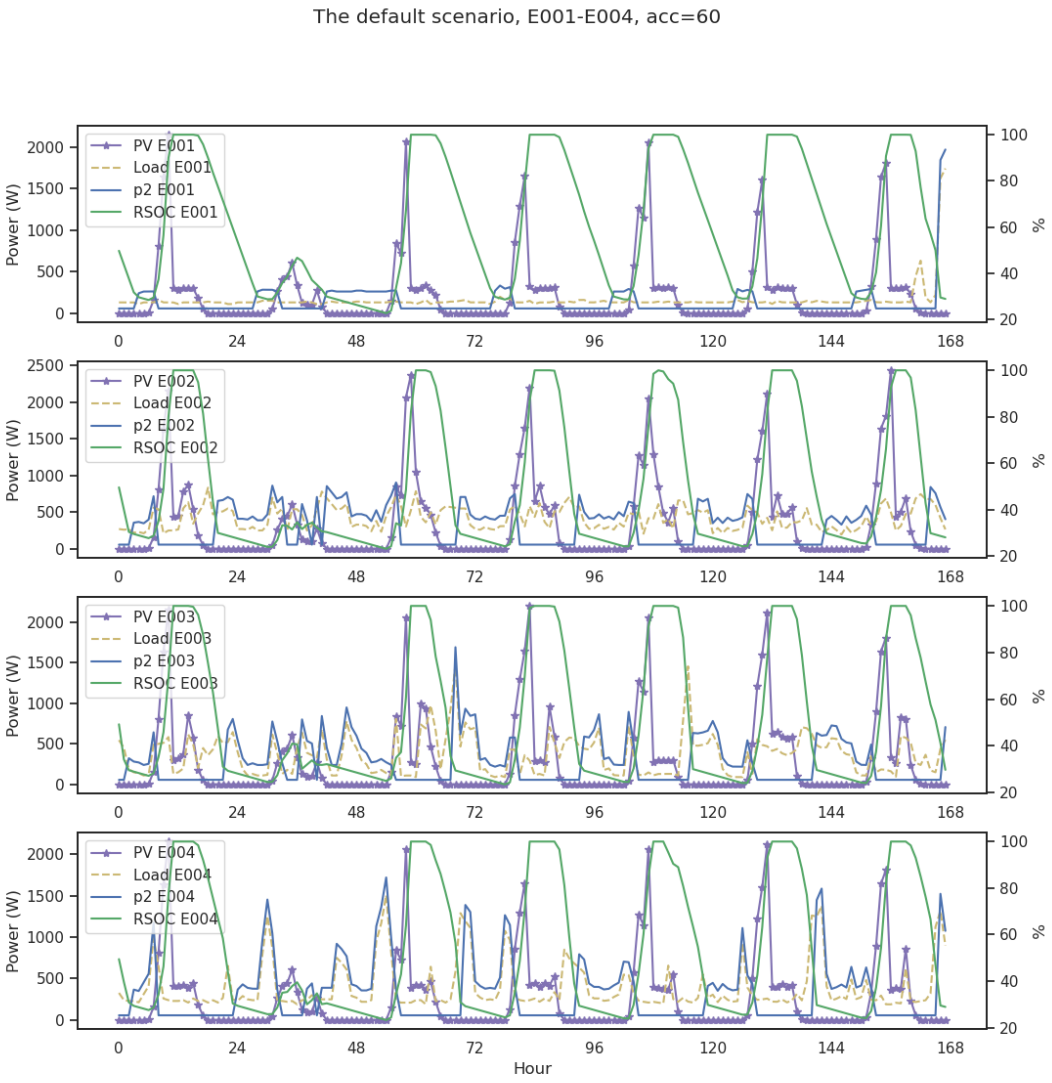


Figure D.3: Simulated data when gl.acc = 60 with sample data.

Appendix E

Simulation Result Data Sample

Table E.1 is a simulation result data for each house generated from the APIS simulator.
Filename: oist_indivLog.csv

Table E.1: Individual data.

id	time	oesunit_id	emu_rsoc	emu_pvc_charge_power	emu_ups_output_power	charge_discharge_power	dcdc_meter_wg	dcdc_powermeter_p2	dcdc_meter_ig
E001	2020/5/8 00:01	Oist_E001	49.85	0	244	429.16	0	57	0
E002	2020/5/8 00:01	Oist_E002	49.84	0	271	459.94	0	57	0
E003	2020/5/8 00:01	Oist_E003	49.89	0	140	310.6	0	57	0
E004	2020/5/8 00:01	Oist_E004	49.06	0	2234	2697.76	0	57	0
E001	2020/5/8 00:02	Oist_E001	49.7	0	246	431.44	0	57	0
E002	2020/5/8 00:02	Oist_E002	49.68	0	273	462.22	0	57	0
E003	2020/5/8 00:02	Oist_E003	49.78	0	135	304.9	0	57	0
E004	2020/5/8 00:02	Oist_E004	48.14	0	2204	2663.56	0	57	0
E001	2020/5/8 00:03	Oist_E001	49.55	0	243	428.02	0	57	0
E002	2020/5/8 00:03	Oist_E002	49.53	0	246	431.44	0	57	0
E003	2020/5/8 00:03	Oist_E003	49.67	0	134	303.76	0	57	0
E004	2020/5/8 00:03	Oist_E004	47.23	0	2177	2632.78	0	57	0
...									

Table E.2 is a simulation result data for all houses generated from the APIS simulator.

Filename: oist_summary.csv

Table E.2: Summary data.

time	pv	demand	acin	wasted	acloss	dcloss	loss	
2020/5/8	35366	52020	36155	1033	9123	14751	23874	
2020/5/9	27711	48582	42375	1027	10435	11560	21996	
2020/5/10	11573	47918	55140	0	12710	6346	19057	
2020/5/11	30591	51819	44688	245	10908	11160	22068	
2020/5/12	38145	57130	42204	0	10464	12782	23246	
2020/5/13	42636	64722	46304	502	10385	14207	24593	
2020/5/14	31037	67631	59204	0	12208	10860	23069	
2020/5/15	42818	66786	49297	153	11000	12896	23897	
2020/5/16	17685	58424	61330	0	12541	8498	21040	
2020/5/17	32652	57010	47676	455	11162	11644	22807	
2020/5/18	34416	71836	61203	493	12081	11530	23611	
2020/5/19	45785	86024	65260	617	12074	13127	25202	
2020/5/20	15151	68399	73756	0	14159	6360	20519	
2020/5/21	37919	46549	33408	0	9622	13300	22923	
2020/5/22	41560	50920	32899	651	9152	14770	23923	
2020/5/23	38819	48515	33979	1158	9534	14060	23595	...
2020/5/24	43369	54029	34636	0	9363	14950	24314	
2020/5/25	42493	51684	33615	311	9334	14639	23973	
2020/5/26	42546	59741	42244	651	9976	14211	24187	
2020/5/27	25254	49968	45801	0	10794	11530	22324	
2020/5/28	18505	61821	63605	0	13303	7085	20389	
2020/5/29	30939	52870	45392	0	10851	11564	22415	
2020/5/30	32807	47758	37532	0	9927	12657	22585	
2020/5/31	32476	62945	53927	850	11492	11778	23270	
2020/6/1	32161	69812	61170	379	12231	10801	23032	
2020/6/2	23691	69533	67848	36	13207	8774	21982	
2020/6/3	24161	65434	63056	0	12780	8985	21765	
2020/6/4	26051	69464	65807	261	12821	9484	22306	
2020/6/5	40600	70558	54260	0	11530	12355	23885	
2020/6/6	40054	75340	60526	1506	11976	12260	24237	
2020/6/7	36924	84485	72186	1124	12779	11462	24242	
sum	1015895	1889727	1586483	11452	349922	360386	710326	
avg	32770.8065	60958.9355	51176.871	369.419355	11287.8065	11625.3548	22913.7419	

avgrsoc	wg	deltaBatt	ssr_real	ssr_pv	sor	r_utility
26	3068	-4545	0.2176	0.5925	0.9716	0.7824
22	3021	-706	0.1132	0.5559	0.9642	0.8868
19	612	-569	-0.1626	0.2296	1	1.1626
25	3196	1168	0.1601	0.6129	0.992	0.8399
24	3307	-210	0.2576	0.664	1	0.7424
21	4538	-572	0.2757	0.6499	0.9884	0.7243
17	4304	-724	0.1139	0.4482	1	0.8861
24	3167	1195	0.2798	0.659	0.9964	0.7202
20	3214	-715	-0.062	0.2905	1	1.062
21	4724	236	0.1679	0.5769	0.9863	0.8321
21	4567	-93	0.1467	0.4778	0.9858	0.8533
18	2776	-492	0.2357	0.5265	0.9867	0.7643
16	2000	-385	-0.084	0.2159	1	1.084
25	3674	1678	0.3183	0.8507	1	0.6817
22	3744	-564	0.3428	0.8051	0.9846	0.6572
24	5401	480	0.3095	0.81	0.971	0.6905
22	4223	-463	0.3504	0.7941	1	0.6496
23	3470	216	0.3538	0.8264	0.9927	0.6462
26	3651	631	0.3034	0.7227	0.9849	0.6966
19	5343	-1394	0.0555	0.4775	1	0.9445
17	1125	-472	-0.0365	0.2917	1	1.0365
21	3914	832	0.1572	0.6009	1	0.8428
20	3371	-167	0.2106	0.6835	1	0.7894
20	4415	-77	0.142	0.5147	0.9745	0.858
21	3062	215	0.1269	0.4638	0.9884	0.8731
19	2998	-284	0.0201	0.3366	0.9985	0.9799
18	2823	-310	0.0316	0.3645	1	0.9684
17	3184	-204	0.0497	0.3721	0.9901	0.9503
18	3126	166	0.2333	0.5778	1	0.7667
21	4065	672	0.2055	0.5406	0.9638	0.7945
21	3272	52	0.1462	0.4377	0.9704	0.8538
648	107355	-5405	4.9799	16.97	30.6903	26.0201
20.9032258	3463.06452	-174.35484	0.16064194	0.54741935	0.99000968	0.83935806