A Distributed Energy Management Strategy for Renewable Powered Communication Microgrid using Game Theory and Reinforcement Learning

by

Rui Hu

Bachelor of Engineer, Sichuan University, 2012

Master of Science, Michigan Technological University, 2015

Submitted to the Graduate Faculty of the

Swanson School of Engineering

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2019

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Rui Hu

It was defended on

November 8, 2019

and approved by

Gregory Reed, Ph.D., Professor, Department of Electrical and Computer Engineering

Zhi-Hong Mao, Ph.D., Professor, Department of Electrical and Computer Engineering

Brandon Grainger, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Nitin Sharma, Ph.D., Associate Professor, Department of Mechanical Engineering and Materials Science

Dissertation Director: Dr. Alexis Kwasinski, Associate Professor, Department of Electrical and Computer Engineering

A Distributed Energy Management Strategy for Renewable Powered Communication Microgrid using Game Theory and Reinforcement Learning

Rui Hu, Ph.D.

University of Pittsburgh, 2019

This dissertation explores the use of energy management strategies for base stations in a communication microgrid, which is intended to be operating in island mode powered exclusively by renewable power sources. The energy management strategy aims at searching for an optimal energy management plan considering both communication quality and energy availability. In this dissertation, the objective is to accomplish such energy management using distributed control architecture. Three approaches have been made: multi-player energy management game, multi-agent reinforcement learning, and a hierarchical load-ratio learning game algorithm. The modeling, performance, and applicable conditions of the proposed algorithms are discussed and compared. Numerical simulation of communication microgrids in multiple cases implemented with the three algorithms was conducted. As the results show, the hierarchical load-ratio learning game algorithm has a better performance compared to the multi-player game approach with less computation complexity and a faster-converging speed compared to that of the reinforcement learning approach.

Table of Contents

List of Tables

List of Figures

Preface

The motivation of this dissertation origins from my passion for realizing distributed control in the power system. As we know it, the power system has been operated for more than a century with a centralized control architecture. It mimics well the conventional social structure in industry era: obedient and hardworking low-level parts pave the foundations, upon which lies the layer of farsighted decision-makers with the power of deciding the marching direction of the whole system. But it may not have to be like this. As distributed power resources and energy storage integrated into our society further, the power system may become a public infrastructure both operated and maintained by everyone who uses it. Most importantly, people are free to join or detach from any energy group, which might be the microgrids we are talking about today.

I would like to thank my supervisor and committees, who have been continuously providing suggestions and insights along my research path. Your sharp thoughts and observations always remind me of my ignorance in front of the knowledge mansion.

I would also like to give a special thanks to my wife, Li Li, and my parents. It would be impossible for me to fight against all the haunted ghosts of self-doubts without you along my way climbing towards an unknown summit. Your words always calm me down and remind me of my coordination in this vast, everlasting-seemed world.

Rui Hu

10/13/19 night

1.0 Introduction and Literature Review

Communication networks, especially emergency communication systems, are required to maintain operational under all circumstances [1]. However, the effect of recent natural disasters was a demonstration of the urgency to improve the resilience of communication sites [2, 3]. During these disasters, wireless base stations (BS), the fundamental components in the communication network, were found especially vulnerable to electric grid power outages [1, 4]. Because although a vast majority of the BSs survived the direct impact of the disaster without any physical damages, they were unable to maintain functioning because of interrupted power supplies [2]. Most of the BSs are equipped with back-up battery units, but these batteries are usually designed to feed the BS load for no more than several hours, which is significantly shorter than the outage duration of the electric grid caused by a natural disaster. The conventional solution to extend the power backup time is to use standby diesel generators. However, equipping all BS with onsite diesel gensets is a practice observed in few areas around the world [2, 5]. Moreover, roads and transportation systems—the lifelines used to refuel these generators—have to be operational after the disaster to ensure the fuel supply. Even if every BS is equipped with a permanent diesel generator or another type of backup power source, the use of them still presents some issues under extreme disruptive conditions, such as failures due to that these generators are not designed for long-time operation [6].

As illustrated in [7-10], renewable energy sources and microgrids may be alternative options with respect to separate generator units. Renewable energy sources such as solar panels and wind turbines do not require any lifeline or additional energy source to keep its operation. Moreover, the microgrid architecture enables the BSs to share load and energy storage such that the overall system

availability is improved. Nonetheless, other challenges appear when harvesting renewable energy sources in an isolated microgrid. One of the most critical issues is that renewable energy sources, like photovoltaic (PV) cells, have a variable output characteristic, which not only makes energy storage devices indispensable but also requires the BS controllers to plan the energy usage in real-time to avoid energy deficiency or interrupted operation.

Much research effort has been done on energy management in the communication community. One of the solutions is by switching off base stations to the total load demand, as discussed in [11] (SWES), [12] and [13]. These algorithms are realized by coordination between BSs with a sequential broadcast system advertising operating status, communication traffic amount, and requests for switching on/off. The objective of SWES is to determine the minimal number of BSs required to serve the area with acceptable communication quality. This number is obtained by a greedy search computed by a master-planner. Another approach called Intelligent Cell brEathing (ICE) is introduced in [14] aims at maximizing the utilization of renewable energy. ICE achieves its goal by rearranging users to BSs with larger renewable generation capacities hence more renewable power is utilized. Aside from the previous two, methodologies considering green energy availability and delay performance include GALA [15], IDEA [16] and TEA [17]. IDEA and TEA. The objectives of them are to minimize a weighted function of the energy consumption from the main-grid and the traffic delivery latency by manipulating the BS coverage area and user connections. The user association and coverage area are controlled in a distributed way with full communication, where users and BSs receive broadcasted information from each other and run an exhaustive search by turn. However, the communication links between BSs are not always available in a natural disaster. For example, when a communication microgrid is hit by a hurricane, the relay stations and BSs themselves might be damaged, resulting in possible disconnections

between the other BSs. In such a scenario, the BSs in the microgrid need to plan their operation independently and adapt their operation mode. Thus, in this project, we aim to design an autonomous energy management algorithm for each BS in this microgrid without need of designated communication link.

First, the BSs need ways to control their energy consumption. In this dissertation, the BSs are equipped with communication traffic shaping technology to accomplish the energy consumption control [9, 18, 19]. Applied with this algorithm, a communication traffic shaping factor (CTSF) σ is applied in a BS to limit the volume of cellular traffic. Usually, a setting of $\sigma<1$ indicates that the traffic through the BS, as well as the correspongding power consumption, is reduced. Such an arrangement may have an effect on the quality of service (QoS) of the call or video stream, which may influence the experience of users in an active call [9]. So the BS needs to decide what CTSF is optimal considering the present load demand and stored energy condition. Then, an objective function measuring a weighted sum of communication quality and the battery SoC distribution is implemented in the BSs as an optimum metric. The objective function will be shown in section 3.2. The last part is to equip BSs with the ability to adapt to a changing environment. Different solutions are provided to explore the unknown environment, including load identifiers and machine learning algorithms.

Two major approaches were proposed to solve for the optimal energy management plan: multi-player game and reinforcement learning. The first approach model the energy management decision-making process as a multi-player game, where BS controllers occupy the role of the 'players.' In the power system industry, game theory has been applied in system operation optimization and load scheduling [20-24]. Most of the studies were discussing power system marketing, price bidding, demand response, and load planning optimization. The application of

game theory in our project could be classified as a form of load response. In this energy management game, each player's payoff is not only decided by his own but all other players' moves. The payoff function for players in this game is the aforementioned objective function and its modified version with limited information. This game could be modeled in two ways: zero-sum game or common-interest game. In the zero-sum game, the BS treats other BSs as competitors. While in the second approach, players share the same payoff and cooperate for a solution that's optimal for the whole system. The two different approaches both lead to solutions of Nash Equilibrium, but their applicable conditions, computation costs, and resulting strategies could be different. Simulations of BS microgrids applying these methods were conducted, and their performance was compared to a centralized controller using exhaustive searching. A load identifier is designed so that the BS can update their power consumption model during the game. One disadvantage of the multi-player game approach is its computation complexity. As will be shown in section 3.6, the computation cost of a game increases exponentially as the number of BSs, and their actions increase. Therefore, a virtual two-player game approach was made to solve the original n-player game with reduced computational complexity. The optimality of such an approach is limited to be minimum-maximized if such game is modeled in a zero-sum form. The performance of the virtual two-player game is also validated using Monte Carlo tests.

The second approach made in this dissertation is to equip the BSs with machine learning algorithms so that the BSs could update their energy management strategies according to the actual feedback from the environment/microgrid. The learning algorithm makes the BSs explore the available load response actions and update their load response strategies based on their corresponding outcomes. The fundamental idea behind machine learning is empiricism, which claims a pre-defined model is capable of capturing critical features of the actual world or a physical

process in it explicitly from experience. Despite their specific form, most of the machine learning algorithms try to achieve a similar goal: predict the outcome of a complex system using a well-tuned/trained, sometimes even more complex model. Intriguing as it is, this procedure is similar to what we engineers are seeking to achieve some times, which ought to be a branchy trail—there is great freedom in the choices of models, parameters, and how should they be organized. In this sense, engineers are some times similar to computer scientists. The type of machine learning algorithm implemented in this project is reinforcement learning, which utilizes interaction between BS (which is also called an agent in an RL process) and their environment to comes up with a strategy that maximizes the agent's overall payoff [25]. Two different RL methods were applied: Q-learning and Linear Reward-Inaction algorithm. Both of the learning algorithms model the energy management as a Markov decision process. However, the main difference is that the Q-learning only maximizes one single agent's long-term payoff while the Linear Reward-Inaction targets the equilibrium of multiple agents. The reinforcement learning algorithm requires no prior knowledge of the microgrid. Nevertheless, the RL process requires a notably long time to train the agents depending on the scale of the searching space, which is related to the numbers of BSs, actions, and the system's possible states.

The latest algorithm was proposed to address the performance drop issue of the virtual two-player game applied to a large scale microgrid. It is an algorithm combining the multi-player game and reinforcement learning. How these two algorithms are integrated is enlighted by observation in virtual player game. In the virtual two-player game, a player evaluates the other player's possible moves knowing the other player's load demand. In an actual two-player game, such knowledge is accurate, thus the actual equilibriums are reachable to players. However, as the number of players increases, the estimated behavior of the virtual player with a high load demand is different from the

actual joint action of multiple players with lower load needs. So, this ‘misunderstanding’ is potentially one of the causes that drive the players away from reaching equilibrium. Implemented with the load-ratio learning algorithm, the BSs adjust their load demand models using machine learning algorithms. And the immediate load response decision is solved in the virtual two-player game way. As the simulation results show, this combined algorithm overcomes the two-player game performance drop in a large scale system with low computation cost and less training time.

2.0 Communication Microgrid Structure and its Energy Management

2.1 Communication Microgrid Architecture

2.1.1 Microgrid Components

An example of a communication dc microgrid equipped with renewable resources is shown in Figure 1. In order to provide some context for the discussion and without loss of generality, it is assumed that one battery unit and renewable power sources are placed at the central station, another BS is only equipped with battery unit while the third site has no energy source or storage device. The connections of batteries to the dc bus could be direct in consideration of reliability. However, in some cases where batteries have a significant difference in sizes and energy levels, it may be required to have power electronic interfaces such as dc converters to stabilize the bus voltage and share energy in batteries properly, just as shown in the scheme [26-28]. Therefore, points of load converters in BS power supplies are represented by buck-and-boost converters, regulated via PI controllers to maintain a rated load voltage.

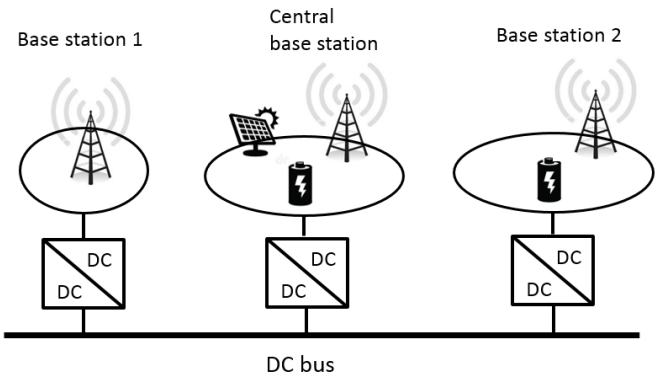

Figure 1: Communication microgrid scheme

2.1.2 Communication Base Station Load

In the base station, the nominal load consists of two parts: base load and communication traffic dependent load

$$P = P_B + vP_T \tag{2-1}$$

where P_B is the constant base load, P_T is the base traffic dependent load, and v is the BS utilization factor that represents the communication traffic intensity. The utilization factor is a random variable that depends on the number of users and the volume of communication traffic data. In the simulations, the BS loads are modeled as controllable resistors with resistance equal to

$$\mathrm{R} = \frac{V^2}{P_B + vP_T} \tag{2-2}$$

where V is the rated load voltage.

2.1.3 Communication Traffic Shaping and Quality of Service

In the base station, the transmitted communication traffic is regulated using the communication traffic shaping technology [9, 29]. This technology allows BS controllers to control their processed traffic by reducing the real-time signal transmission rate and data traffic throughput. When communication traffic shaping is applied to the BS, a traffic shaper controls ("shapes") the actual throughput (equivalent to the total volume of traffic) at the output of a BS. Such a setting reduces the BS's energy consumption as shown in eqn. (2-3)

$$P = P_B + \sigma v P_T \tag{2-3}$$

where σ is called the communication traffic shaping factor (CTSF). Since the action of shaping traffic entails a reduction of bit rate from the one required by different network traffic, it will lead to an increased delay or higher compression ratio for interactive video or speech traffic. In an LTE base station, a radio frame is divided into minimum units of transmit resources called "Resource block" (RB). Without traffic shaping, all ongoing calls will require ${R_B}^T$ resource blocks. However, when applying traffic shaping, the actual number of active resource blocks becomes

$$R_B(t) \leq \sigma \cdot {R_B}^T \tag{2-4}$$

Correspondingly, the maximum transmitted bit in this BS is limited. The impact of QoS caused by CTSF is measured for video traffic through the objective quality metric of peak signal-to-noise ratio (PSNR). The relation between PSNR and QoS is demonstrated in [13]. Also, as discussed in [30], the relation between PSNR and CTSF can be approximated by a function

$$q_v = a \cdot \log(\sigma \cdot r) + b \tag{2-5}$$

where q_v is the video quality measured in PSNR, r is the nominal bit rate, and *a* and *b* are constants based on the choice of source codecs. Details on how parameters in eqn. (2-5) are obtained could also be found in [13].

2.1.4 Power Electronics Interface

In the communication microgrid, the dc converters are responsible to interface the BSs with the microgrid, as shown in Figure 1. The converters can be realized using conventional single-input-single-output topologies such as a buck-and-boost converter, as long as the bi-directional power flow between BS and the microgrid is allowed. In this project, the converters are ideal buck-and-boost converters. The scheme of a buck-and-boost converter connecting BS load and the dc bus is shown in Figure 2. The converter's dynamic equations are shown in eqn. (2-6) and eqn. (2-7).

$$L\frac{dI_L}{dt} = V_i - g \cdot V_{o1} \tag{2-6}$$

$$C\frac{dV_{o1}}{dt} = \frac{V_{bus} - V_{o1}}{R_o} + g \cdot I_L \tag{2-7}$$

where g is the switch control signal. Assuming the converter works in steady-state, and the duty cycle is D, the two dynamic equations can be replaced by eqn. (2-8) and eqn. (2-9) [31].

$$V_i = D \cdot V_{o1} \tag{2-8}$$

$$\frac{V_{bus} - V_{o1}}{R_o} = -D \cdot I_L \tag{2-9}$$

In this system, the load voltage is controlled by the dc converter with a PI controller. Assuming the sensor and controller have sufficient measurement accuracy and response speed, the load

voltage V_i could be regulated tight and treated as a constant. Therefore, the load current I_L only depends on the load resistant *Rload*, as shown in eqn. (2-10).

$$I_L = -\frac{V_i}{Rload} \tag{2-10}$$

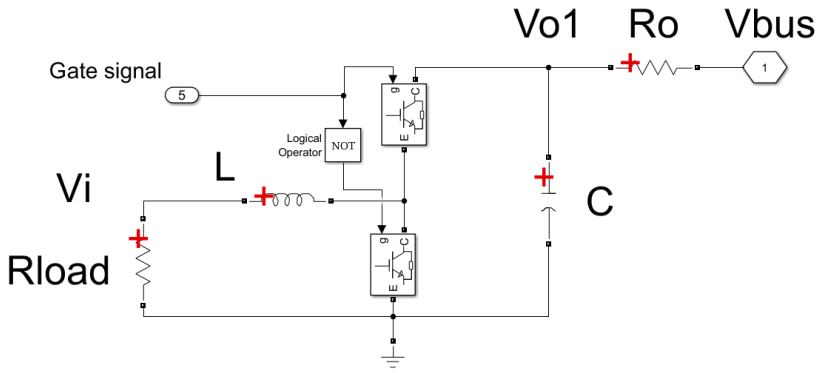

Figure 2: Buck-and-boost converter scheme.

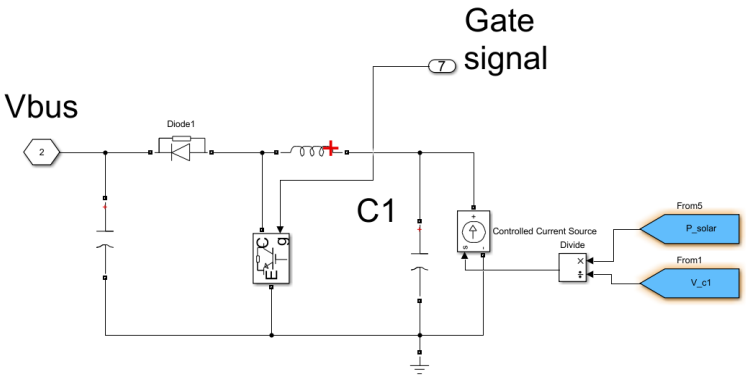

Figure 3: Soar cell generator with converter model

2.1.5 Solar Cell Generator

The solar cell power generator in this system is modeled as a controlled current source. It is assumed that maximum power point tracking (MPPT) is implemented in the PV dc converter during the whole operation to get the most power out from the solar cell. The model of the solar cell and its converter are shown in Figure 3. The solar cell power output curve is obtained from Cambridge Solar Panels, which is a typical 1kW solar panel, as shown in Figure 4 [32].

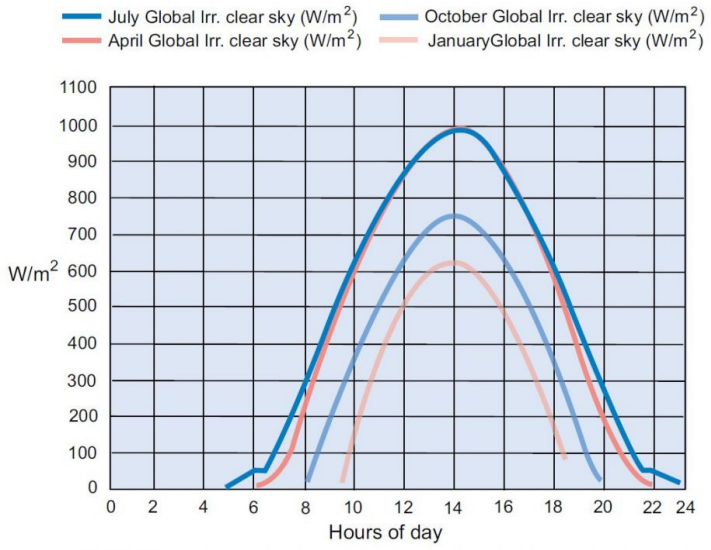

Figure 4: Solar panel output power curve

2.1.6 Battery SoC Discretization

The batteries simulated in the system are assumed to be ideal energy storage devices and have no loss during the charge and discharge process. Also, nor battery lifetime impact or switching effect of charging/discharging is considered. The battery's SoC level is divided into

multiple discrete stages. For any given stored energy, the battery energy level i could be calculated by eqn. (2-11):

$$i = \frac{E_{n_i}}{E_f} m \tag{2-11}$$

where $floor(x)$ is a function that obtains the largest integer smaller than x, m is the total number of energy stages, E_f is the fully charged battery energy and E_{n_i} is the current stored energy. In this study, it is assumed that each BS has the capability of measuring its battery SoC, such as the SoC-V detection method [33].

2.2 Microgrid Energy Management in Island Mode

2.2.1 Objective Function

An islanded microgrid needs to maintain its power-load balance and keep sufficient stored energy to deal with possible deficient-power situations. In this study, the energy objective set for the controller in the microgrid is to maintain 80% of the total battery SoC at the end of one day's operation. As mentioned in section 2.1.3 , a lower CTSF reduces the BS energy consumption and stores more energy in the battery but also results in worse QoS. Therefore, the BS needs to evaluate the outcomes of different CTSF strategies and choose the one that has an optimal result considering both QoS and battery SoC distribution in the future. A metric measuring the optimality of a CTSF strategy is defined in (2-12). The metric has the form of a weighted objective function:

$$obj(\sigma, SoC, t) = w_{com} \cdot f_{com}(\sigma, t) + w_{SoC} \cdot f_{av}(\sigma, SoC, t) \quad (2\text{-}12)$$

where w_{com} and w_{SoC} are weighting factors for communication quality and SoC distribution /energy availability, f_{com} is the normalized SINR, and f_{av} is the energy availability function. In this study, the energy availability function is defined as the probability of the battery SoC reaching a certain level. Therefore, the goal of the microgrid is to search for a CTSF strategy $\sigma(t)$ that maximizes the objective function eqn. (2-12)

$$\max_{\sigma(t)} \text{obj}(t, \sigma(t), SoC(t)), \sigma(t) \in [\sigma_{min}, \sigma_{max}] \quad (2\text{-}13)$$

In the following section, the energy availability function $f_{av}(\sigma, SoC, t)$ will be discussed in detail.

2.2.2 Computing Battery SoC Distribution

The availability function $f_{av}(\sigma, SoC, t)$ is the battery SoC distribution at time *t*. In this study, both renewable power generation and load consumption are modeled as random variables as a function of time *t*. Two assumptions are made to compute this function:

- BS load and solar cell generation curves information are shared among all controllers. The data is extracted from the actual load and weather records. In this project, the load and solar cell data are obtained from [32, 34] and abstracted using a curve fit tool in MATLAB. The fitted power/load curves will be shown in the simulation chapter.

- The BS load follows a Poisson distribution, and the PV power follows an exponential distribution. Both of them are time-independent variables. This assumption is made based on [35] and the empirical results from [36].

At time *t*, the probability density function (PDF), the mean value, and variance of the PV power generation are listed in eqn. (2-14)-(2-16) [36]:

$$\mathrm{f}(P_{solar}) = \lambda_{solar}(\mathrm{t})\, e^{-\lambda(t)P_{pv}} \tag{2-14}$$

$$E(P_{solar}) = \lambda_{solar}(\mathrm{t})^{-1} \tag{2-15}$$

$$V(P_{solar}) = \lambda_{solar}(t)^{-2} \tag{2-16}$$

where λ_{solar} is the rate parameter of the PV power distribution. The probability mass function (PMF), mean value and variance of the BS load are listed in eqn. (2-17)-(2-19) [35].

$$\mathrm{f}(P_T) = \frac{\lambda_{com}(\mathrm{t})^k e^{-k}}{k!} \tag{2-17}$$

$$E(P_T) = \lambda_{com}(\mathrm{t}) \tag{2-18}$$

$$V(P_T) = \lambda_{com}(\mathrm{t}) \tag{2-19}$$

where $\lambda_{com}(\mathrm{t})$ is the mean communication traffic data arrival rate. Then, the power provided by the batteries units are

$$\Delta \mathrm{P} = P_{solar} - P_{load} = P_{solar} - (P_B + \sigma P_T) \tag{2-20}$$

which forms a new random variable with mean and variance equal to:

$$E(\Delta P) = E_{P_{solar}} - (E_{P_B} + \sigma E_{P_T}) \tag{2-21}$$

$$V(\Delta P) = \sqrt{{V_{P_{solar}}}^2 + (\sigma V_{P_T})^2} \tag{2-22}$$

where *E* stands for expectation, and *V* stands for standard variation. The time interval *T* was chosen to be one hour and was divided into *n* sub-intervals. During each sub-interval, the change of power/load is negligible so that its value during the small period is constant. Then, the overall energy change S_{total} during *t* hour is computed as the sum of $n \cdot t$ independent random variables as shown in eqn. (2-23).

$$S_{total} = \sum_{k=1}^{n \cdot t} \frac{T}{n} \Delta P(k) \tag{2-23}$$

According to the central limit theory, the distribution of a sum of independent variables converges to that of normal distribution [37]. If *n* is chosen to be large enough (typically larger than 17), the distribution of the sum could be replaced by

$$S_{total} \rightarrow N(\mu, \delta^2) \tag{2-24}$$

where $\mu = \sum_{k=1}^{nt} \frac{T}{n} E(\Delta P_k)$, $\delta^2 = \sum_{k=1}^{nt} (\frac{T}{n} V)^2 (\Delta P)$. Then, the probability of the overall battery SoC transferring from level *i* to level *j* in time *t* (Hr) is

$$p(i,j) = F_{S_{total}}(E_{n_i}) - F_{S_{total}}(E_{n_i}) \tag{2-25}$$

where F is the cumulative distribution function of S_{total}, and E_{n_i} and E_{n_j} are energy stages. Then, the probability of battery SoC reaching the desired level is

$$P(SoC > thresohold) = \sum_{l=thresohold}^{l=top\_limit} p(i,l) \tag{2-26}$$

2.2.3 Objective Function Analysis

In this section, the optimum of the objective function will be discussed. The objective function is

$$obj(\sigma, SoC, t) = w_{com} \cdot f_{com}(\sigma, t) + w_{SoC} \cdot f_{av}(\sigma, SoC, t) \tag{2-27}$$

where

$$f_{com}(\sigma, t) = c \cdot \log(\sigma(t) \cdot \bar{r}(t)) \tag{2-28}$$

$$f_{av}(\sigma, SoC, t) = P\left(SoC(t) \geq SoC_{goal}\right) = \int_{(SoC_{goal}-SoC_{now})E_f}^{+\infty} pdf(S_{total})d(S_{total}) \tag{2-29}$$

where c is the PSNR normalization factor, $\sigma(t)$ is the CTSF, $\bar{r}(t)$ is the average nominal bit rate, and eqn. (2-29) is a continuous expression of eqn. (2-25) using probability density function $pdf(S_{total})$. To make the formula clearer, expectation and variation of battery power consumption are rewritten in the following forms:

$$E(S_{total}) = (24 - t) \cdot 3600 \cdot (\overline{E_{P_{solar}}} - (\overline{E_{P_B}} + \sigma\,\overline{E_{P_T}})) \tag{2-30}$$

$$V(S_{total})^2 = \sum_{k=t}^{24n}[(\frac{T}{n}V_{P_{solar}}(k))^2 + (\sigma \cdot \frac{T}{n}V_{P_T}(k))^2] \tag{2-31}$$

where

$$\overline{E_{P_{solar}}} = \frac{\sum_{k=t}^{24n} E_{P_{solar}}(k)}{n(24-t)} \tag{2-32}$$

is the algebraic mean of all solar cell power expectations in the remaining operating periods $24 - t$. The same rule applies to E_{P_B} and E_{P_T}. Thus, the energy availability function eqn. (2-29) could be expressed as

$$f_{av}(\sigma, SoC, t) = \int_{(SoC_{goal} - SoC_{now})E_f}^{+\infty} \frac{1}{\sqrt[2]{2\pi\, V(\Delta P)^2}} e^{-\frac{(S_{total} - E(S_{total}))^2}{2\, V(\Delta P)^2}} d(S_{total}) \tag{2-33}$$

Substituting eqn. (2-33) to the objective function eqn. (2-27) and the objective function becomes

$$\begin{aligned} obj(\sigma, SoC, t) &= \mathrm{w_{com}} \cdot (\mathrm{c} \cdot \log(\sigma \cdot \vec{\mathrm{r}})) + \mathrm{w_{SoC}} \\ &\cdot \int_{(\mathrm{SoC_{goal}} - \mathrm{SoC_{now}})\mathrm{E_f}}^{+\infty} \frac{1}{\sqrt[2]{2\pi\, \mathrm{V}(\Delta \mathrm{P})^2}} \mathrm{e}^{-\frac{(\mathrm{S_{total}} - \mathrm{E}(\mathrm{S_{total}}))^2}{2\, \mathrm{V}(\Delta \mathrm{P})^2}} \mathrm{d}(\mathrm{S_{total}}) \end{aligned} \tag{2-34}$$

After solving the integration, eqn. (2-34) could be expressed as

$$obj(\sigma, SoC, t) = w_{com} \cdot (c \cdot \log(\sigma \cdot \vec{r})) + \cdots$$

$$\ldots + w_{SoC} \left(\frac{1}{2} - \frac{(\mathrm{t} - 24)(\overline{V_{P_{solar}}}^2 + \overline{V_{P_T}}^2 \sigma^2)\mathrm{Erf}\left[\frac{\mathrm{Ebattery}(-SoC_{goal} + SoC_{now}) + (24 - \mathrm{t}) \cdot (\overline{E_{P_B}} - \overline{E_{P_{solar}}} + \overline{E_{P_T}}\sigma)}{\sqrt{2}(t-24) \cdot \sqrt{\overline{V_{P_{solar}}}^2 + \overline{V_{P_T}}^2 \sigma^2}}\right]}{2\sqrt{(\overline{V_{P_{solar}}}^2 + \overline{V_{P_T}}^2 \sigma^2)^2 (t-24)^2}} \right) \tag{2-35}$$

where $Ebattery$ is the stored energy of the battery when fully charged. The analytical maximum of this function is difficult to compute due to the error function, but its shape could be seen from a case portrait, as shown in Figure 5.

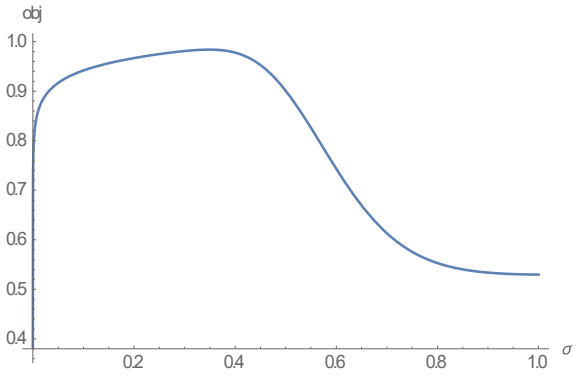

Figure 5: Objective function vs. CTSF case 1, t=5

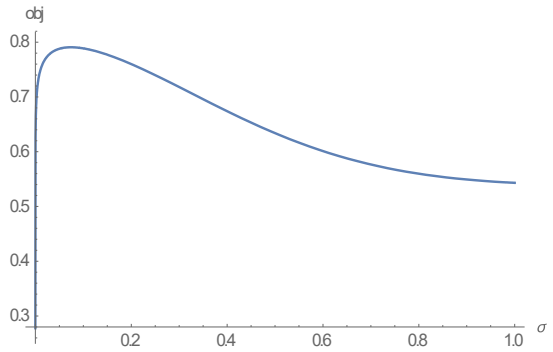

Figure 6: Objective function vs. CTSF case 2

The substituted data are listed in Table I. As can be observed from Figure 5, the objective function has a single maximum in the range of $\sigma \in [0,1]$ in this scenario. Generally, higher load demand and larger battery capacity would result in a maximum point closer to $\sigma = 0$. The slope of the function's decreasing region is influenced by the renewable power and load demand variations $\overline{V_{P_{solar}}}$ and $\overline{V_{P_T}}$. For instance, if these variations are larger, the obj-σ curve becomes smother as shown in Figure 6 where $\overline{V_{P_{solar}}}$=1000. In most scenarios, the objective function has one single maximum in the range of $\sigma \in [0,1]$ as shown in Figure 7. But if the SoC level or power generation is too low, it is possible the objective function has a higher value with large CTSF. The shape of the objective function vs. CTSF in this situation is shown in Figure 8. From the controller

perspective, it means the benefit of a large CTSF overwhelms the increase in energy availability. Therefore, the BS controller yields the conclusion that a large CTSF is preferred with a critically low energy level. To prevent this unreasonable decision-making from happening, a large energy availability weighting factor w_{SoC} could be set so that the increasing in QoS does not dominating the obj function as shown in Figure 9 where the $w_{SoC} = 0.7$. Alternatively, a probability threshold could be set, and the objective function is modified to a piecewise function:

$$obj^* = \begin{cases} Sum(obj), & P(SoC > thresohold) > P_{min} \\ (1-\sigma)/10, & otherwise \end{cases} \quad (2\text{-}36)$$

which turns the objective into a negatively related function to the CTSF when the probability of reaching the desired SoC goal is lower than a threshold (P_{min}). With this modification, the objective function vs. communication shaping factor would be like the one shown in Figure 10. So the optimal CTSF is either the peak of the error function or the CTSF that guarantees at least a probability of P_{min} reaching the desired SoC goal.

Table 1: Evaluation parameter values

Symbol	PARAMETER	Value
w_{com}	Communication quality weight	0.5
w_{SoC}	Energy availability weight	0.5
$Ebatte$	Battery fully charged energy	24 kWh
$\overline{E_{P_{solar}}}$	Solar power generation expectation	1 kW
$\overline{E_{P_B}}$	BS base load expectation	200 W
$\overline{E_{P_T}}$	BS traffic depended load expectation	800 W
$\overline{V_{P_{solar}}}$	Solar power generation variance	4000
$\overline{V_{P_T}}$	BS traffic depended load variance	4000
SoC_{goo}	Desired battery SoC level	0.8
SoC_0	Initial Battery SoC level	0.7
BW	BS total bandwidth	10MHz
a	PSNR-rate bit curve parameter	10.4
b	PSNR-rate bit curve parameter	-23.8
r	Nominal transmit rate bit	2 Mbps

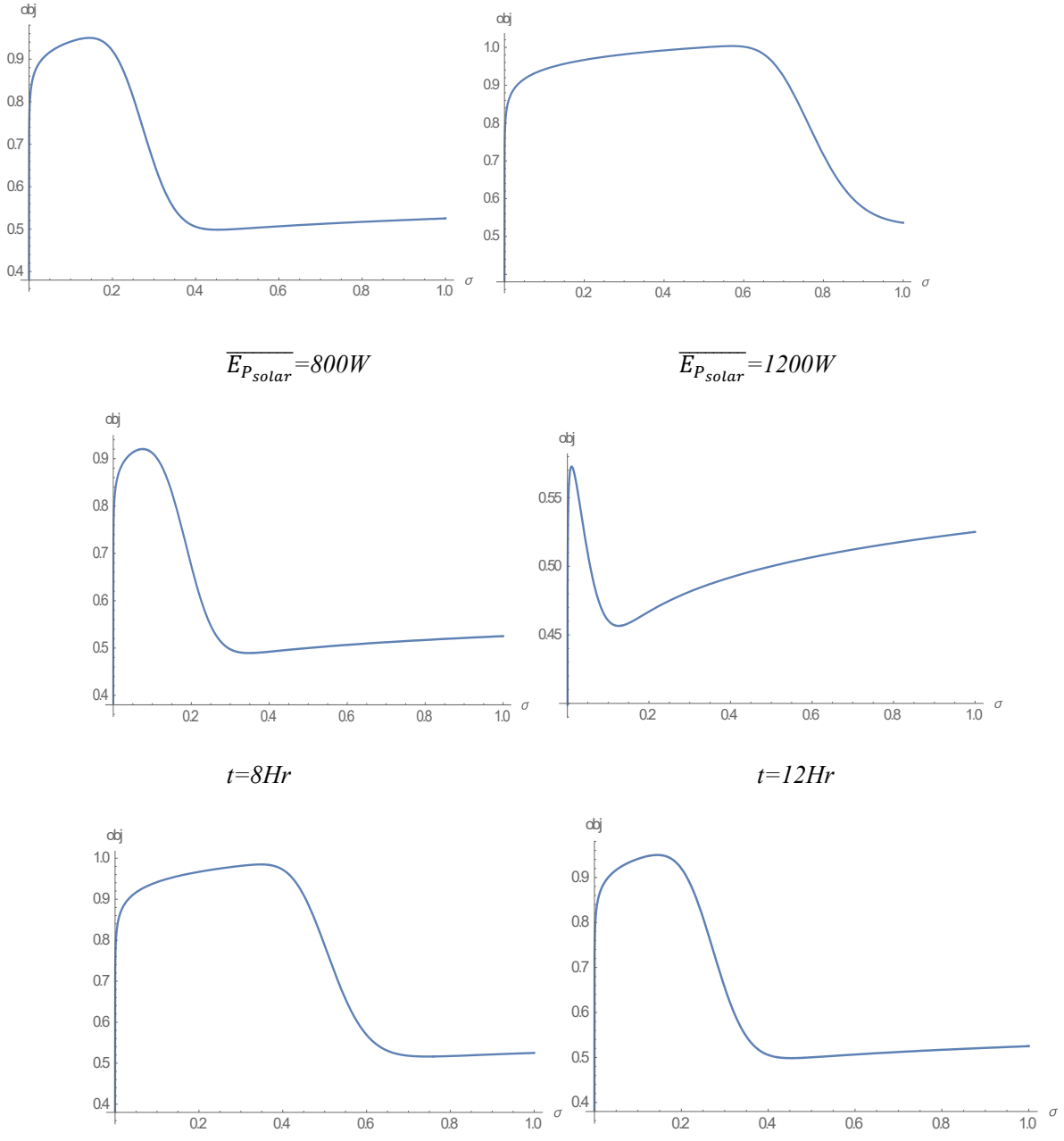

Figure 7: Objective function vs. CTSF with different conditions

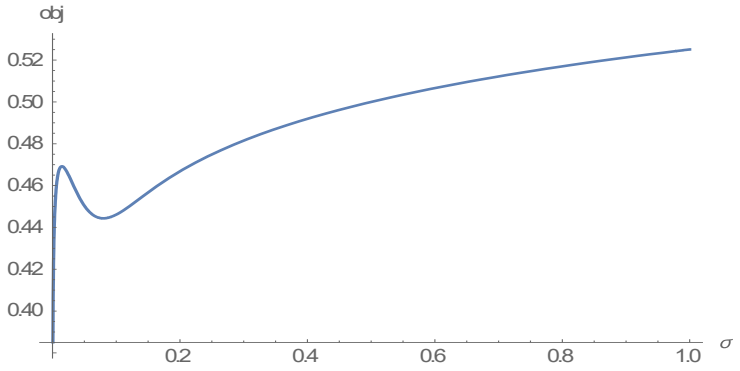

Figure 8: Objective function vs. CTSF case 3 (SoC(t)=0.49, t=16 Hr)

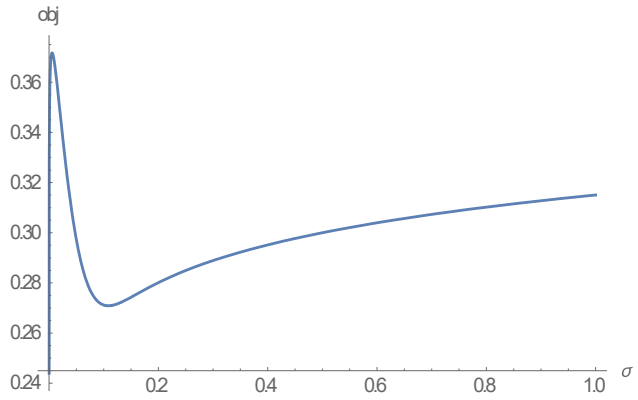

Figure 9: Objective function vs. CTSF case 4 (SoC(t)=0.49, t=16 Hr, w_SoC=0.7)

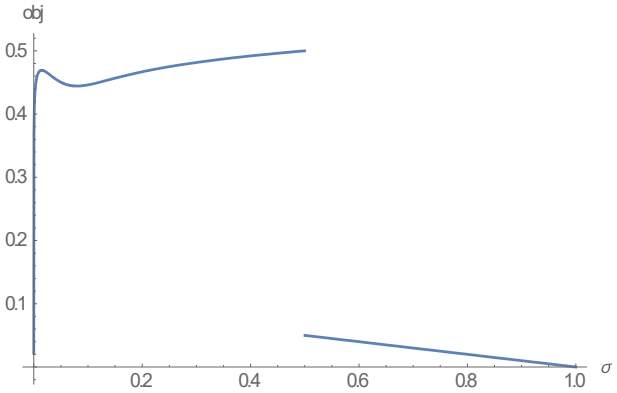

Figure 10: Modified objective function (SoC(t)=0.49, t=15)

2.2.4 Energy Management Strategy Obtained by Exhaustive Search

If a central controller coordinates the overall communication network microgrid, the optimal load planning could be obtained using an exhaustive search. In this setting, the central controller monitors ever BSs' power generation, load demands, and battery SoC levels. Then, the CTSF that maximize the objective function could be found by exhaustive examining the possible CTSFs

$$\sigma^0 = \underset{\sigma}{\operatorname{argmax}}\, obj^*(\sigma) \tag{2-37}$$

This method, however, requires accurate knowledge of the microgrid and reliable communication links between BSs. Additionally, the monitoring of all BS required a reasonable amount of communication bandwidth, which increases the load burden for all BSs. The system performance relies mostly on the proper operation of the central control center, wireless communication linkage quality, and fidelity. Failures or disturbances in these elements could cause a sub-optimal operation or failure of the whole system. For example, if one of the communication links between the central controller and one BS is cut off (see Figure 11), the controller could not decide what the optimal CTSF is because it has no information on whether and how the disconnected BS is operating. Such circumstances might be avoided by presetting a set of protocols such as setting load consumptions to the minimum or shutting down the BS when no control signal is received. However, such solutions do not make the most use of the available resources for a disaster-affected area. As we shall see in the distributed control approaches, instead of receiving orders from other controllers

passively, a base station could operate by estimating the other BS's actions or develop a load plan based on its past experience.

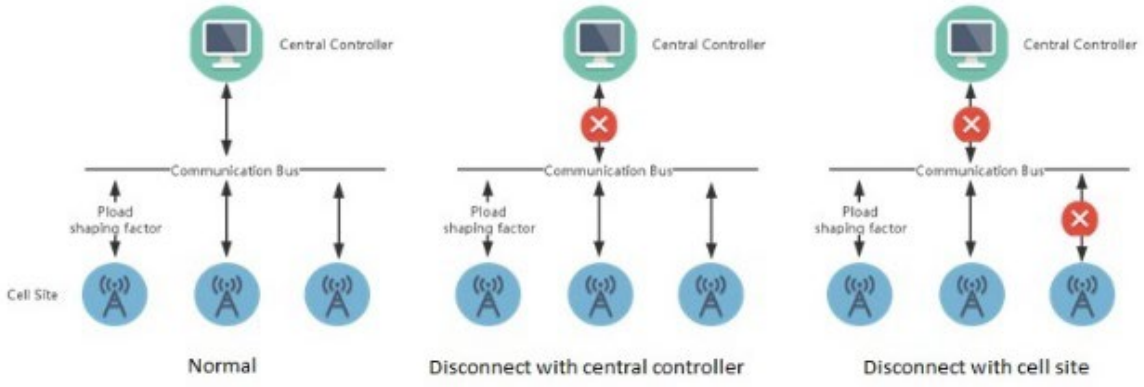

Figure 11: BS communication network status in different scenarios

2.2.5 Energy Management Obtained using Multi-agent System

In this project, we aim at solving the energy management in a multi-agent system (MAS) setting. The definition of an agent is a computational mechanism that operates highly autonomous and takes actions based on information obtained from the environment [38]. A multi-agent system is one in which there is more than one agent, where they interact with each other, and where there are constraints on that environment such that agents may not at any given time know everything about the environment that other agents know (including the internal states of the other agents themselves). These constraints are crucial elements that distinguish MAS from a centralized system. Because if all agents could synchronize with each other and know exactly the situations of the other agents and what choices they will make, they would act just as if a single master controller manipulates them. In our case, to eliminate the need for a communication link, the information of CTSF, load demand, and renewable power are not guaranteed to be shared among

BSs. The need for solving for an optimal strategy for the agents in MAS has led to two major branches: game theory and multi-agent learning.

Game theory studies the strategy in activities where multiple decision-makers are involved. In a multi-player game, all 'players' are rational and treat every other player as rational too. The players evaluate their rewards/payoffs considering the joint-action made by all players and search for a strategy that maximizes their payoffs, which are also called equilibriums. The equilibriums are sets of strategies where no player could gain a higher payoff by deviating from the equilibrium if other players follow the strategies in equilibrium. The payoff at equilibrium is not necessary the maximized payoff at all circumstances such as the one found in cases like prisoner dilemma, but as long as the other players are playing rationally, it is the best result a player can expect [39]. In Chapter 4, we will discuss details on how microgrid energy management could be modeled as a multi-player game. This game could either be cooperative or competitive, and both have their advantages and disadvantages.

Multi-agent learning has emerged from a separate realm—machine learning, dynamic programming, robotics, evolutionary computation, and complex system [40]. Two key features differ multi-agent learning from conventional machine learning: large searching space due to the multiple agents and complex behavior due to the interaction between agents and their learning process. In this project, the motivation of applying reinforcement learning to BSs is to exempt them from the requirement of historical data (which is necessary for the game approach) and enable them to adapt to changes in the environment. Two reinforcement learning algorithms were applied to the microgrid: Q-learning and linear-reward inaction.

3.0 Multi-player Game

In this section, we will discuss how to model and solve the microgrid energy management as a multi-player game. In this approach, the CTSF decisions are made by BS controllers autonomously. Depending on the operation condition and setting of the game model, the BSs could be either cooperative or competitive. The energy management games under the two conditions are called common-interest or zero-sum games. In this chapter, the multi-player game approach will be introduced in the following steps. First, concepts and notations of the multi-player game are denoted. Then, a microgrid with two BSs or two groups of BSs is discussed. After that, the general game approach with more players scenario is discussed and showed why it might be impractical to be implemented in a BS. Then, a virtual two-player game approach that transfers the general n-player game to a two-player game is proposed and discussed. In the end, a load identifier is designed to adjust the energy consumption model of BS controllers.

3.1 Definition, Notations, and Assumptions

A game in strategy form is an ordered triple

$$G = (N, (S_i)_{i \in N}, (u_i)_{i \in N}) \tag{3-1}$$

where $N = \{1,2, \dots, n\}$ is a finite set of players, S_i is the set of the strategy of player i, for every player $i \in N$. The set of all vectors of strategies is noted by $S = S_1 \times S_2 \times \cdots \times S_n$. u_i is the payoff (utility) function to player *i* related to each vector of strategy S.

A game is a zero-sum game if for each pair of strategies $(S_1, S_2, \dots, S_n)$ one has

$$\sum_{i=1}^{n} u_i = 0 \tag{3-2}$$

Also, a game is called a non-zero-sum game if (3-2) does not hold. Let $N = \{1,2, \dots, n\}$ be a finite set, and for each $i \in N$ let X_i be any set. Denote $X \coloneqq \times_{i \in N} X_i$, and for each $i \in N$ denote $X_{-i} \coloneqq \times_{j \neq i} X_j$. For each $i \in N$ we will denote X_{-i} by the Cartesian product of all the set X_j except for the set X_i. In another word,

$$X_{-i} = [(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) : x_j \in X_j, \forall j \neq i] \tag{3-3}$$

A strategy S_i of player *i* is strictly dominated if there exists another strategy t_i of player i such that for each strategy vector $s_{-i} \in S_{-i}$ of the other players,

$$u_i(S_i, S_{-i}) < u_i(t_i, S_{-i}) \tag{3-4}$$

Also, a strategy is called weakly dominated if there exists another strategy t_i satisfying the following two conditions:

$$u_i(S_i, S_{-i}) \leq u_i(t_i, S_{-i}), \forall s_{-i} \in S_{-i} \tag{3-5}$$

$$u_i(S_i, S_{-i}) < u_i(t_i, S_{-i}), \exists s_{-i} \in S_{-i} \tag{3-6}$$

In a game, all players are assumed to be rational, which indicating

1. A rational player does not use a dominated strategy.

2. The fact that all players are rational is common knowledge among the players.

A strategy vector $s^* = (s_1^*, \dots, s_n^*)$ is an equilibrium if for each player $i \in N$ and each strategy $s_i \in S_i$ the following is satisfied:

$$u_i(s^*) \geq u_i(s_i, s_{-i}^*) \tag{3-7}$$

meaning that when all other players follow the strategy s_{-i}^*, there exists no other strategy than s_i^* that could give player *i* a higher payoff.

3.2 Breaking Down of the Objective Function

Assuming the BS controller only has access to the CTSF choice made by itself, the total system load could be represented by two parts

$$P_{load}{}^{i} = \overbrace{(P_B{}^{i} + \sigma^{i} v^{i} P_T{}^{i})}^{BS\ i\ load} + \overbrace{(P_B{}^{o} + \sigma^{o} v^{o} P_T{}^{o})}^{other\ BSs'\ load} \tag{3-8}$$

where the superscript *i* indicates the *i*th BS load and *o* collectively demonstrates all the other BSs' load. In this study, it is assumed that the information of the base load $P_B{}^{o}$ and traffic dependent load $v^{o} P_T{}^{o}$ are shared among BS controllers, leaving σ^{o} as the only unknown variable.

In a two-player game, the objective functions of the players become $obj_1(\sigma_1, \sigma_2)$ and $obj_2(\sigma_1, \sigma_2)$. Based on the available CTSF range, the objective function values of a BS could be expressed in a tablet form, as shown in Table 2. This table is called the payoff table of the game, and the BS controllers (marked 'Player *I*' and 'Player II') could find its payoff given both BS controllers' CTSF choices [39]. For instance, if the player I choose communications CTSF $\sigma 21$ and player II choose $\sigma 22$, the payoff for player I is $obj_1(\sigma_{21}, \sigma_{22})$.

Table 2 : Payoff table of a two-player game

			Player II	
			P21	P22
			σ21	σ22
Player	P11	σ11	$(obj_1(\sigma_{11}, \sigma_{21}), obj_2(\sigma_{11}, \sigma_{21}))$	$(obj_1(\sigma_{11}, \sigma_{22}), obj_2(\sigma_{11}, \sigma_{22}))$
I	P12	σ12	$(obj_1(\sigma_{21}, \sigma_{21}), obj_2(\sigma_{21}, \sigma_{21}))$	$(obj_1(\sigma_{21}, \sigma_{22}), obj_2(\sigma_{21}, \sigma_{22}))$

3.3 Two-player Zero-sum Game

First, assuming there are only two BSs in the system, or that there are two groups of coordinated BSs that lost communication links to the other group. Then the CTSF decision-making process could be modeled as a two-player game. In this section, we will discuss modeling the CTSF decision-making process as a zero-sum game.

In a zero-sum game, the BS controllers assume they are in a hostile environment. In this environment, the BS controllers assume the worst case where the other BS is minimizing its payoff. This game has two possible solution forms: pure solution or a mixed solution.

3.3.1 The Pure Strategy of a Two-player Zero-sum Game

A pure strategy could be obtained if there exists a single dominating strategy. For example, in a game as shown in Table 3, the payoff of Player I choosing strategy T is always higher than the one choosing strategy B. Therefore, the strategy B is called 'dominated' by strategy T. Thus, the player *I* would never choose B assuming he is rational.

Table 3 : Strategy table of a game with pure strategy solution

		Player II	
		L	**R**
Player I	**T**	**(3, -3)**	**(2, -2)**
	B	**(1, -1)**	**(0, 0)**

Table 4 : Strategy table of a game with pure strategy solution after elimination

		Player II		
		L	**M**	**R**
Player I	**T**	**(3, -3)**	**(2, -2)**	**(2, -2)**
	D	**(1, -1)**	**(0, 0)**	**(3, -3)**

However, for players with more strategies, such as the microgrid in our study, the dominating strategy might not be solved by simple comparison. For example, in a game shown in Table 4, the original dominating strategy T could lead to a loss—a negative payoff. In this game, the player *I* needs to consider the other player's motivation: if player II knows the payoff table as the player *I* does, since player I's payoff is his cost, he will try to avoid the high payment (Choosing R results in a minimum cost of 2). Therefore, knowing player II would not choose strategy R, the player *I* could eliminate column R and transfer the game into the same one shown in Table 3. Then, T is still the dominating strategy.

3.3.2 Mixed Strategy Solution of a Two-player Zero-sum Game

In some games, there may not always be pure solutions. For example, in a game shown in Table 5, both players have no dominating strategies since any strategy has a chance of getting a negative payoff. Based on the nature of the game (zero-sum), the player might be unwilling to broadcast its choice of strategy to other players in this game. Once player *II* knows what player *I* would choose, the optimal response is clear: R if the player *I* plays T and L if the player *I* plays D. Therefore, in this game, the reasonable strategy for each player is not playing one move but a series of probabilities choosing each one. Because now the players need to consider expected rather than

definite payoff. For example, in the game shown in Table 5, if player I claims to choose the strategy T at a probability of $\frac{3}{8}$ and D at a probability of $\frac{5}{8}$, the expected payoff to player II playing L and M are then

$$u_L = \frac{3}{8} \times 3 + \frac{5}{8} \times (-1) = \frac{1}{2} \tag{3-9}$$

$$u_M = \frac{3}{8} \times (-2) + \frac{5}{8} \times 2 = \frac{1}{2} \tag{3-10}$$

Thus, the expected payoff for player II is the same no matter what strategy it applies. This condition is also denoted as 'indifference principle' and will be discussed in section 3.3.2.2. Generally, let $G = (N, (S_i)_{i \in N}, (u_i)_{i \in N})$ be a strategic-form game in which the number of players and strategies are finite, there always exists an equilibrium in mixed strategy for each player [41], denoted by

$$\Sigma_i := \left\{ p_i : S_i \to [0,1] : \sum_{s_i \in S_i} (p_i) = 1 \right\} \tag{3-11}$$

Table 5 : Strategy table of a game with no pure strategy solution

		Player II	
		L	**R**
Player I	**T**	**(3, -3)**	**(-2, 2)**
	D	**(-1, 1)**	**(2, -2)**

3.3.2.1 Direct Approach

The direct approach to finding equilibria in mixed strategy is to write down the mixed extension of the strategic-form game and compute the equilibria in the mixed extension. In the case of a two-player game where each player has two pure strategies, the mixed extension is a game over the unit square with bilinear payoff functions.

Consider the game in Table 5, If Player I plays the mixed strategy [x(T), (1 − x)(D)] where x is the probability choosing T, his payoff, as a function of x, depends on the strategy of Player II:

- If player II plays L: u(x,L)=3x-(1-x)=4x-1
- If player II plays R: u(x,R)=-2x+2(1-x)=-4x+2

Figure 12 shows these two functions. The thick red line plots the function representing the minimum payoff that Player I can receive if he plays x. This minimum is called the lower envelope of the payoffs of Player I. And player I want to maximize its minimal expected payoff indicated by the lower envelope, which is attained at the intersection point of the two corresponding lines appearing in Figure 12, i.e., at the point at which

$$4x - 1 = -4x + 2 \tag{3-12}$$

whose solution is $x = \frac{3}{8}$. And the expected payoff u= $\frac{1}{2}$ is called the value of the game [39].

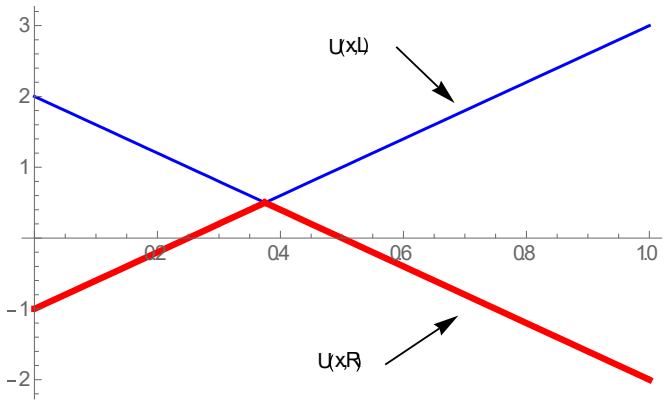

Figure 12: The payoff function of Player I and the lower envelope

A similar calculation for finding Player II's optimal strategy could be done aimed at finding the strategy σ_{II} at which the minmax of $min_{s_{II} \in S_{II}} u(s_I, \sigma_{II})$ is attained. For each one of the pure strategies T and D of Player I, we compute the payoff as a function of the mixed strategy y of Player II and look at the upper envelope of these two lines (see Figure 13). The minimum of the upper envelope is attained at the point of intersection of these two lines. It is the solution of the equation 5y - 2 = 2 − 3y, which is y = 0.5.

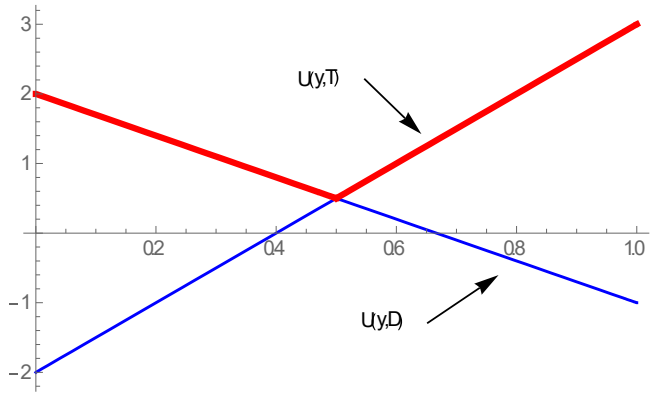

Figure 13: The payoff function of Player II and the higher envelope

The graphical procedure can be extended to games in which one of the players has two pure strategies, and the other player has any finite number of strategies. Thus, it is possible to design the microgrid communication CTSF decision as a two-player game where one of the players has two options. The values of the shaping factors are common knowledge to both controllers. Then, using the direct approach, the controllers would be able to compute their maximized minimum expectation of the payoff, which according to the deduction in this section, being optimal. However, this setting limits the number of CTSF a BS can have. To expand this solution to a game with more variables, the indifference principle is introduced.

3.3.2.2 Indifference Principle

The indifference principle claims that if a mixed equilibrium calls for a player to use two or more distinct pure strategies with positive probability, then the expected payoff to that player for using one of those pure strategies equals the expected payoff to him for using any other pure strategy, assuming that the other players are playing according to the equilibrium [39, 42]. Suppose the equilibrium of a mixed game is σ^*, and s_i and $\widehat{s_i}$ are two strategies of player i. Then, if $\sigma_i^*(s_i) > 0$ and $\sigma_i^*(\widehat{s_i}) > 0$, then

$$\mathrm{u}(s_i, \sigma_{-i}^*) = \mathrm{u}(\widehat{s_i}, \sigma_{-i}^*) \tag{3-13}$$

Therefore, we could use this property to solve for each player's strategy using his opponent's utility values. For example, in a game shown in Table 6, the strategy at the equilibrium point for player *I* could be obtained by solving:

$$p_T + p_B + p_D = 1$$

$$u_{II}(L) = u_{II}(M) \tag{3-14}$$

$$u_{II}(M) = u_{II}(R)$$

Table 6 : Strategy table of a game where players have three pure strategies

		Player II		
		L	**M**	**R**
Player I	**T**	**(3, -3)**	**(2, -2)**	**(-2, 2)**
	B	**(1, -1)**	**(0, 0)**	**(2, -2)**
	D	**(-2, 2)**	**(1, -1)**	**(5, -5)**

where

$$u_{II}(L) = p_T \times (-3) + p_B \times (-1) + p_D \times 2u_{II}(L) = u_{II}(M) \tag{3-15}$$

$$u_{II}(M) = p_T \times (-2) + p_B \times 0 + p_D \times (-1) \tag{3-16}$$

$$u_I(D) = p_T \times (2) + p_B \times (-2) + p_D \times (-5) \tag{3-17}$$

And the solution is

$$p_T = \frac{5}{12}, p_B = \frac{1}{3}, p_D = \frac{1}{4} \tag{3-18}$$

which makes the player II 'indifferent' with his choices. The general equations for solving player I's equilibrium solution are listed:

$$\sum_{i=1}^{N} p_i = 1\,, 0 \le p_i \tag{3-19}$$

$$\sum\nolimits_{i=1}^{N} (p_i\, u_{II}(\sigma_I^i, \sigma_{II}^k)) = \sum\nolimits_{i=1}^{N} (p_i\, u_{II}(\sigma_I^i, \sigma_{II}^{\hat{k}})), \forall k, \hat{k} \in N[1, M] \tag{3-20}$$

where M and N is the number of available strategies for player II and player I. The solution of eqn. (3-19) and (3-20) could be expressed in a matrix form. After eliminating the dominated strategies, the payoff table is denoted as

$$A = \begin{pmatrix} U_{11} & \cdots & U_{1n} \\ \vdots & \ddots & \vdots \\ U_{m1} & \cdots & U_{mn} \end{pmatrix}_{m \times n} \tag{3-21}$$

where m and n are the numbers of strategies of players. The probability of player I playing each strategy is written as a vector Pr_1

$$Pr_1 = (Pr_1 \quad \cdots \quad Pr_m)_{1 \times m} \tag{3-22}$$

So the expected payoff of player II is the dot product of Pr and A as shown in

$$Pr_1 \cdot A = \left(\sum_{i=1}^{n} Pr_i U_{i1} \quad \dots \quad \sum_{i=1}^{n} Pr_i U_{in} \right)_{1\times n} \tag{3-23}$$

Based on eqn. (3-20), each entry in eqn. (3-23) is identical as shown in eqn. (3-24)

$$Pr_1 \cdot A = \alpha \cdot (1 \quad 1 \quad \cdots \quad 1)_{1\times n} \tag{3-24}$$

where α is an unknown scalar. Multiplying both sides of eqn. (3-23) by A^T and assuming AA^T is invertible, Pr_1 could be expressed as

$$Pr_1 = \alpha \cdot (1 \quad 1 \quad \cdots \quad 1)_{1\times n} A^T \cdot (AA^T)^{-1} \tag{3-25}$$

Also, eqn. (3-19) could be rewritten as

$$Pr_1 \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{m\times 1} = 1 \tag{3-26}$$

Then, α could be solved by substituting eqn. (3-25) into (3-26):

$$\alpha = \frac{1}{(1 \quad 1 \quad \cdots \quad 1)_{1\times n} \cdot A^T \cdot (AA^T)^{-1} \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{m\times 1}} \tag{3-27}$$

Substitute eqn. (3-27) to (3-25) and the probability vector Pr is

$$Pr_1 = \frac{\mathrm{R} \cdot A^T \cdot (AA^T)^{-1}}{R \cdot A^T \cdot (AA^T)^{-1} \cdot \mathrm{C}} \tag{3-28}$$

where R=$(1 \quad 1 \quad \cdots \quad 1)_{1\times n}$ and C=$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{m\times 1}$.

After solving Pr for both players, the expected utility could be calculated as

$$E(u_I) = Pr_1 \cdot A \cdot {Pr_2}^T \tag{3-29}$$

Practically, it is computationally inefficient to solve eqn. (3-28) because it requires the calculating of matrix inverse and eliminating of dominated strategies, whose complexities are at least O($n^{2.3}$) and NP-hard [3, 10]. Even before that, the payoff matrix may not be invertible. Therefore, eqn. (3-28) is only calculated at a single time step during the simulation to validate the performance of the two-player game approach. In the rest of the simulation, instead of computing eqn. (3-28), the mixed strategies of a BS are computed using a numerical algorithm by transferring the two-player zero-sum game to a linear programming problem.

3.3.2.3 Linear Programming

Because player II's goal is to minimize layer I's payoff, the expected payoff of player I becomes

$$E(u_I) = \min_{1\le k\le M} \sum_{i=1}^{N} (p_i\, u_I(\sigma_I^i, \sigma_{II}^k)) \tag{3-30}$$

Thus, from player I's point of view, he wants to choose $p_1, p_2, \dots, p_N$ such that his payoff is maximized:

$$maximize \min_{1\le k\le M} \sum_{i=1}^{N} (p_i\, u_I(\sigma_I^i, \sigma_{II}^k)) \tag{3-31}$$

$$\text{st.} \sum_{i=1}^{N} p_i = 1\,, 0 \le p_i \le 1 \tag{3-32}$$

Eqn. (3-31) is not a linear programming problem because the *min* function is not linear. Then, by introducing a new variable z and restrict it to be less than the objective function $v \le \min_{1\le k\le M} \sum_{i=1}^{N} (p_i\, u_I(\sigma_I^i, \sigma_{II}^k))$, the problem becomes

$$maximize\ z \tag{3-33}$$

$$\begin{aligned} \text{st.}\ z &\le \sum_{i=1}^{N} (p_i\, u_I(\sigma_I^i, \sigma_{II}^1)) \\ z &\le \sum_{i=1}^{N} (p_i\, u_I(\sigma_I^i, \sigma_{II}^2)) \\ &\vdots \\ z &\le \sum_{i=1}^{N} (p_i\, u_I(\sigma_I^i, \sigma_{II}^M)) \\ \sum_{i=1}^{N} p_i &= 1\,, 0 \le p_i \end{aligned} \tag{3-34}$$

So this problem is a linear programming problem that is identical to eqn. (3-31). For a two-player zero-sum game, computing the equilibrium, where each player has a finite number of strategies, can always be presented as a linear programming problem [43]. The fact that the value of a game in mixed strategies can be found using linear programming is based on the connection between the Minmax Theorem and the Duality Theorem. These two theorems are equivalent to each other in a zero-sum setting [43-45].

To solve the linear programming problem, it needs to be converted to standard form. The standard form of linear programming is

$$\begin{aligned} &minimize\ \boldsymbol{c}^T\boldsymbol{x} \\ &sbject\ to\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\ and\ \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{3-35}$$

where x is an n-dimension column vector, $\boldsymbol{c}^T$ is an n-dimension row vector, $\boldsymbol{A}$ is a $m \times n$ matrix, and $\boldsymbol{b}$ is an m-dimension column vector. All linear programming problems could be converted into this standard form. However, in the MATLAB linear programming solver, the equalities are replaced to inequalities:

$$\begin{aligned} &minimize\ \boldsymbol{c}^T\boldsymbol{x} \\ &sbject\ to\ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\ and\ \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{3-36}$$

Numerical algorithms for solving Linear programming problems have been heavily studied and well understood. Still, there are several features of this problem worth mentioning. Firstly, a general linear programming problem has no analytical solution. Secondly, in principle, the time required to solve for a linear programming problem might be an exponential function of the number of variables, which could happen in some contrived cases. However, in practice, either

using the simplex method or interior-point methods (the two most popular solving algorithms), the computation is highly efficient, usually a small multiple of the number of the variables [44]. As a comparison, the quadric programming problem applied in the common-interest-sum game next section usually requires an exponential computation time. In the simulation, the zero-sum game solution is obtained by solving eqn. (3-36) using the lingprog function in MATLAB, and the solver applied is the interior-point algorithm. The form of LP that the solver needs is

$$\min_{x} \boldsymbol{f}^T \boldsymbol{x} \;\; such\ that \begin{cases} \boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b} \\ \boldsymbol{Aeq} \cdot \boldsymbol{x} \leq \boldsymbol{beq} \\ \boldsymbol{lb} \leq \boldsymbol{x} \leq \boldsymbol{ub} \end{cases} \tag{3-37}$$

where $\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{b}, \boldsymbol{beq}, \boldsymbol{lb}$ and $\boldsymbol{ub}$ are vectors, and $\boldsymbol{A}$ and $\boldsymbol{Aeq}$ are matrices. And the original LP problem eqn. (3-33)-(3-34) need to be converted into the form of eqn. (3-37). First, the variable vector is the same:

$$\boldsymbol{x} = [x(s_1), x(s_2), \dots, x(s_N), z] \tag{3-38}$$

Then the inequality constraints could be expressed as

$$\begin{bmatrix} u({s_I}^1, {s_{II}}^1) & \cdots & u({s_I}^N, {s_{II}}^1) \\ \vdots & \ddots & \vdots \\ u({s_N}^1, {s_{II}}^N) & \cdots & u({s_I}^N, {s_{II}}^N) \end{bmatrix} \begin{bmatrix} x(s_1) \\ \vdots \\ x(s_N) \end{bmatrix} \geq \begin{bmatrix} z \\ \vdots \\ z \end{bmatrix} \tag{3-39}$$

which could be converted to

$$\begin{bmatrix} -{A_{pay}}^T & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \end{bmatrix} \begin{bmatrix} x(s_1) \\ \vdots \\ x(s_N) \\ z \end{bmatrix} \leq \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3-40}$$

where A_{pay} is the payoff matrix/strategy form table. Therefore,

$$\boldsymbol{A} = \begin{bmatrix} -A_{pay}{}^{T} & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3-41}$$

And the equality constraint is expressed as

$$[1 \quad \dots \quad 1 \quad 0] \begin{bmatrix} x(s_1) \\ \vdots \\ x(s_N) \\ z \end{bmatrix} = 1 \tag{3-42}$$

Thus,

$$\boldsymbol{Aeq} = [1 \quad \dots \quad 1 \quad 0], \boldsymbol{beq} = 1 \tag{3-43}$$

Then, the CTSF probability, which is the first *N* variables in vector $\boldsymbol{x}$, could be solved using linprog function in MATLAB. The overall flowchart of the zero-sum game solving process is shown in Figure 14.

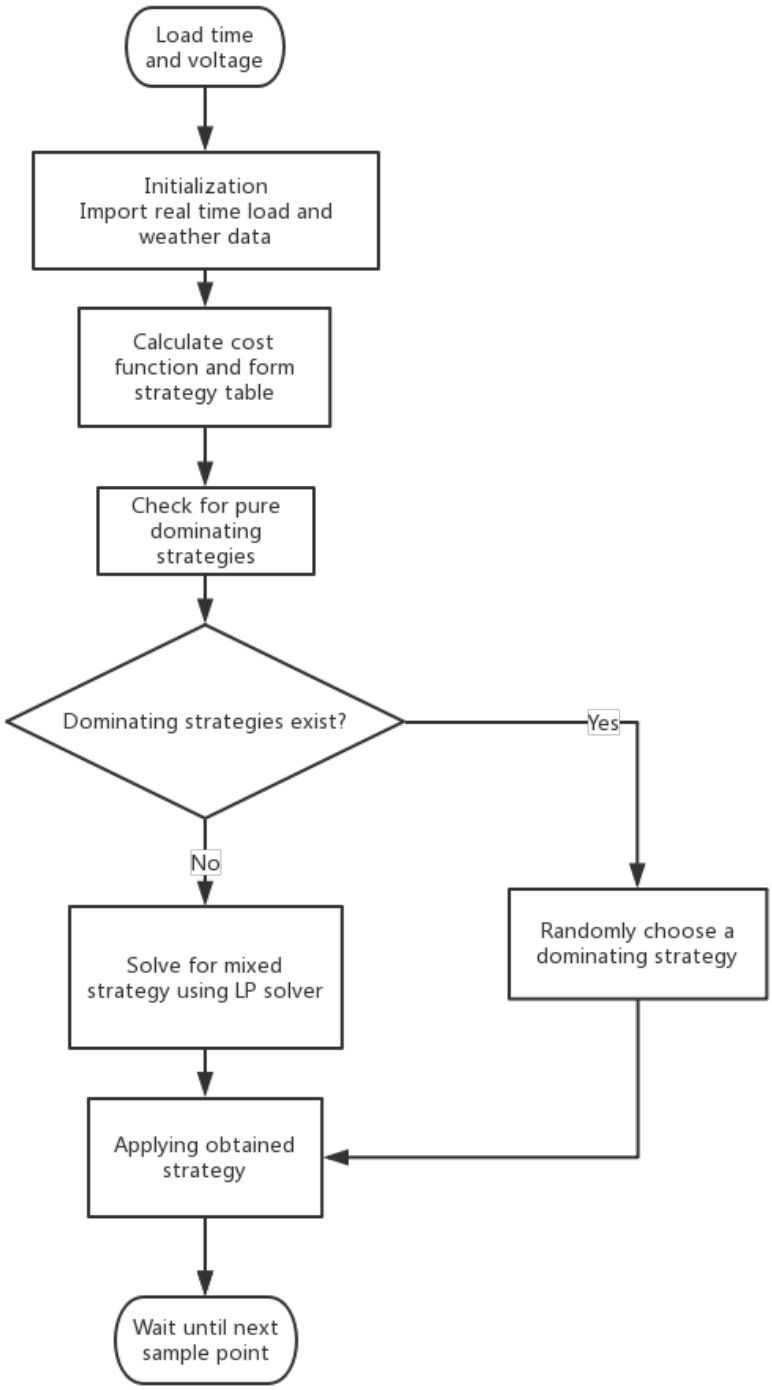

Figure 14: Zero-sum two-player game solving flowchart

3.4 Two-player Common-interest Game

If the original objective function is applied as the same utility function for all BSs, meaning

$$obj_i(\sigma_1, \sigma_2, \cdots, \sigma_N) = \text{obj}(\sigma_1, \sigma_2, \cdots, \sigma_N), \forall i, j \in [1, N] \tag{3-44}$$

then this game becomes a common-interest game, in which all players share the same objective: to maximize $\text{obj}(\sigma_1, \sigma_2, \cdots, \sigma_N)$. However, because no communication is guaranteed between players, they still need to estimate their payoff assuming the other player's moves. The common-interest game could also be solved using the indifference principle. But considering the computation cost again, it is solved numerically as a quadratic programming problem. Suppose the payoff matrix of a BS is *A*, then the objective of a BS is

$$\begin{aligned} &\textit{Maximize} && x'Ay \\ &\textit{Subject to} && e'x - 1 = 0 \\ & && x \geq 0 \end{aligned} \tag{3-45}$$

where x and *y* are the strategy vectors of player I and player II, *e* is a vector of ones with the same dimension of *x,* and the primes denote transpose. *F*or player II, its objective is

$$\begin{aligned} &\textit{Maximize} && x'By \\ &\textit{Subject to} && l'y - 1 = 0 \\ & && y \geq 0 \end{aligned} \tag{3-46}$$

where B is the player's payoff table, l is a vector of ones with the same dimension of y. A Nash equilibrium point (x^0, y^0) is defined as a pair of strategies x^0 and y^0 where (3-47) and (3-48) are simultaneously fulfilled. In a more precise way,

$$x^{0\prime}Ay^0 = \max_x\{x'Ay^0 | e'x - 1 = 0, \qquad x \geq 0\} \tag{3-47}$$

$$x^{0\prime}By^0 = \max_x\{x^{0\prime}By | l'y - 1 = 0, \qquad y \geq 0\} \tag{3-48}$$

According to Karush–Kuhn–Tucker (KKT) conditions, the necessary and sufficient conditions for (x^0, y^0) to be the optimal solution are

$$\begin{gathered} x^{0\prime}Ay^0 - \lambda_A = 0 \\ x^{0\prime}By^0 - \lambda_B = 0 \\ Ay^0 - \lambda_A e = 0 \\ B'x^0 - \lambda_B l = 0 \\ e'x - 1 = 0 \\ l'y - 1 = 0 \\ x^0 \geq 0 \\ y^0 \geq 0 \end{gathered} \tag{3-49}$$

where λ_A and λ_B are the expected payoff to player I and II respectively at equilibrium (x^0, y^0). According to the equivalence theorem in [46], the necessary and sufficient condition that (x^0, y^0) is an equilibrium is that it is a solution of the following programing problem

$$\begin{aligned} maximize \quad & x'(A+B)y - \lambda_A - \lambda_B \\ \text{Subject to} \quad & Ay - \lambda_A e = 0 \\ & B'x - \lambda_B l = 0 \\ & e'x - 1 = 0 \\ & l'y - 1 = 0 \\ & x^0 \geq 0 \\ & y^0 \geq 0 \end{aligned} \tag{3-50}$$

The values of λ_A and λ_B at the equilibrium, ${\lambda_A}^0$ and ${\lambda_B}^0$, equal the expected payoff to player I and the player II respectively. Also,

$$x^{0\prime}(A+B)y^0 - {\lambda_A}^0 - {\lambda_B}^0 = 0 \tag{3-51}$$

Since the payoff matrix is identical to all players, *A*=*B*, the problem becomes a special scenario of the general quadratic problem:

$$\begin{aligned} maximize \quad & x'(2A)y - \lambda_A - \lambda_B \\ Subject \text{ to} \quad & Ay - \lambda_A e = 0 \\ & A'x - \lambda_B l = 0 \end{aligned} \tag{3-52}$$

$$e'x - 1 = 0$$

$$l'y - 1 = 0$$

$$x^0 \geq 0$$

$$y^0 \geq 0$$

Therefore, if the common-interest game setting is applied, the BSs compute (3-52) and apply the CTSF strategy obtained by solving this quadratic programming problem. The solver applied in the simulation is fmincon in MATLAB with the interior-point algorithm.

3.5 Discussion on the Game Types

As illustrated in the above sections, the microgrid energy management could either modeled as a zero-sum game or a common-interest game. Both setting have their advantages and drawbacks. So they may be applied under different circumstances.

By modeling the CTSF decision-making process as a zero-sum game, the BSs are operating in a 'safe mode.' Since there is a significant penalty when the CTSF is large, the BS will likely keep low CTSF most of the time, except when the probability of reaching the desired SoC level is sufficiently high. In this mode, the system is expected to have a conservative energy plan, which could be desired during a natural disaster. Another benefit is that the computation cost is almost linear to the number of variables in the problem so that the BS controllers could handle a relatively large number of CTSF choices [44].

On the other hand, by modeling the energy management as a common-interest game, the BSs obtain higher payoffs. As will be shown in the numerical results, the common-interest game setting encourages the BSs to choose higher CTSF without missing the energy availability target. However, the computation cost of a non-zero-sum game could be high. Also, the common-interest game has a large performance drop when applied in a virtual two-player game as will be discussed in section 3.7.

Generally, if the BS controllers have sufficient computing power and the communication QoS is not negotiable, a common-interest game would be an optimal arrangement. Otherwise, modeling energy management as a zero-sum game could be a more cost-efficient solution.

3.6 Games with Arbitrary Number of Players

So far, we have been working on a two-player game, which means the microgrid has only two BSs, or only two groups of BSs loss the connection between them. More often, there may be more than dozens of BSs in a communication network, and the disconnected BSs could be of any number. Therefore, a solver for a game with an arbitrary number of BSs is needed. However, there might not always be an effective method to solve them. Unlike the two-player game, a game with more players cannot be converted to a linear programming or quadratic programming problem. For a general n-player game, utilizing the indifference principle, the KKT condition yields a set of n-1 order equations. This phenomenon could be seen from the following example. Suppose a three-player game has a payoff table shown in Table 7, where each player has two available actions *L* and *R*. Denote the probabilities of players' strategies as

Table 7: Example of a three-player non-zero-sum game

Player Z L (P3)					Player Z R (1-P3)				
			Player Y P2	Player Y 1-p2				Player Y P2	Player Y 1-P2
Player X			L	R	Player X			L	R
	P1	L	3, -6 ,4	3,3, 2		P1	L	3, -2, 0	2, 2, 2
	1-P1	R	5, 1, 1	2, 3, 2		1-p1	R	2, 2, 2	3, 1, -2

$$(s_X = L) = p1, P(s_X = R) = 1 - p1 \tag{3-53}$$

$$P(s_Y = L) = p2, P(s_Y = R) = 1 - p2 \tag{3-54}$$

$$P(s_Z = L) = p3, P(s_Z = R) = 1 - p3 \tag{3-55}$$

And the expected payoff of their choice of actions are

$$E(s_X = L) = \mathrm{p3}(3\mathrm{p2} + 3(1 - \mathrm{p2})) + (1 - \mathrm{p3})(3\mathrm{p2} + 2(1 - \mathrm{p2})) \tag{3-56}$$

$$E(s_X = R) = \mathrm{p3}(5(\mathrm{p2}) + (1 - \mathrm{p2})2) + (1 - \mathrm{p3})(2\mathrm{p2} + 3(1 - \mathrm{p2})) \tag{3-57}$$

$$E(s_Y = L) = \mathrm{p3}(-6\mathrm{p1} + 3(1 - \mathrm{p1})) + (1 - \mathrm{p3})(-2\mathrm{p1} + 2(1 - \mathrm{p1})) \tag{3-58}$$

$$E(s_Y = R) = \mathrm{p3}(1(\mathrm{p1}) + 3(1 - \mathrm{p1})) + (1 - \mathrm{p3})(2\mathrm{p1} + 2(1 - \mathrm{p1})) \tag{3-59}$$

$$E(s_z = L) = \mathrm{p1}(4\mathrm{p2} + 2(1 - \mathrm{p2})) + (1 - \mathrm{p1})2\mathrm{p2} \tag{3-60}$$

$$E(s_z = R) = \mathrm{p1}(1\mathrm{p2} + (1 - \mathrm{p2})2) + (1 - \mathrm{p1})(2\mathrm{p2} - 2(1 - \mathrm{p2})) \tag{3-61}$$

According to the indifference principle, at the equilibrium, the expected payoffs of each player's different actions should be equal:

$$E(s_X = L) = E(s_X = R) \tag{3-62}$$

$$E(s_Y = L) = E(s_Y = R) \tag{3-63}$$

$$E(s_Z = L) = E(s_Z = R) \tag{3-64}$$

which is a ternary quadratic equation set, whose solution is

$$p1 = -10, p2 = \frac{11}{26}, p3 = -\frac{4}{3}\ (infeasible)$$

$$or \quad p1 = 0, p2 = 1, p3 = \tfrac{1}{3}$$

So, an equilibrium of the three-player game is obtained. As the reader could image, the order of the equation set (3-53)-(3-55) increases linearly as the number of players increases. Since there is no algebraic solution to the general polynomial equations of degree five or higher according to the Abel-Ruffini theorem, it is only possible to approach the equilibrium through numerical algorithms such as Newton's method [33, 35]. However, the computation cost of such an algorithm could be high. The computation time of a game with multiple players using global Newton algorithm is shown in Table 8, as the results suggest, the computation time of a game with more than three players could go beyond hours [47].

Table 8 : Computation time in second required to solve for equilibrium with global Newton algorithm

$\|N\|$ m_n :	2	4	6	8	10	12	14	16
2	0.0070	0.0140	0.0370	0.0791	0.2373	0.3281	0.5358	1.2820
3	2.2530	8.9950	33.0900	143.1000	251.2000	536.6000	853.3000	1471.0000
4	3.6500	25.8200	144.5000	469.2000	854.9000			
5	10.4200	126.3000	1116.0000	9745.0000				
6	19.6700	315.9000						

Generally, computing the NE in a game with more than three players has yet to be proved that it could be solved effectively [48]. The computation time and complexity of communication increase exponentially as the number of players and actions increases even for solving an approximate equilibrium [21, 49]. Therefore, in this project, instead of developing an exact solver for a multi-player game with more players, we made a simplification transition and modified the original multiplayer energy management game into a virtual two-player game.

3.7 Virtual Two-player Game

In a system with more than two BSs, the power consumption is

$$P_{load} = \sum_{i=1}^{N} ({P_B}^i + \sigma^i v^i {P_T}^i) \tag{3-65}$$

Recall that in a two-player game, the load estimation of BS i is:

$${P_{load}}^i = \overbrace{({P_B}^i + \sigma^i v^i {P_T}^i)}^{BS\ i\ load} + \overbrace{({P_B}^o + \sigma^o v^o {P_T}^o)}^{other\ BSs'\ load} \tag{3-66}$$

In the view of BS i, the power consumption of all the other BSs could be represented by one BS and its CTSF σ^o

$${P_B}^o + \sigma^o v^o {P_T}^o = \sum_{j \neq i}^{N} ({P_B}^j + \sigma^j v^j {P_T}^j) \tag{3-67}$$

Suppose the BS base load information is known to all BSs, the actual transition is

$$\sigma^o v^o {P_T}^o = \sum_{j \neq i}^{N} (\sigma^j v^j {P_T}^j) \tag{3-68}$$

where $v^o {P_T}^o = \sum_{j \neq i}^{N} (v^j {P_T}^j)$ is the imagined load consumption of all the other BSs. Then the controller could use the two-player game solving algorithms to solve for the equilibrium. The

performance of a small microgrid implemented with this approach has a performance that is close to an exhaustive searching, as will be shown in the numerical results section. Theoretically, the CTSF strategy obtained by this virtual two-player game with zero-sum setting guarantees a maximized worst-case payoff of the BS. However, because the other BSs are not practically minimizing each other's payoffs, the actual payoff is always higher than the worst-case expectation.

However, as the number of BSs (players) increases, from the view of a BS controller, the response of the microgrid becomes more unpredictable. Because the virtual player can not fully represent the joint-actions of multiple players. As a consequence, as the number of BSs increases, the strategy obtained using this virtual two-player game becomes less optimal. This phenomenon could be observed in an extreme case where the load ratio of one BS is approaching 0, meaning

$$P_{load} \approx \overbrace{({P_B}^o + \sigma^o v^o {P_T}^o)}^{other\ BSs'\ load} \tag{3-69}$$

Here, the choice of CTSF of player *i* does not affect the total power consumption hence the energy availability function $f_{av}(\sigma, SoC, t)$. And the objective function of player *i* becomes

$$obj(\sigma, SoC, t) = w_{com} \cdot f_{com}(\sigma, t) + c \tag{3-70}$$

where the constant *c* is

$$c = w_{SoC} \cdot f_{av}(\sigma^o, SoC, t) \tag{3-71}$$

Because the communication quality function is a positive-related function of CTSF σ, the objective function of player *i* then becomes a monotonic function. Because the final objective function is piece-wised, as shown in eqn. (3-72)

$$obj^* = \begin{cases} Sum(obj), & P(SoC > thresohold) = f_{av}(\sigma, SoC, t) > P_{min} \\ (1-\sigma)/10, & otherwise \end{cases}, \tag{3-72}$$

So the optimal solutions for the player could are:

$$\sigma^{i^*} = \begin{cases} {\sigma^o}_{max}, & P(SoC > thresohold) > P_{min} \\ {\sigma^o}_{min}, & otherwise \end{cases} \tag{3-73}$$

where

$$P(SoC > thresohold) = f_{av}(\sigma, SoC, t) = f_{av}(\sigma^o, SoC, t) \tag{3-74}$$

which is only decided by the CTSF choice of the other player σ^o. Then, depending on whether the energy management is modeled as a zero-sum or common-interest game, the optimal CTSF varies:

- Zero-sum game:
    - If there exists σ^o such that $P(SoC > thresohold) < P_{min}$, the player assumes the other player would choose the maximal CTSF so that the objective function $(1-\sigma)/10$ is minimized. In this condition, its optimal strategy is $\sigma^{i^*} = {\sigma^i}_{min}$.

- If there is no σ^o such that $P(SoC > thresohold) < P_{min}$, the player assumes the other player would choose the minimal CTSF so that the objective function $obj(\sigma, SoC, t) = w_{com} \cdot f_{com}(\sigma, t) + w_{SoC} \cdot f_{av}(\sigma^o, SoC, t)$ is minimized. In this condition, its optimal strategy is $\sigma^{i^*} = \sigma^i{}_{max}$.

- Common-interest game:

  - If there exists σ^o such that $P(SoC > thresohold) > P_{min}$, the player assumes the other player would choose the proper CTSF so that the objective function form is still $obj(\sigma, SoC, t) = w_{com} \cdot f_{com}(\sigma, t) + w_{SoC} \cdot f_{av}(\sigma^o, SoC, t)$. In this condition, its optimal strategy is $\sigma^{i^*} = \sigma^i{}_{max}$.

  - If there is no σ^o such that $P(SoC > thresohold) > P_{min}$, the objective function has the form, $(1 - \sigma)/10$. Then the player assumes the other player would choose the minimal CTSF. In this condition, the optimal strategy is $\sigma^{i^*} = \sigma^i{}_{min}$.

As these conditions show, the BS controllers with small load-ratios would only choose either the maximal or the minimal CTSF under all circumstances. Moreover, this behavior applies to all the BSs in such a system. So, the overall behavior of this system will be like an over-fitted controller with a huge gain: the system CTSF switches from maximum to minimum based on the condition whether 'there exists σ^o such that $P(SoC > thresohold) > P_{min}$', and resulting in the same switching in PSNR. As a result, the system might not always reach the desired SoC level, which causes a performance drop measured in the objective function. A portrait showing the performance descending of the virtual two-player zero-sum game as the number of player increases

is shown in Figure 15. As this figure suggests, the virtual two-player game experiences a 40% performance loss as the number of players reaches 20. This performance drop is also related to the system configuration, mostly on weighting factors, which will be shown in the simulation section.

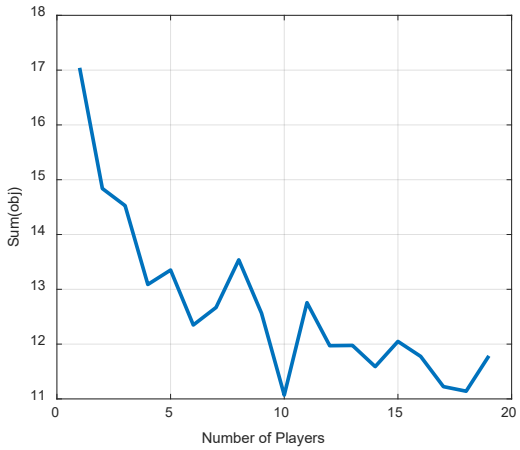

Figure 15: Performance of virtual two-player zero-sum game with more players, w_soc=0.2

Nonetheless, the virtual-two player approach leaves a clue-- the players' 'misunderstanding' of the game is one of the reasons causing this performance drop. Therefore, in the learning-game section, we will discuss how to adjust the BS controller's 'understanding' of its load model with the help of reinforcement learning.

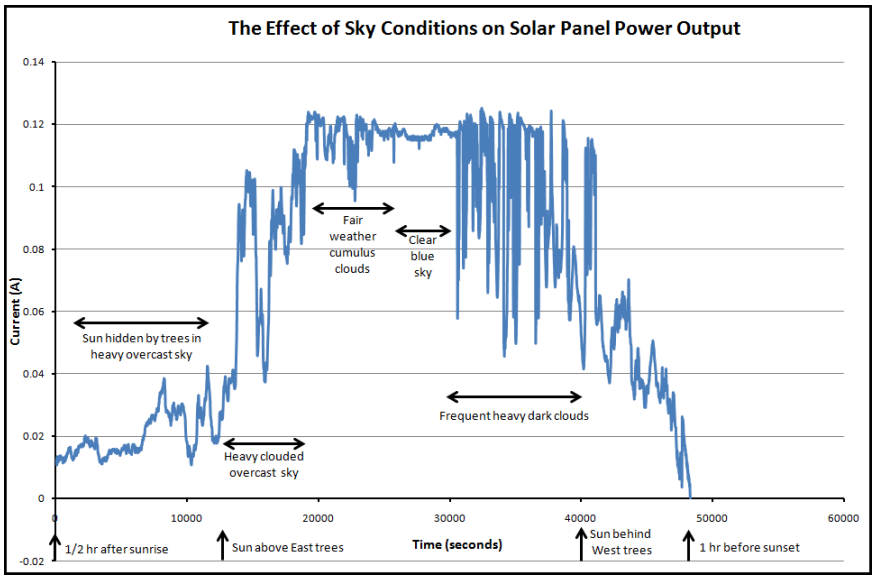

Figure 16: Solar cell output power in a day

3.8 Communication Base Station Load Identifier

In the previous design, there is an assumption that the historical data of BS load and PV power generation are accurate and known to all BSs in the system. However, in reality, these values might be deviant from their expected ones. For example, the output of a solar cell generator could be affected by a lot of conditions: sun irradiation, temperature, and cloud movement to a large extent. The solar cell output power could have significant variance in a daily power output curve as shown in Figure 16 [50]. If the real load and power generation experience large deviations compared to the historical data, the BS could have a wrong estimation on its energy condition. Suppose the real load consumption is higher than the preset data, the BS controller might not tune down its CTSF until the battery SoC is too low when the probability of SoC reaching the desired level is too small. Therefore, a system identifier is introduced to update the power consumption model for BS

controllers such that they can adjust their CTSF strategy faster. The adaptive control scheme is shown in Figure 17.

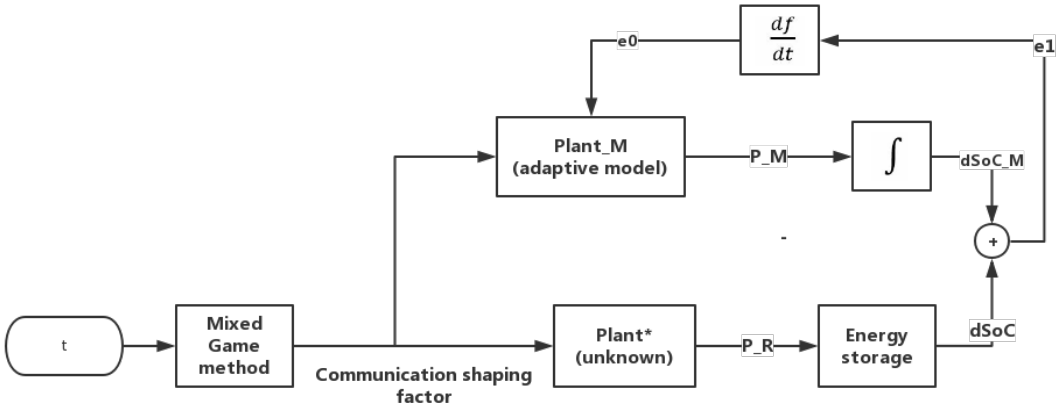

Figure 17: Adaptive controller scheme

From the perspective of the BS controller, changes either in renewable power generation or load consumption both result in changes in the battery power storage:

$$P_R = {P_S}^* - ({P_B}^* + \alpha^* P_L^*) \tag{3-75}$$

where ${P_S}^*$ and ${P_B}^*$, ${P_L}^*$ are the real value of renewable power and load consumption. And a reference model is given to BS controller

$$P_M = {P_S}^M - ({P_B}^M + \sigma P_L^M) + \Delta P_D \tag{3-76}$$

where ${P_S}^M$ and P_L^M are the initial estimations of the PV power output and BS loads; σ is the CTSF and ΔP_D is a tuning variable called virtual load

$$\Delta P_D = \theta P_0 \ \ (P_0 > 0) \tag{3-77}$$

where P_0 is a base power consumption chosen by the system designer and θ is the adjusting factor. Then, the error signal is

$$e_0 = P_R - P_M = ({P_S}^* - ({P_B}^* + \alpha^* P_T^*)) - ({P_S}^M - \alpha P_L^M - {P_B}^M + \theta P_0) \tag{3-78}$$

Also, we define the difference between the desired adjusting factor and current factor to be the parameter error $\emptyset$:

$$\emptyset = \theta^* - \theta \tag{3-79}$$

whereas the desired adjusting factor is

$$\theta^* = \frac{({P_S}^* - \alpha P_L^*) - ({P_S}^M - \alpha P_L^M - {P_B}^M)}{P_0} \tag{3-80}$$

Assume now that the updating rate is much higher than the changing rates of PV power and BS load, then a Lyapunov energy function in which c is an arbitrary constant is defined as

$$V = \frac{1}{2}{e_0}^2 + \frac{1}{2}c\emptyset^2 \geq 0 \tag{3-81}$$

Its partial derivative with respect to time becomes

$$\frac{\partial V}{\partial t} = e_0 \dot{e_0} + c\emptyset\dot{\emptyset} = -e_0 P_0 \dot{\theta} + c\emptyset\dot{\emptyset} = (e_0 P_0 + c\emptyset)\dot{\emptyset} = (-e_0 P_0 + c\emptyset)\dot{\theta} \tag{3-82}$$

From (3-82), the parameter updating rule is proposed

$$\dot{\emptyset} = \dot{\theta} = -(-e_0 P_0 + c\emptyset) \approx e_0 P_0 \tag{3-83}$$

where *c* is chosen to be relatively small so that $c\emptyset$ is negligible and does not need to be measured. Substituting (3-83) into (3-82) and the partial derivative of Lyapunov energy function with respect to time becomes

$$\frac{\partial V}{\partial t} = -(-e_0 P_0 + c\emptyset)^2 \tag{3-84}$$

which is non-positive for all *t*. Therefore, according to the Lyapunov stability theory [51], the adaptation laws (3-83) guarantees error signal e_0 to be bounded [52]. Hence, the system identifier can track the changes caused both by BS loads and the renewable power. Additionally, the quadratic error equation is defined by

$$f_e = \frac{1}{2} {e_0}^2 \geq 0 \tag{3-85}$$

and its derivative with respect to e_0 is

$$\dot{f_e} = e_0\dot{e_0} = -e_0\dot{\theta}P_0 \tag{3-86}$$

Substituting (3-83) into (3-86)

$$\dot{f_e} = -(e_0P_0)^2 \leq 0 \tag{3-87}$$

So that f_e (e_0)>0 for all $e_0 \neq 0$ and $\dot{f_e}(e_0) \leq 0$ for all e_0, $\dot{f_e}(e_0) = 0$ if and on if $e_0 = 0$. By La Salle's invariance principle, the original point $e_0 = 0$ of f_e is asymptotically stable with any initial condition [53]. Thus, the adaptation law eqn. (3-83) guarantees finding the deviation between real and estimated load consumption. However, this updating law requires the information of error signal e_0, which is obtained by measuring the battery SoC as shown in (3-88).

$$e_0 = P_R - P_M = \frac{dSoC}{dt} \cdot E_{battery} - P_M \tag{3-88}$$

Therefore, the controllers in the system must keep monitoring the entire battery SoC to get the error signal. Without communication, this monitoring could be done by measuring the dc bus voltage using the battery SoC balancing control algorithms [49, 54]. In the simulation, the SoC information is given to BSs directly.

4.0 Reinforcement Learning

In this approach, the BS controllers are modeled as 'agents' utilizing reinforcement learning algorithms to obtain a set of energy management strategies. Reinforcement learning (RL) is a general class of algorithms in the field of machine learning that aims at training an agent to learn how to behave in an environment where the only feedback consists of a scalar reward signal [25, 55]. RL suits well in solving long-term decision-making problems such as Markov Decision Process (MDP), which coincidence with the energy management solving process [56, 57]. The essential part of RL implementation is to design the instantaneous reward function properly such that the agents are attracted to the desired behavior pattern. In this section, we will discuss two different kinds of RL algorithms: Q-learning and Linear Reward Inaction. In the end, a combined algorithm using Linear Reward Inaction and the two-player game is proposed which aims at reducing the training time.

4.1 Markov Decision Process

A (finite) Markov decision process is a tuple $\langle X, U, f, \rho \rangle$ where X is the finite set of environment states, U is the set of agent actions, $f: X \times U \times X \to \mathbb{R}$ is the reward function. A state $x_k \in X$ denotes the environment at each time step k. The agent observes the state and takes an action $u_k \in U$. As a result, the environment changes its state to another state $x_{k+1} \in X$ according to the transition function f, which tells the probability of reaching different states after u_k. Then, the agent receives a scalar reward $r_{k+1} \in \mathbb{R}$ according to the reward function $r_{k+1} =$

$\rho(x_k, u_k, x_{k+1})$, which is an immediate effect of the action just taken. However, this reward does not contain any long-term effect of this action. For some MDP, there is a terminal state, which ends the process and the agent cannot leave this state. In our case, the MDP does not have such a state, so the decision-making process is conducted forever.

The behavior of the agent is described by its policy, which depicts how the agent chooses its actions given a state. The policy could either be stochastic or deterministic. A policy is said to be stationary if it does change over time. Denoting the policy by *l,* the agent's goal is to find a policy that maximizes its expected discounted return from every state *x*:

$$\max_{l:X\to U} R_l(x) = \max_{l:X\to U} E\left\{\sum_{k=0}^{\infty} \gamma^k r_{k+1} | x_0 = x, l\right\} \tag{4-1}$$

where $\gamma \in [0,1)$ is a discount factor for the future reward, the return *R* represents the reward accumulated by the agent in the long term. There are multiple ways of defining the long-term return [25]. The discount factor γ could be seen as a way to represent an increasing uncertainty about the reward that will be received in the future. The task of the agent then is to find the optimal policy by only receiving feedback about its immediate performance. One way it can achieve this is by computing the optimal state-action value function (Q-function) as discussed in the next section.

4.2 Q-learning Algorithm

Dr. Watkins first developed Q-learning in 1989, which assigns each pair of state-action a reward and builds a decision strategy based on the state-action value table [58, 59]. There is a broad variety of single-agent RL algorithms based on Q-learning developed ever since, e.g., model-free methods based on online value function estimation [60-62], model-based method (usually called dynamic programming) [63, 64], and model-learning methods [65]. In this project, the battery SoC level is utilized as the state, and the choice of CTSF σ is modeled as actions at each state in this MDP. Every action in every state comes with an instantaneous reward *r*, and a bonus reward is given at the end of the operation once the SoC goal is reached. For every state-action pair *(s,a),* there is a Q* value that indicates its optimization level, which not only includes the instantaneous reward but partially contains the delayed reward after taking action *a*. The Q* value is the expected reward for each state-action pair. It acts as an operation instruction under the circumstance. The optimal Q-function is defined as

$$Q^*(x,u) = \max_l Q_l(x,u) \tag{4-2}$$

which is unknown initially to the agent. The optimal Q-function satisfies the Bell-man optimality equation:

$$Q^*(x,u) = \sum_{x' \in X} f(x,u,x') \left[r_{k+1} + \gamma \max_l Q^*(x',u') \right] \; \forall x \in X, u \in U \tag{4-3}$$

This equation states that the optimal value of taking u in x is the expected immediate reward plus the expected discounted optimal Q value from the next state. There are different strategies in choosing action given the Q value table, such as greedy strategy, ε −greedy policy, and Boltzmann strategy [25]. The greedy strategy $\bar{l}^*$ is to choose an action with the largest optimal Q-value at every state

$$\bar{l}^*(x) = \arg\max_u Q^*(x,u) \tag{4-4}$$

In the most extreme case, the agents are not given any information about the world or environment. The agents use temporal differences to estimate the true Q* value of each state-action pair based on their experience. The updating equation is shown in (4-5) [58].

$$\begin{aligned} Q_{k+1}(\mathrm{x}_k,u_k) = Q_k(\mathrm{x}_k,u_k) + \alpha(r_{k+1} + \gamma \max_{u'} Q_k(\mathrm{x}_{k+1},u') \\ - Q_k(\mathrm{x}_k,u_k)) \end{aligned} \tag{4-5}$$

where α is an updating step size factor, γ is the horizon factor, x_{k+1} is the possible future state that is reachable from state s. The term $r_{k+1} + \gamma \max_{u'} Q_k(\mathrm{x}_{k+1},u') - Q_k(\mathrm{x}_k,u_k)$ is called the temporal difference, indicating the difference between the current estimation $Q_k(\mathrm{x}_k,u_k)$ of optimal Q-value of (x_k,u_k) and the updated estimate $r_{k+1} + \gamma \max_{u'} Q_k(\mathrm{x}_{k+1},u')$. This new update is a sample of the right-hand side of the Bellman equation (4-3). The sequence Q_k converges to Q^* under the following conditions [66-68]:

- The value of the Q-function is stored and updated for all state-action pairs.

- All the state-action pairs are visited infinitely often.

- The following requirements (4-6)-(4-7) of learning rate is satisfied:

$$0 < \alpha_t < 1, \sum_{t=1}^{\infty} \alpha_t = \infty \tag{4-6}$$

$$\sum_{t=1}^{\infty} {\alpha_t}^2 < \infty \tag{4-7}$$

A conventional choice is to pick the step size factor as

$$\alpha_t = \frac{1}{t} \tag{4-8}$$

since

$$\sum_{t}^{\infty} \frac{1}{t} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots = \infty \tag{4-9}$$

$$\sum_{t=1}^{\infty} \frac{1}{t^2} = \frac{\pi^2}{6} \tag{4-10}$$

The second requirement could be fulfilled if we set nonzero probabilities for the agent to try all available actions in all states. The exploration process is achieved by choosing at each time step a random action with probability ε, and the present optimal action with probability $1 - \varepsilon$ (such is

referred to as ε −greedy exploration). Another common option is to use the Boltzmann exploration procedure, which selects action *u* in a state *x* with probability:

$$p(x,u) = \frac{e^{Q(x,u)/\tau}}{\sum_{\hat{u}} e^{Q(x,\hat{u})/\tau}} \tag{4-11}$$

where $\tau > 0$, denoted as the temperature, controls the randomness of the exploration. The remaining part is to design the reward function *r(s,a).* The value of reward function has a significant impact on the final strategy found by Q-learning. In the project, the designed reward functions in the single-agent and multi-agent simulations are different. For the single-agent scenario, the reward function is the same as the objective function:

$$r(SoC,\sigma) = obj(\sigma,SoC,t) = w_{com} \cdot f_{com}(\sigma,t) + w_{SoC} \cdot f_{av}(\sigma,SoC,t) \tag{4-12}$$

because the agent has all the information needed to compute the communication quality and energy availability. However, for a system with more learners without communication links, the actual CTSF choice is unknown to each agent. Thus, a simple reward function only considering the current SoC and CTSF is designed

$$r(s(t),\sigma(t),t) = w_{soc} \cdot SoC(t) \cdot + w_{com} \cdot \sigma(t) \tag{4-13}$$

where $\sigma(t)$ is the action just taken by the BS and w_{soc} and w_{com} are the weighting factors. Giving this reward function, each controller could update its Q* values. Also, an extra reward is given to the agent at t_{end}

$$r(s,a,t_{end}) = \begin{cases} 10 \;\; if \;\; SoC(t_{end}) \geq SoC_{goal} \\ 0 \;\; if \;\; SoC(t_{end}) < SoC_{goal} \end{cases} \tag{4-14}$$

which gives a bonus to the actions that enable the system reaching the desired SoC goal.

In the original design, there is only one agent in this MDP, and the environment is assumed to be static; thus the optimal Q values are fixed. These assumptions are not always valid in the microgrid discussed in this dissertation. In the simulation, two types of learning mechanisms were tested: centralized BS controller equipped with Q-learning and multiple BSs implemented single-agent Q-learning. The stability and convergence of multi-agent Q-learning without communication have not yet been proven [25]. There have been modified versions of multi-agent Q-learning such as Nash-Q, Team-Q, Minimax-Q, and distributed-Q [38, 69]. However, most of them require designated communication between agents to observe the reward or actions of the other agents, which is not guaranteed in the microgrid discussed.

4.3 Linear Reward-Inaction Algorithm

Linear reward-inaction is a learning algorithm designed for the multi-agent system by P.S. Sastry in 1994 [67]. This method is built upon the model of learning automata, which aims at reaching equilibriums of Markov games. The Markov games are an extension of game theory to MDP-like environment. In these games, the designer not only considers any single agent's reward but all the agents' learning process. Then, algorithms were proposed aiming at attracting agents to equilibriums in the Markov game. The definition of equilibrium in the Markov game is the same

with a classical multi-player game, which claims no player can obtain a better payoff by deviating from the equilibrium. These algorithms could be seen as a policy updating methods since the agents update the probabilities of choosing available actions directly. The updating sequence of an agent implementing Linear reward inaction is listed below:

1. At time *t*, the player (automaton) choose an action according to its action probability vector $\boldsymbol{q}_i$. Suppose the action taken is δ_i.
2. Each player obtains a payoff based on the set of all players' actions. The reward of player *i* is $r_i(t)$.
3. Each player updates his action probability according to the rule

$$\boldsymbol{p}_i(t+1) = \boldsymbol{p}_i(t) + b \cdot r_i(t)\left(\boldsymbol{e}_{\delta_i} - \boldsymbol{p}_i(t)\right), i = 1, \dots, N \tag{4-15}$$

where $0<b<1$ is a learning rate parameter and $\boldsymbol{e}_{\delta_i}$ is a unit vector with its δ_ith component unity. This updating law can be represented as

$$P(k+1) = P(k) + bG(P(k), a(k), r(k)] \tag{4-16}$$

where $a(k)$ denotes the action chosen by the agent at step/time k and $r(k)$ are the resulting rewards, and G(.) represents the updating law specified by eqn. (4-15). P(k) converges weakly to a solution of an ordinary differential equation

$$\frac{dP}{dt} = E[G(P(k), a(k), r(k)|P(k) = k] \tag{4-17}$$

whose solutions are the pure Nash equilibriums in the original Markov game. The detailed proof could be found in [67] Theorem 3.4. So, this learning algorithm convergences to a pure Nash Equilibrium of the multi-agent Markov decision process (a.k.a Markov game). Other than that, whether an equilibrium of a mixed strategy could be obtained or if the learning process is trapped in a limit cycle is not guaranteed.

SoC/CTSF	CTSF_1	CTSF_2	…	CTSF_n
SoC_1	p11	P12	…	P1n
SoC_2	p21	P22	…	P2n
…	…	…	…	…
SoC_m	pm1	pm2	…	pmn

Figure 18: Learning space of a Linea reward-inaction agent

When applied to the microgrid energy management, the BS controller update a list of CTSF probability vectors given the system SoC level. The battery SoC is divided into multiple levels, and the agent follows different CTSF strategies at each level as shown in Figure 18. The summation of elements in every row in this table is one. The reward function could be the objective function eqn. (2-12) if the communication network in the microgrid is functioning. Otherwise, if the BS is not capable of sharing CTSF and SoC status with the others, a local-information-based reward function is available for the agent to use, as shown in eqn (4-18).

$$r_i = \begin{cases} 1 & t > t_d, SoC_i(t) > SoC_{goal} \\ w_{load}\dfrac{1}{1+\mathrm{e}^{-\alpha_i SoC_i(\mathrm{t})}} + w_{SoC}\dfrac{1}{1+\mathrm{e}^{-\frac{kSoC_i(\mathrm{t})}{\alpha_i}}} & t < t_d, SoC_i(\mathrm{t}) \geq SoC_{min} \\ (1-\alpha_i)/10 & t < t_d, SoC_i(t) \leq SoC_{min} \\ 0 & (t < t_d, SoC_i(t) < SoC_{low}) \; or \; (t > t_d, SoC_i(t) < SoC_{goal}) \end{cases} \tag{4-18}$$

where $\sigma_i(t)$ is the CTSF chosen by the *i*th agent, the two Logistic functions ensure the reward stays in the range of [0,1], $\alpha_i SoC_i(\mathrm{t})$ is the local load satisfaction rate weighted by the battery SoC level, and $\frac{kSoC_i(\mathrm{t})}{\alpha_i}$ approximates the remaining operating time assuming no power supply is provided. The factor *k* accounts for the battery energy/load energy ratio, which could be different in each microgrid. As will be shown in the simulation section, the approximated reward function makes the training process longer (mostly due to its requirement of lower training rate) but guarantees that the system achieves the same performance as the actual objective function does.

Two modifications were made considering the specific application. First, the probabilities of actions are limited to

$$\boldsymbol{p}_i(t)\epsilon[\frac{1-top}{n-1}, top] \tag{4-19}$$

where *top* is a value close to 1 (e.g., 0.8), *n* is the number of available actions. This limit on the probability of action was made to prevent the agents from actually reaching the pure NE. Because at the pure NE, one of the actions is dominating the other actions

$$\boldsymbol{p}_{dom}{}^*(t) = 1, \boldsymbol{p}_i{}^*(t) = 0 \; \forall i \neq dominator \tag{4-20}$$

meaning the exploration of the agent is finished, and the agent will never choose other CTSF actions in the future. This feature might leads the agent to local optimums and leaves no room for updating when environmental changes. By applying probability limiting (4-19), the agent has a chance to escape from the local optimums.

Another modification is made a low-SoC barrier. This rule is set for each agent as follows:

Low-SoC-check (p(SoC_i, CTSF))

1: *Scan the whole learning table*
2: *if p(SoC_l, CTSF_min)>=$P_{barrier}$*
3: *p(SoC_k, CTSF_min)= p(SoC_l, CTSF_min)* $\forall k \leq l$
4: *repeat*

where $P_{barrier}$ is a probability threshold close to one, and *p(SoC, CTSF))* represents the probability of choosing CTSF at SoC_i. The principle behind this setting is simple: if the optimal action at SoC_l is to maintain at the minimum CTSF, then SoC_l could be classified as 'critically low' from the agent's experience. Thus, the agent needs not to explore any SoC states lower than this value to decide what the optimal strategies are because those states could only mean worse

energy conditions. Therefore, the optimal action at any lower SoC levels could only be maintaining the lowest load consumption as well. The agents applying this modification have responses better when encountering a power loss situation where the original learning setting requires the agents to explore the lower SoC region. The other benefit of this mechanism is that it prevents the agent from converging to a possible equilibrium leading to critical low system SoCs.

5.0 Load-ratio Learning Game

5.1 Inspiration from Virtual Two-Player Game

This approach is enlightened by the virtual-two player game. A microgrid implemented with the virtual two-player game has a performance as the number of players increases. This is because of the BS controller's biased understanding of its and the virtual BS's *loads*. Therefore, a new parameter *load ratio* r_t is introduced:

$$r_{t_i} = \frac{{P_B}^i + {P_T}^i}{\sum_{i=1}^{N} ({P_B}^i + {P_T}^i)} \tag{5-1}$$

which represents how much a BS's load demand takes in the total load demand of the microgrid. This value was assumed to be known to the BS controllers in the previous two-player game and exhaustive search approaches. In the virtual two-player game, however, as mentioned before, knowing the real value of this load ratio does not guarantee a strategy that maximizes the BS's payoff. Because in the two-player virtual game, a BS controller estimates the virtual player's moves assuming its load ratio is 1-r_{t_i}. However, the actual join-action of the other BS controllers can only be obtained if an actual n-player game is solved. But this joint-action action could be the solution of another two-player game with different load-ratios. Moreover, it is possible for the BS controller to locate that load-ratio through a learning process.

5.2 Load-Ratio Updating Process

Therefore, instead of precise knowledge of the load ratio, a vector of possible load ratios and probability list of the load ratio are given to the BSs:

$$l = \left[{r_{t_i}}^1, {r_{t_i}}^2, \dots, {r_{t_i}}^M\right] \tag{5-2}$$

$$l_r = [\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_M], \sum \boldsymbol{p_i} = 1 \tag{5-3}$$

where ${r_{t_i}}^k$ are the *i*th BS's possible load ratios, M is the number of possible load ratios, and $\boldsymbol{p_k}$ indicates the confidence of the BS controller that r_{t_i} is equal to ${r_{t_i}}^k$. l_r is called the load-ratio policy of the BS controller. When choosing the CTSF, the agent picks its load ratio based on the load-ratio policy

$$r_{t_i} = {r_{t_i}}^k \; with \; probability \; \boldsymbol{p_i} \tag{5-4}$$

Then, the player conducts a regular virtual two-player game and obtain a mixed strategy as described in section 3.7. Whether the CTSF is obtained through a common-interest game or zero-sum game depends on the players' available information and the integrity of the microgrid. When the microgrid is operating normal, the BSs in the system share the same objective, thus, the immediate CTSF could be solved using the common-interest setting. However, when the microgrid is affected by a natural disaster, its components could be damaged or malfunctioning. In this condition, the behavior modes of BSs could be different. Some BSs might experience a surge in load demand, which is not desired to be shed, but other BSs may reduce their load consumption in

order to conserve more stored energy. In this case, the BSs in this microgrid do not necessarily share a common goal. If the communication links between BSs are functioning, BSs can broadcast their load conditions so that the other BSs can adjust accordingly. Otherwise, BSs can only estimate the objective function assuming the worst scenario, in which the virtual BS in the two-player game. With this assumption, the objective of the BSs becomes a zero-sum game. Thus, it should be solved, as indicated in section 3.3. Depending on the available information, the agent computes a reward function based on the objective function eqn. (2-12) (normal condition) or the local reward function eqn (4-18). Then, the load-ratio policy is updated using the Linear reward-inaction algorithm. The architecture of this learning-gaming algorithm is shown in Figure 19, and the overall flow chart is shown in Figure 20.

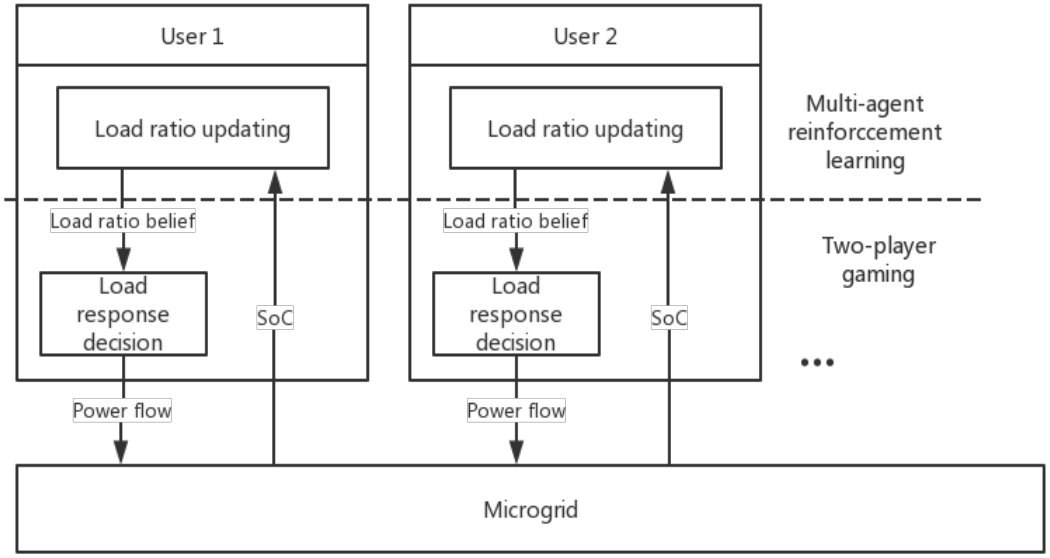

Figure 19: Learning-gaming algorithm scheme

This approach divides the original load planning process into two parts: load-ratio Markov game and CTSF two-player game. In the Markov game, using the RL algorithm, the agents explore the load-ratio space and search for equilibriums that give them the maximal reward in the two-player game. The lower level is a two-player game acting as an actuator. Compare to the original

game approach, this algorithm could be easily extended to an arbitrary number of players without any burden on the computation cost and is highly flexible to the environment changes. Additionally, because the search space for load-ratio learning is smaller than the SoC-CTSF space, the converging speed of the load-ratio learning process is higher. All these features will be shown in the simulation section.

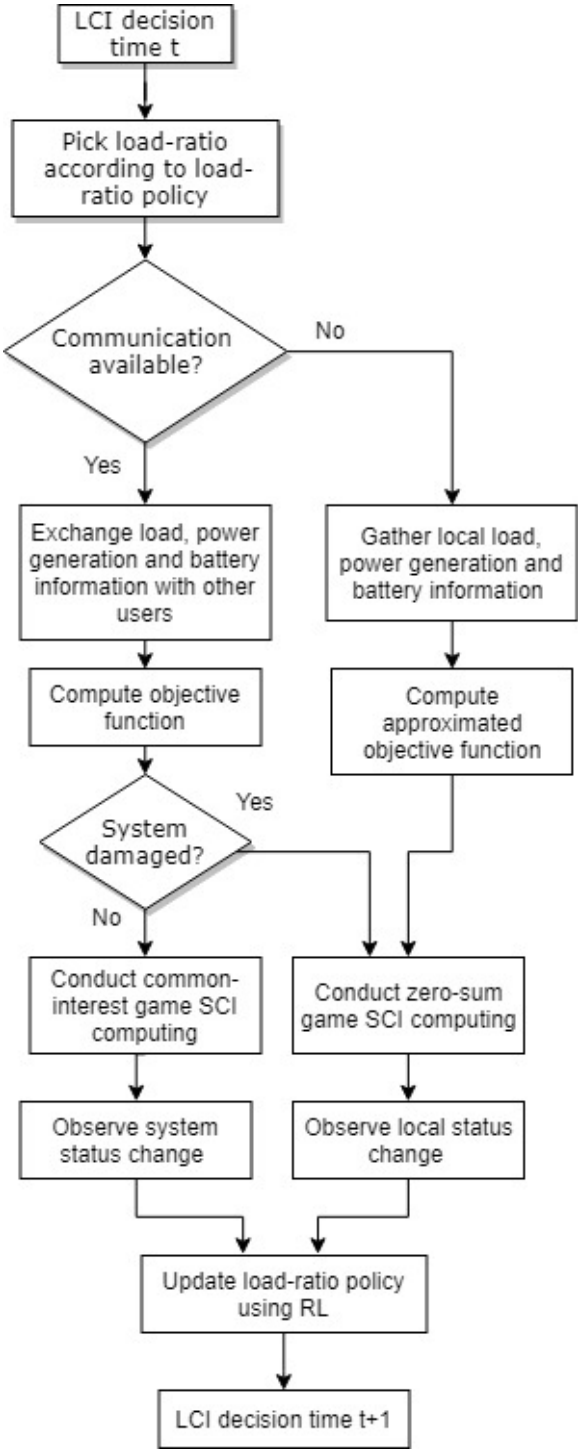

Figure 20: Load-ratio learning game algorithm flowchart

6.0 Numerical Results

In this section, the numerical results of simulated communication microgrid implemented with the two-player game, reinforcement learning, and load-ratio learning game are demonstrated and compared. The primary BS parameters are shown in TABLE II. The BS load and renewable power functions are

$$P_{BS}=\max\left(500\frac{0.000839t^3+1.205t^2-12.02t+34.29}{t^2-6.495t+43.45}+500,0\right) \tag{6-1}$$

$$P_{solar}=\max\left(\frac{-8185t^2+1108750t-707800}{25t^2-3590t+5928}+80,0\right) \tag{6-2}$$

The PV power and BS load curves vs. time are shown in Figure 21 and Figure 22. We choose the sum of objective function over 24 hours, as shown in equation (6-3), as a metric of overall system performance, because it directly relates to the average performance over 24 hours if such sum is divided by 24.

$$\mathrm{Sum}(obj)=\sum_{t=1}^{24} obj\big(t,\sigma(t)\big) \tag{6-3}$$

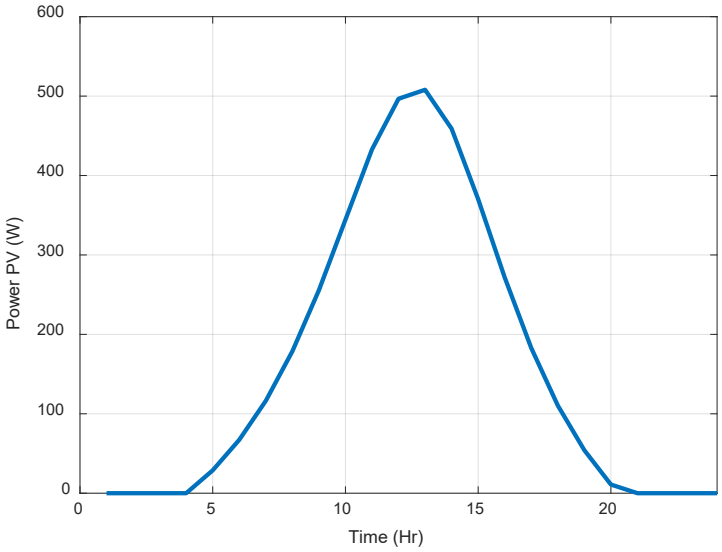

Figure 21: PV power curve

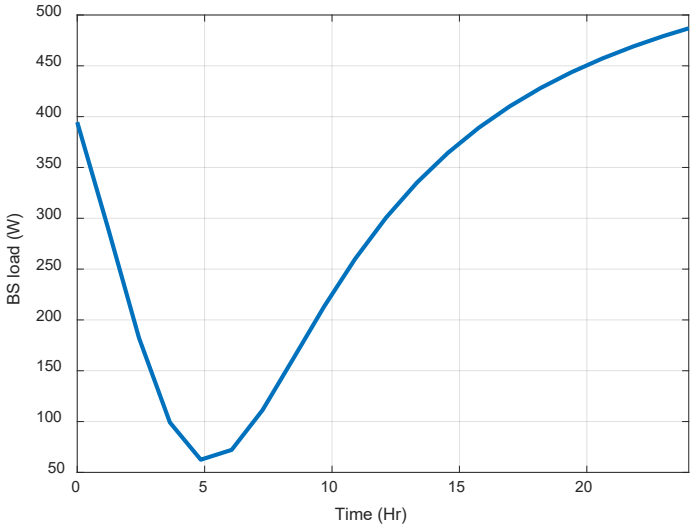

Figure 22: BS load curve

Table 9 : Simulated BS parameters

Symbol	PARAMETER	Value
w_{com}	Communication quality weight	0.5
w_{SoC}	Energy availability weight	0.5
$Ebatt$	Battery fully charged energy	24 kWh
$\overline{E_{P_{solar}}}$	Solar power generation expectation	1 kW
$\overline{E_{P_B}}$	BS base load expectation	200 W
$\overline{E_{P_T}}$	BS traffic depended load expectation	800 W
$\overline{V_{P_{solar}}}$	Solar power generation variance	4000
$\overline{V_{P_T}}$	BS traffic depended load variance	4000
SoC_{go}	Desired battery SoC level	0.8
SoC_0	Initial Battery SoC level	0.7
BW	BS total bandwidth	10MHz
a	PSNR-rate bit curve parameter	10.4
b	PSNR-rate bit curve parameter	-23.8
r	Nominal transmit rate bit	2 Mbps

6.1 Multi-Player Game

The operation of microgrid consists of identical BSs with the same load curves is simulated with no communication link between the BSs. The performance of this microgrid implemented with two-player games and its comparisons with the exhaustive search are demonstrated.

6.1.1 Two-Player Zero-Sum Game.

First, the microgrid implemented with the two-player zero-sum game is simulated. The PSNR and battery SoCs of the system applying exhaustive search and zero-sum game are shown in Figure 23 and Figure 24. As the results show, the system applied with a two-player game method has a performance that is close to that of the globally exhaustive search. Both the system battery SoCs reach the desired goal (80%), and similar trends are seen in both PSNR curves. According to [30], a moderately good target for the quality of the video stream is 37 dB PSNR, whereas a 32

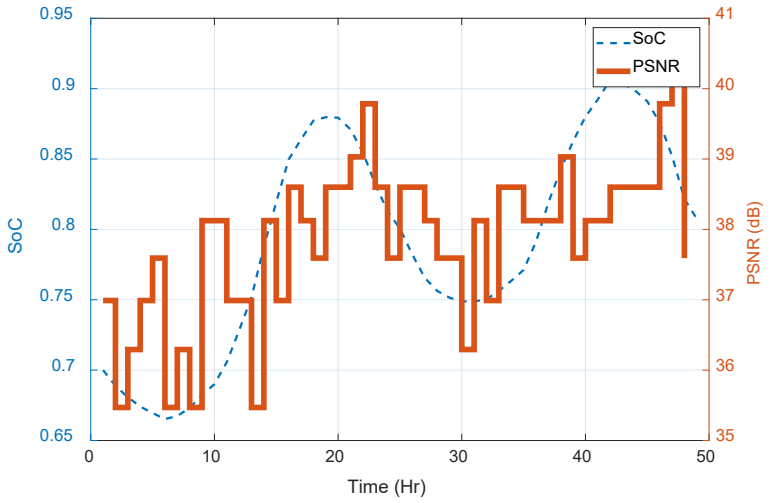

Figure 23: CTSF and SoC of the simulated microgrid applying exhaustive search, sum(obj)= 17.2512

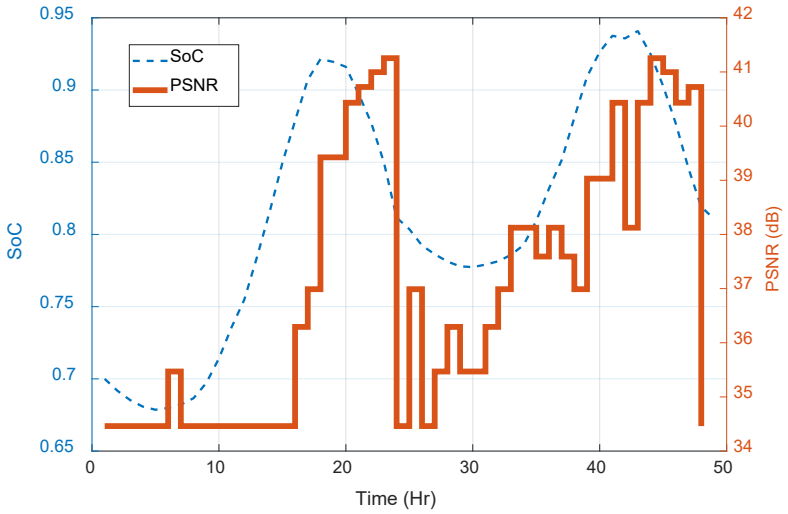

Figure 24: PSNR and SoC of the simulated microgrid, two-player zero-sum game, sum(obj)= 16.5616

dB PSNR is considered as acceptable. In term of the objective function, the sum(obj) of the mixed two-player game method is 16.5616 compared to 17.2512 that of the exhaustive search. In order to ensure that this is a consistent result, a Monte Carlo test comparing the performance of the mixed

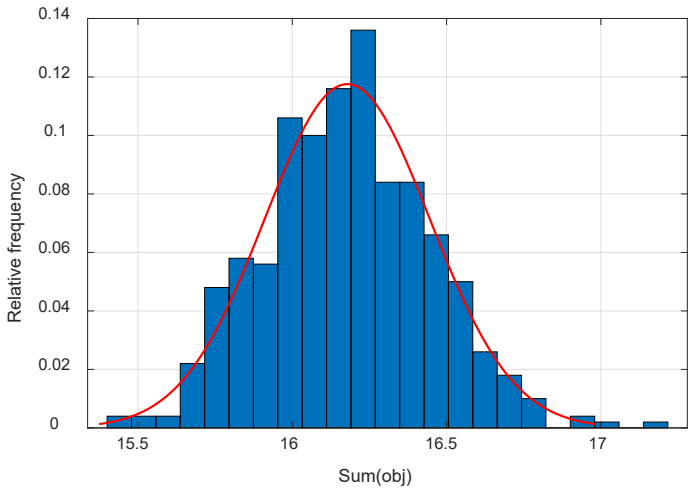

Figure 25: Distribution of Sum(U) obtained by two-player zero-sum game

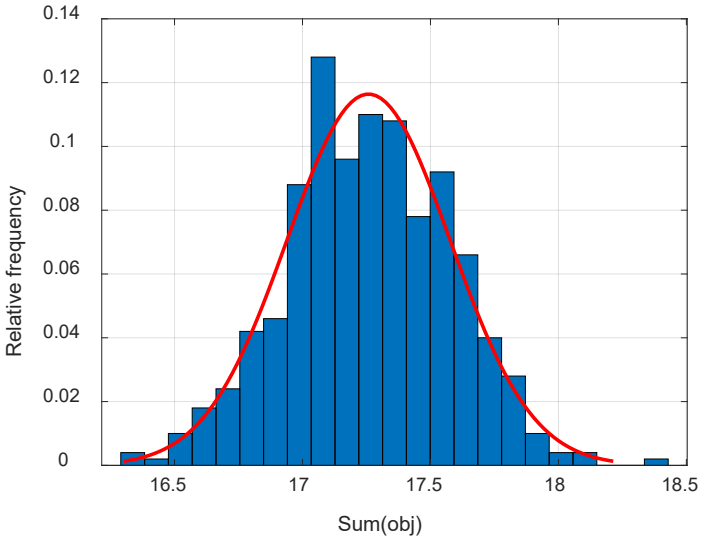

Figure 26: Distribution of Sum(U) obtained by exhaustive search

game method and the exhaustive search was conducted. In this test, 48 hours operation of the microgrid with two BSs was simulated 500 times implemented with the two algorithms considering the same initial battery SoCs and power/load distribution functions. The resulting system performance Sum(U) has an average value of 17.2666 for exhaustive search and 16.1591 for the two-player mixed game algorithm. As the results show, in terms of Sum(U), the optimality of the solution obtained by the mixed game algorithm is sufficiently close to that obtained by exhaustive search but with a much less implementation complexity, as needed in practical applications. The distributions of Sum(U) obtained by the two algorithms could be seen in Figure 26 and Figure 25. To further investigate the performance of the game approach, we performed a series of simulations with different settings of initial battery SoCs and weighting factors in the objective function. The Sum(U) function with different pairs of settings is plotted in one figure. By doing so, an overall performance trend could be observed. The simulation result is shown in Figure 27. As the result shows, under most conditions, the zero-sum two-player game method has

a performance that is close to the globally exhaustive search. A figure demonstrating the difference between the two methods under different system settings is shown in Figure 28. As shown in this result, in scenarios with low initial SoCs and small SoC weighting factors, the differences between the two-player game and globally exhaustive search are within 20% of the global optimum, and that gap is close to 0 as the initial SoC goes higher.

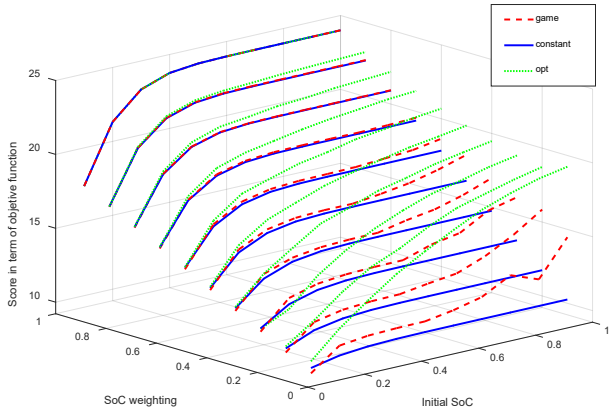

Figure 27: Sum(obj) of two-player zero-sum game with different initial SoC and weighting factor

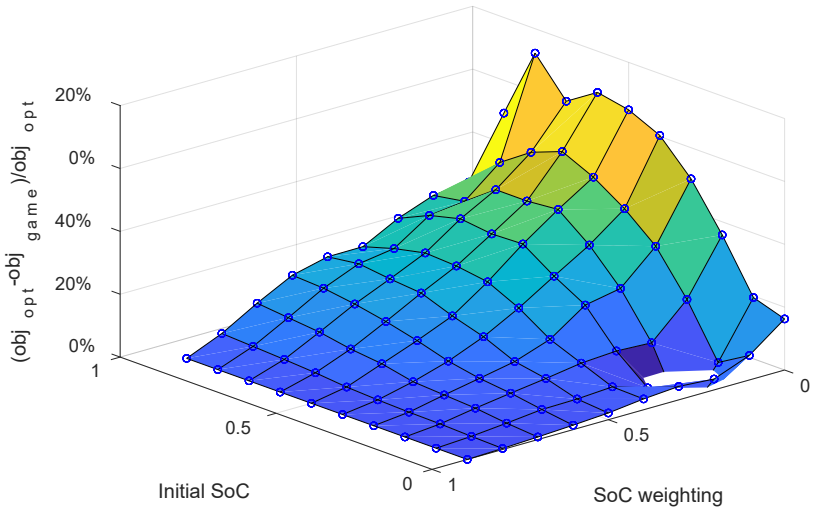

Figure 28: Difference between exhaustive search and zero-sum game solutions in percentage

6.1.2 Two-player Common-Interest Game

In this simulation, a microgrid operation with two identical BSs implemented with a common-interest game setting is simulated. The only difference is now the solver applied is quadratic programming solver. The SoC and PSNR results of the microgrid applying this method are shown and compared to the zero-sum result in Figure 29. As the result shows, the PSNRs chosen by the BS controller are higher than the ones obtained using zero-sum setting most of the time. The SoC still achieved the desired goal in the end. In terms of the objective function, the common-interest game has a score of 17.0653, which is higher than the 16.5616 obtained with the zero-sum setting. An overall comparison of the common-interest game with the exhaustive search was also made with varied initial SoC and its weighting, whose result is shown in Figure 30 and the difference percentage is shown in Figure 31. It could be seen from these results that the common-interest setting has a better performance than the zero-sum setting in most scenarios.

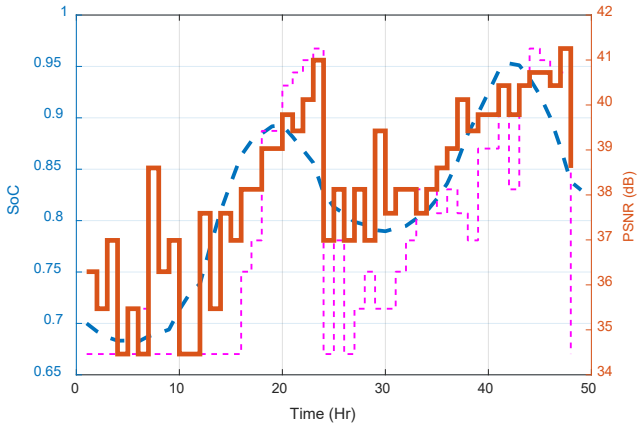

Figure 29: SoC and CTSF of simulated microgrid, two-player common-interest game, sum(obj)= 17.0807

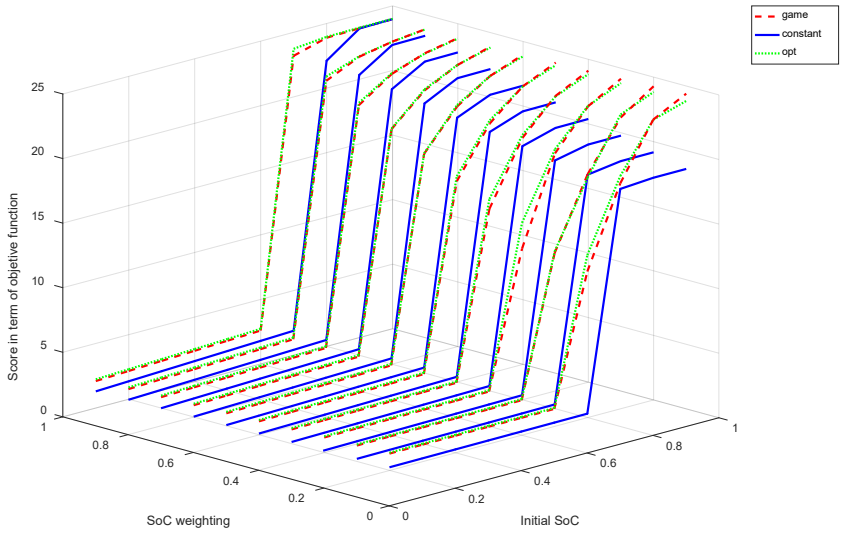

Figure 30: Sum(obj) of two-player common-interest game with different initial SoC and weighting factor

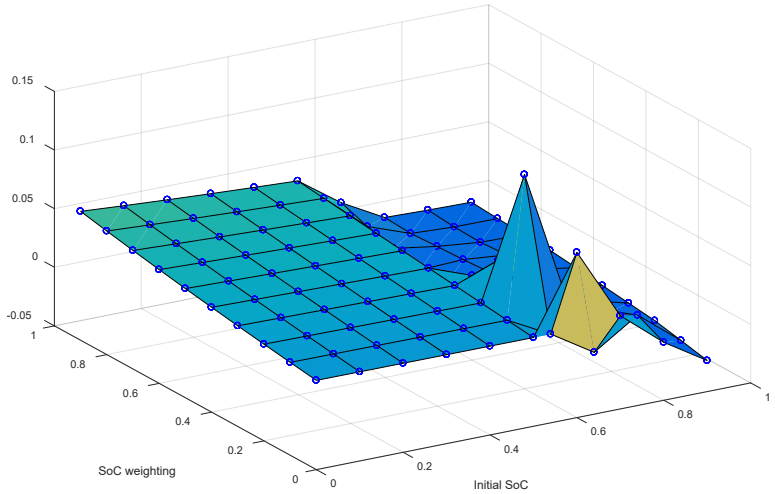

Figure 31: Difference between exhaustive search and common-interest game solutions in percentage

6.1.3 Communication Base Station Load Identifier

A simulation with estimation deviation in PV power generation is performed to test the BS load controller. The microgrid is the same with two identical BSs and load curves. But in this simulation, the PV panel is set to lose half of its power generation from t=5 to t=15 hr. The BSs controllers in the system apply the two-player zero-sum game to solve for their CTSF strategies. The resulting PSNR and SoC of the microgrid with and without a load identifier are shown in Figure 32 and Figure 33. As the figures show, the microgrid equipped with a load identifier has a performance (Sum(U)=8.1629) better than the one without a load identifier (Sum(U)=5.9233). The battery SoC level with load identifier is also higher because the BS controllers choose lower CTSFs with a more accurate power consumption model. The power consumption model of a BS with and without the load identifier is shown in Figure 34 and Figure 35, showing that the BS controller with a load identifier is capable of tracking the actual power consumption. Therefore, with a more accurate load model, the controller could detect the insufficient power generation and turn down its communication CTSF earlier.

Because of the effects of deviations between estimation and actual renewable sources generated power and communications traffic load can be considered as an additional load (when $\theta > 0$) or a power source (when $\theta < 0$), the proposed reference model can represent system structure changes and offers a better estimation to improve BS controller performance. Accordingly, the virtual load can also represent newly connected or disconnected devices (i.e., other base stations within the same dc microgrid or power sources), which makes the system more flexible.

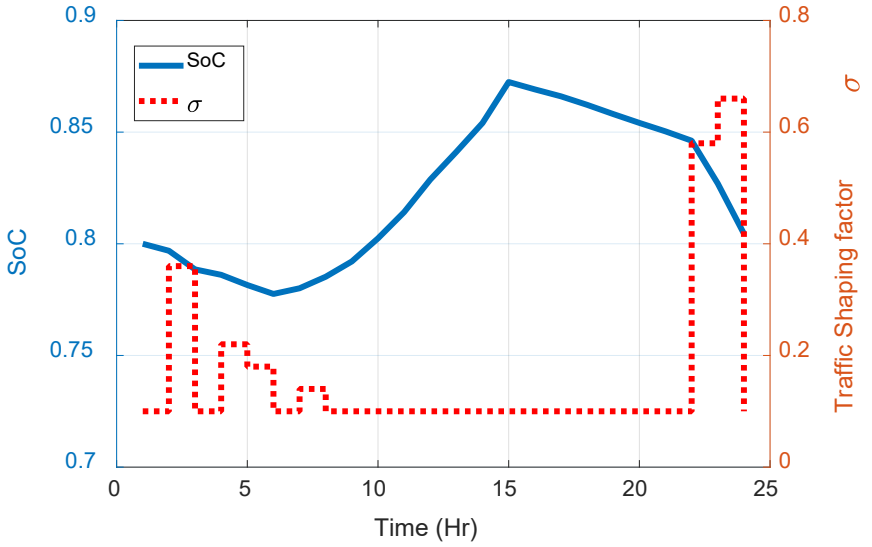

Figure 32: Simulated microgrid with the load identier. Sum(U)= 8.1629.

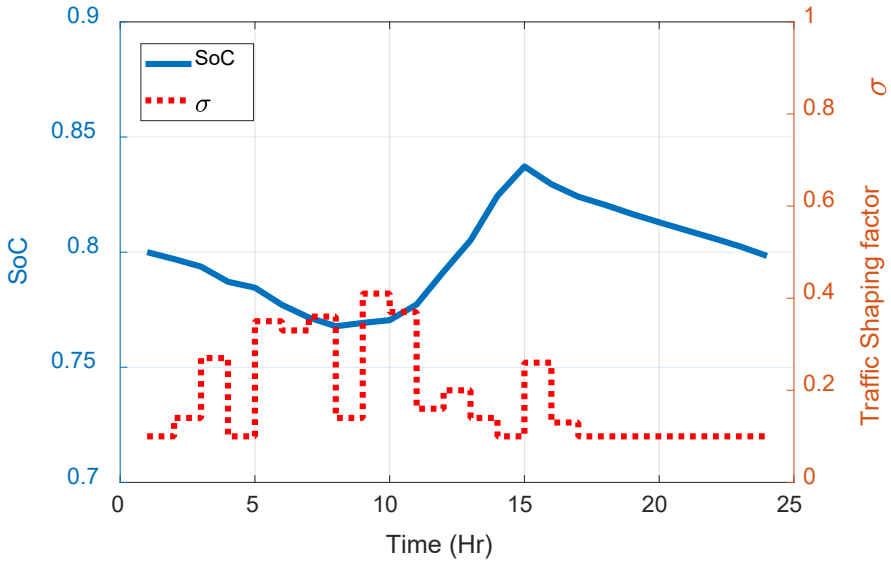

Figure 33: Simulated microgrid without load identifier. Sum(U)= 5.9233

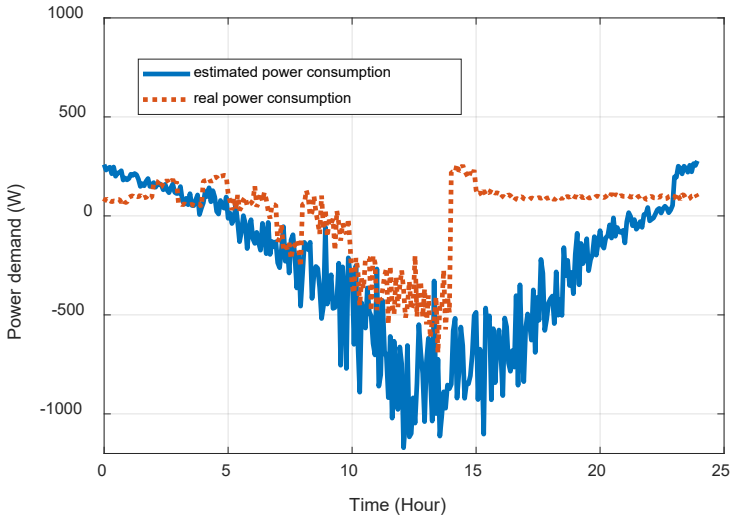

Figure 34: Power consumption estimation of a BS without load identifier

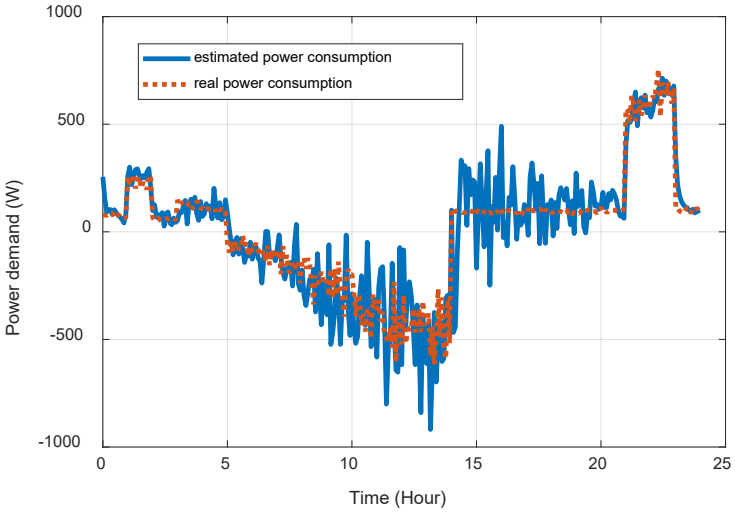

Figure 35: Power consumption estimation of a BS with load identifier

6.1.4 Virtual Two-Player Game

In this section, the virtual two-player game performance drop along with the increase of players is demonstrated with different choices of system configuration. The configurations parameters manipulated in this simulation are the energy availability weighting factor and the game modeling form (zero-sum or common-interest).

In the first case, a microgrid applied with a two-player game is tested with different w_soc and the number of players. The system performance with the zero-sum setting is shown in Figure 36. As this result suggests, the performance drop has a limit, which is positively related to the energy availability weighting factor. The drop limit is caused by the maximin feature of the zero-sum setting. Also, if the energy availability weighting is larger, the system performance has a more significant decreasing as the number of BSs increases.

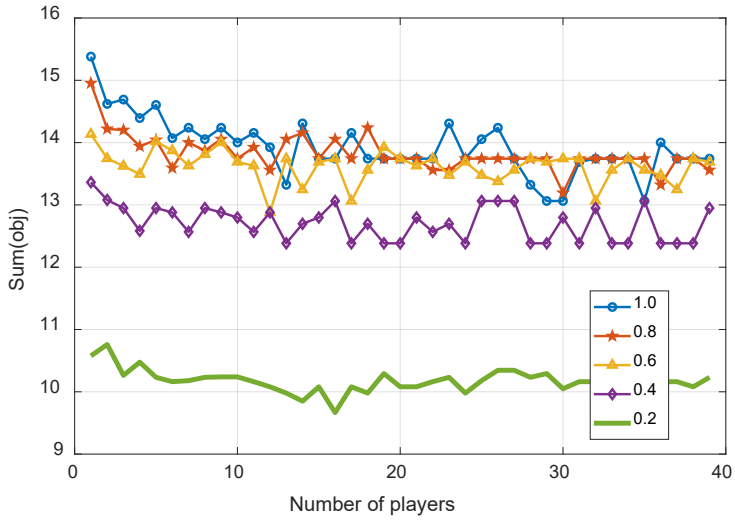

Figure 36: System performance with different w_soc, zero-sum game

In the second case, the microgrid energy management is solved using the common-interest game setting, and the system performance with the different number of players is shown inFigure 37. In this case, the performance drop of the system does not have any minimum level. This performance drop is because the system could not guarantee the minimal probability P_{min} of reaching the SoC goal. Generally, the smaller the energy availability weighting is, the larger the performance drop is.

The system SoC and PSNR with twenty BSs in the microgrid implemented with the zero-sum and common-interest two-player game are shown in Figure 38 and Figure 39. As the simulation results show, the zero-sum game sill ensures the system reaching the SoC goal, but the common-interest game suggests to have a high PSNR strategy and fails to reach the SoC goal. The overfitted behavior discussed before the behavior of PSNR is also observed. Both of the system PSNRs are either at their maximum or minimum most of the time.

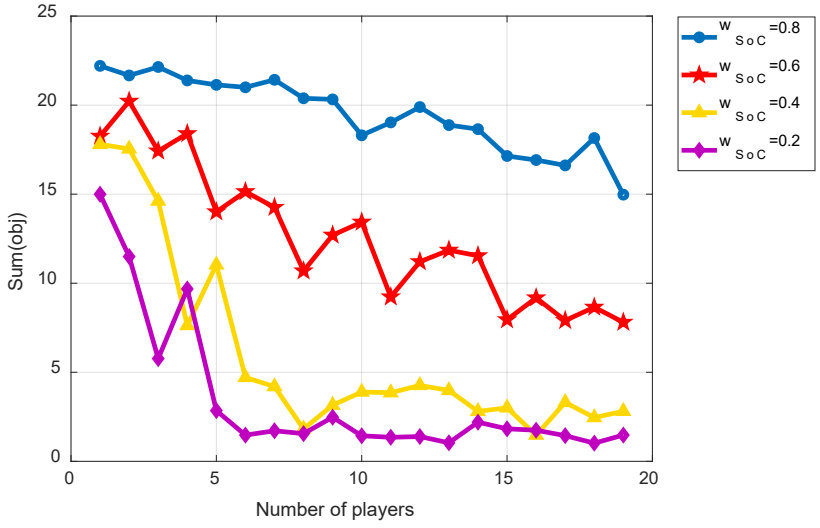

Figure 37: System performance with different w_soc, common-interest game

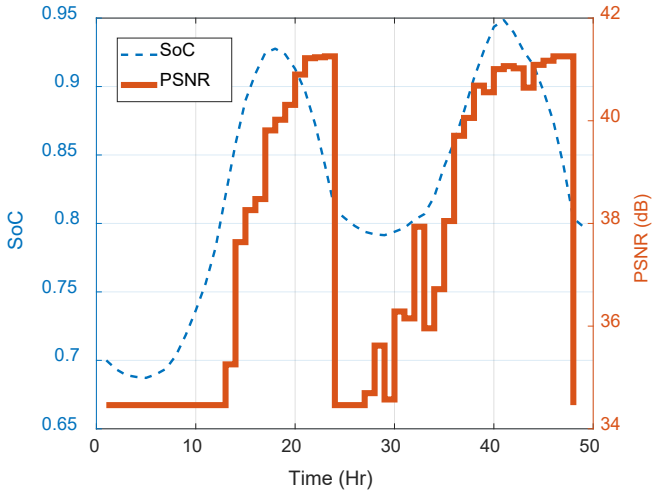

Figure 38: System SoC and PSNR implemented zero-sum game, 20 BSs

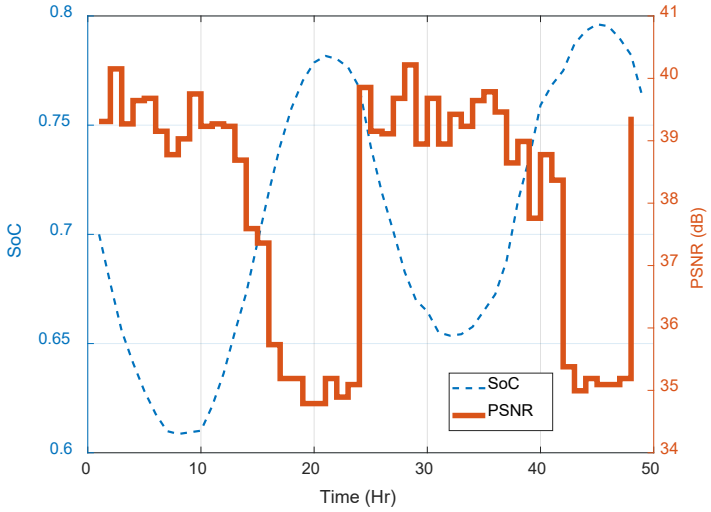

Figure 39: System SoC and PSNR implemented common-interest game, 20 BSs

6.2 Q-learning Algorithm

6.2.1 Single-Agent Q-Learning

In this simulation, the microgrid is controlled by a single agent applying Q-learning. The initial Q value for all state and actions are set to be 0.05, updating step is $\alpha_t = \frac{1}{t}$ and the horizon factor is $\gamma = 0.7$. The SoC weighting factor is 0.6. Remark here the time t is counted independently for each SoC level. The available communication TSFs for the controller are [0.05, 0.10, … 0.95, 1]. The final CTSF strategy obtained by the agent and battery SoC after 100 days of training are shown inFigure 40. It can be seen that the agent's load planning strategy is highly dependent on the battery SoC. The Q-value table and the learning curve showing the cumulated reward function could be seen in Figure 41and Figure 42, which show that the main learning was happening during the first five days. Also, the system performance measured in the objective function is shown in Figure 43, indicating that the trend of objective function fits that of the reward function.

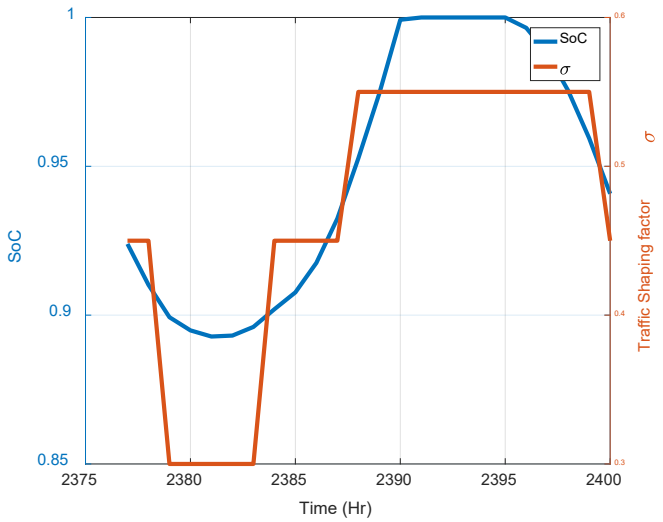

Figure 40: CTSF and SoC obtained by single agent Q-learning

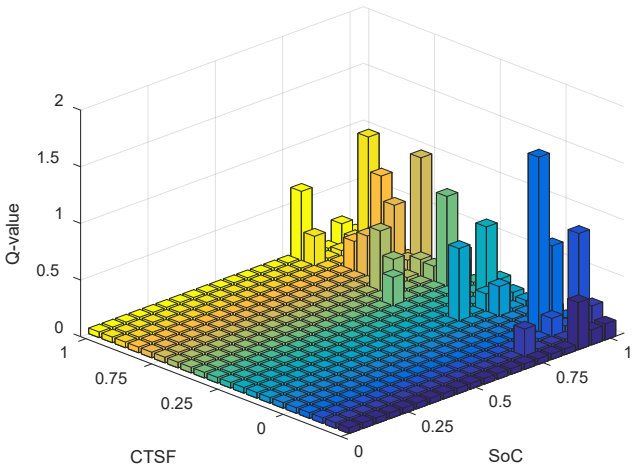

Figure 41:Q-value chart after 100 days after training

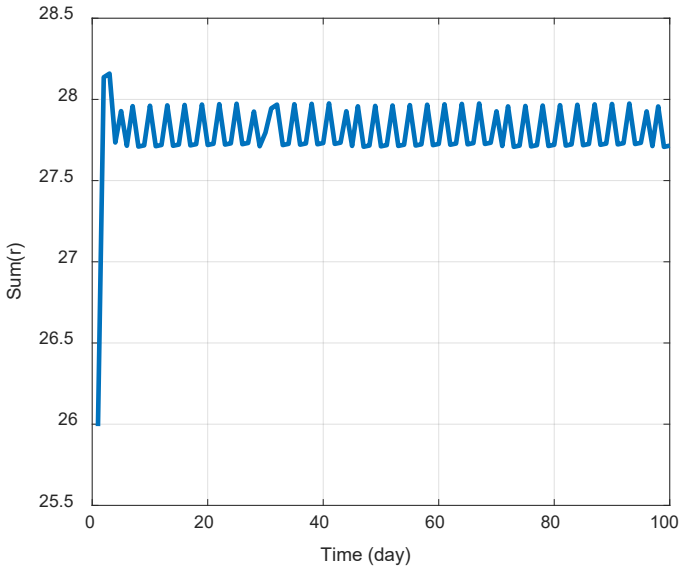

Figure 42: Learning curve of the agent

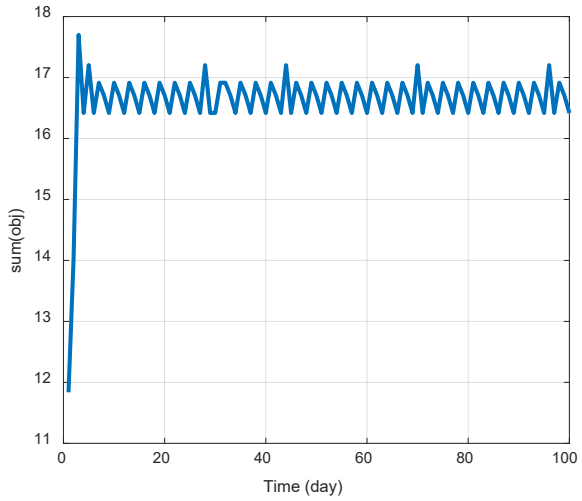

Figure 43: Objective function of the system during the learning process

6.2.2 Multi-Agent Q-Learning

In this scenario, five BSs are equipped with the Q-learning algorithms with the same setting applied in the single-agent scenario. The loads in each BS are equal, and the variances of power generation and loads are set to zeros. The multi-agent Q-learning algorithm worked fine in some cases as shown in Figure 44 and Figure 45, where the system SoC is stabilized around 90% while the CTSF is around 0.5. In another simulation, the agents in the system obtained a different strategy and the system SoC is kept around 85% as shown in Figure 46. Also, the final reward of the agent is lower as shown in Figure 47. However, in some cases, the agents fail to maintain the battery SoC above the desired level as shown in Figure 48 and Figure 49. As the results revealed, the behavior of the system obtained by the multiagent Q learning is not always consistent or stable hence needs further improvement.

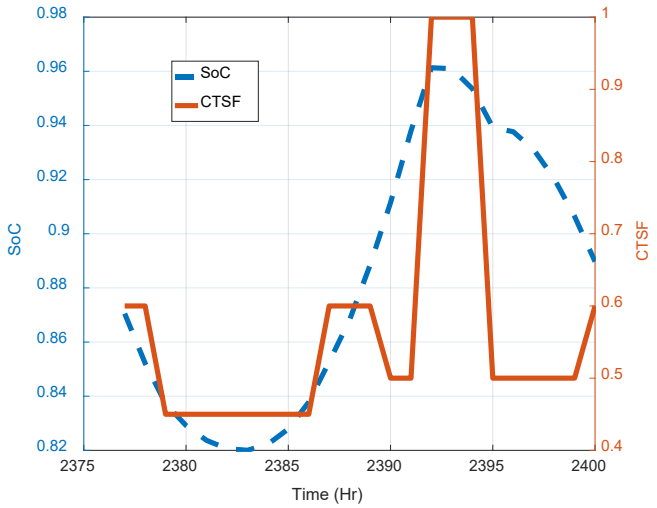

Figure 44: SoC and CTSF obtained by multi-agent Q-learning case 1

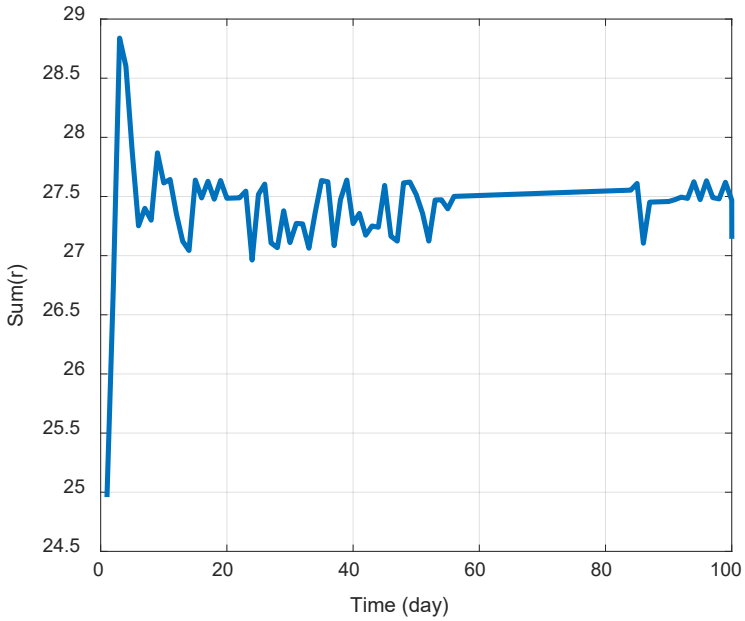

Figure 45: Learning curve of one agent, case 1

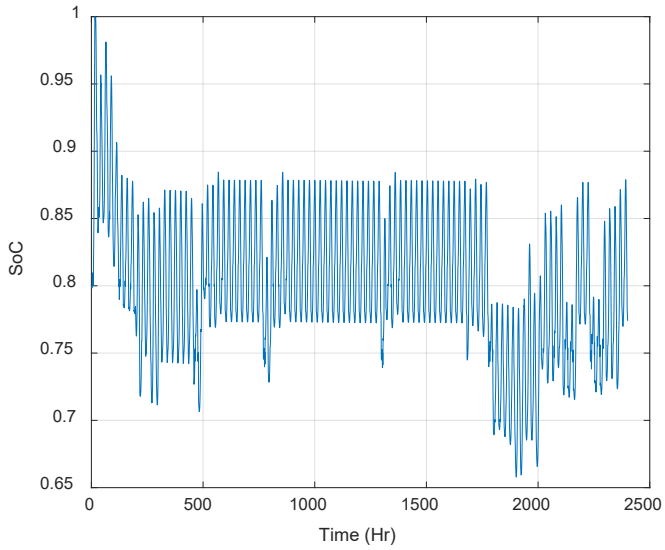

Figure 46: System SoC during the learning process, case 2

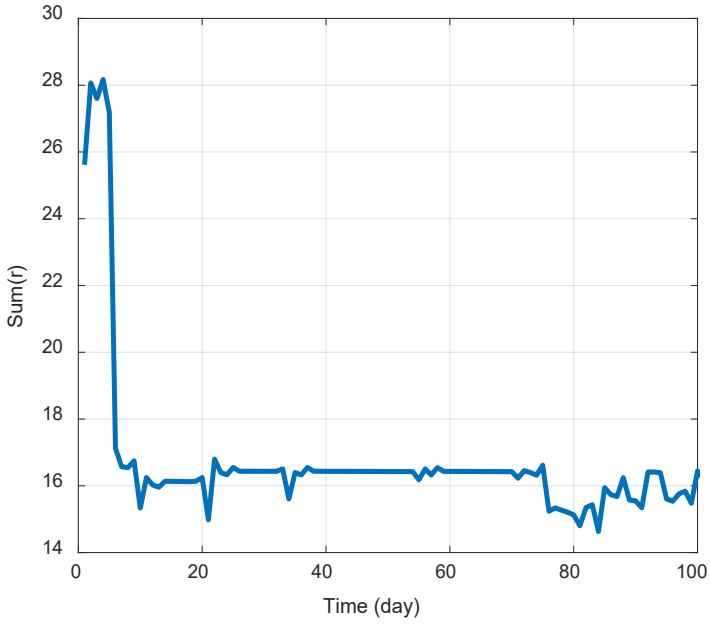

Figure 47: Learning curve of an agent case 2

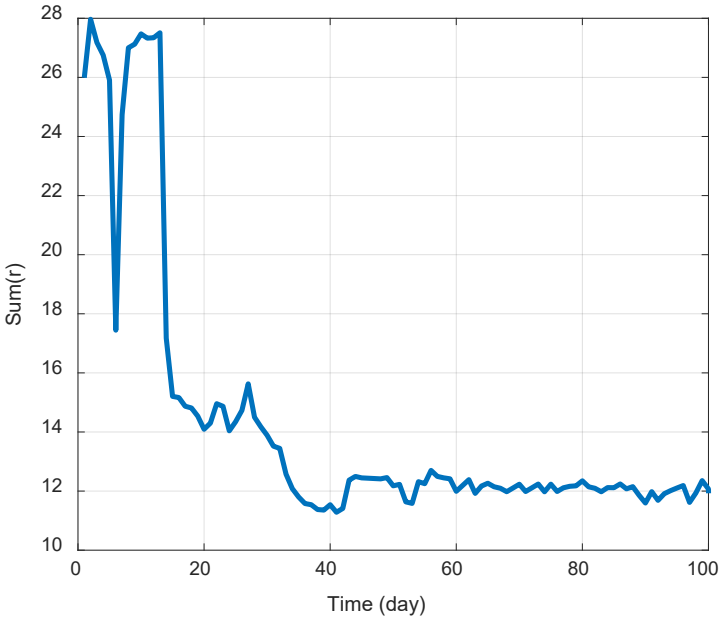

Figure 48: Learning curve of an agent, case 3

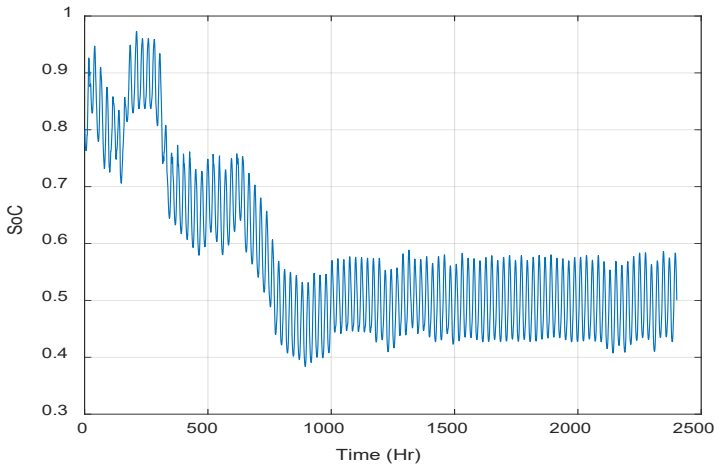

Figure 49: System SoC during the learning process, case 3

6.3 Linear Reward-Inaction

6.3.1 Normal Operation

In this simulation, the performance of the system applying Linear-reward inaction is conducted in a normal environment where BSs controllers have full access to power/load information, and there are no unexpected changes in the PV power generation or load curves. There are 20 BSs in the simulated microgrid, and the parameters are the same as the one shown in Table I. The battery SoC is divided into 20 levels with the step of 0.05, while each BS has an available CTSF range of 0.2-1.0 with the step of 0.05 thus the search space for each BS is a 20×16 matrix. The resulting battery SoC, system CTSF and system performance are shown in Figure 50 and Figure 51. Also, the trained strategy space is shown in Figure 52. In this simulation, the learning rate is 0.1, and the agent spent around ten days before the system performance is stabilized. As Figure 51 shows, the performance of this system was improved during the trial-and-error process and reached a steady level in the end.

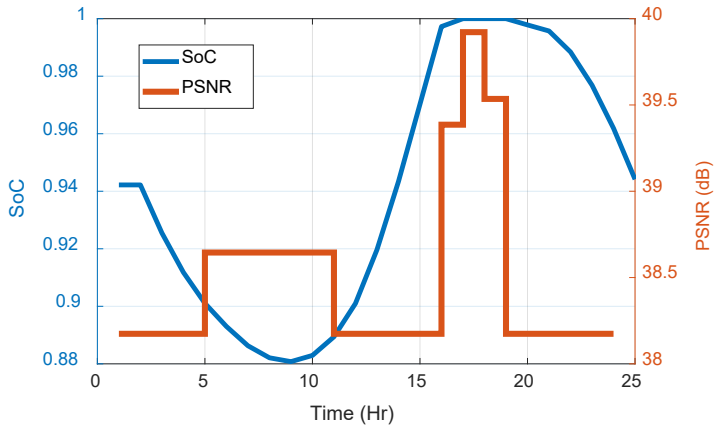

Figure 50: System SoC and CTSF obtained by Linear-reward inaction

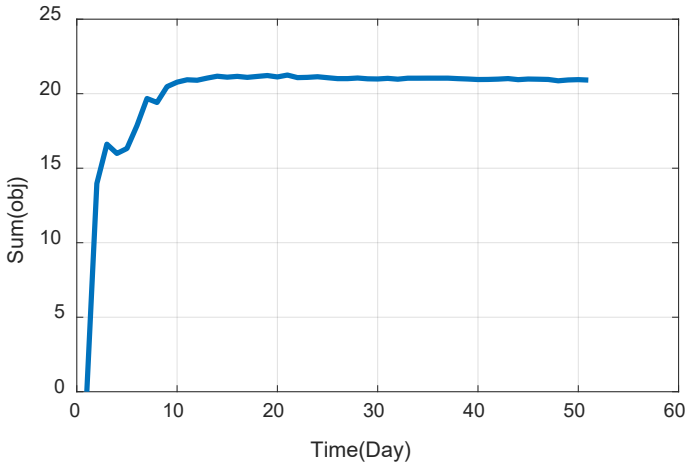

Figure 51: System performance with learning rate b=0.1

One way to increase the converging speed is to have a larger learning rate. For example, if the learning rate is set to 0.2, the learning only takes five days, as shown in Figure 53. However, this modification has two potential drawbacks: firstly, the converging of the Linear-reward inaction requires a sufficient small learning step, so a larger learning rate might cause instability of the learning process; secondly, when the environment is changing, large learning rate might cause the agents to be overfitted and trapped in sub-optimal solutions. The last phenomenon could be seen from a simulation with a large learning rate as shown in Figure 54 and Figure 55. As seen from these results, the BSs/agents converged to an equilibrium where reward is too low. Also, the BS controllers failed to maintain the system energy level. In fact, the batteries in this system are fully discharged. Therefore, the choice of learning rate for the RL algorithm plays an essential rule in the learning process and needs to be chosen carefully.

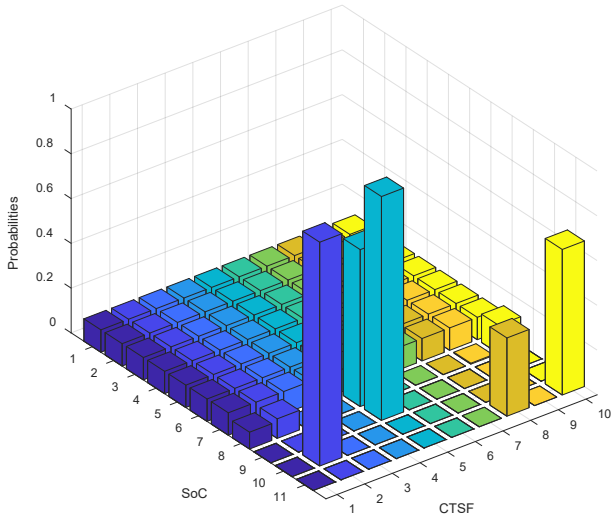

Figure 52: Obtained CTSF strategy space

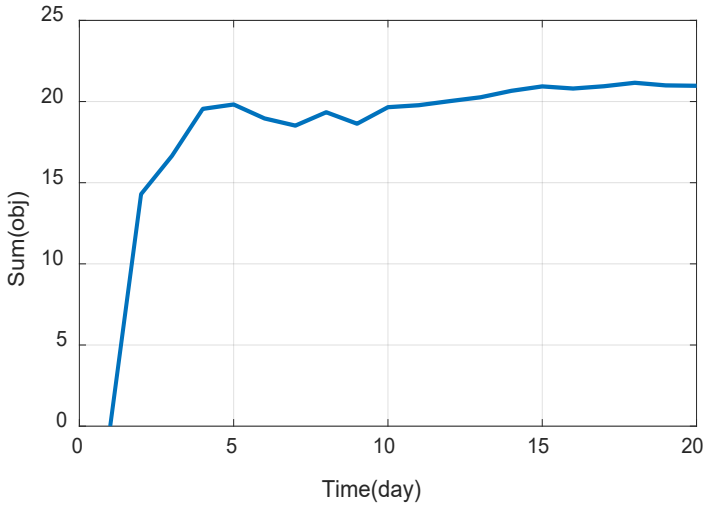

Figure 53: System performance with learning rate b=0.2

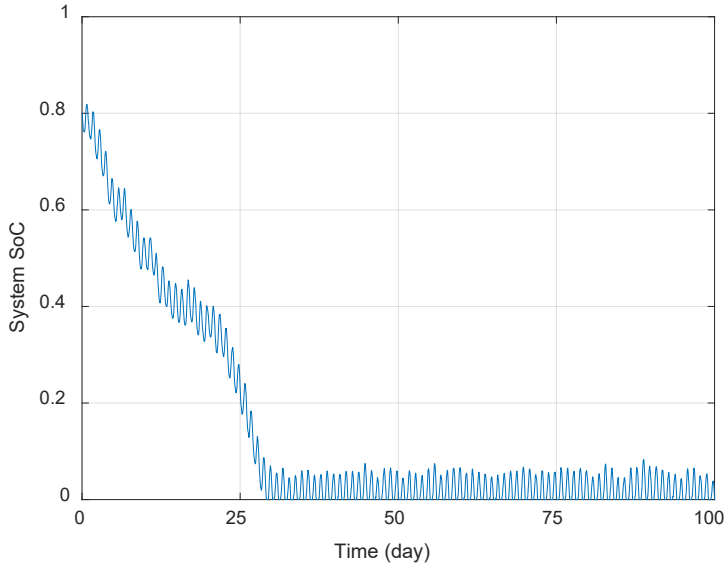

Figure 54: System SoC with learning rate b=1.0

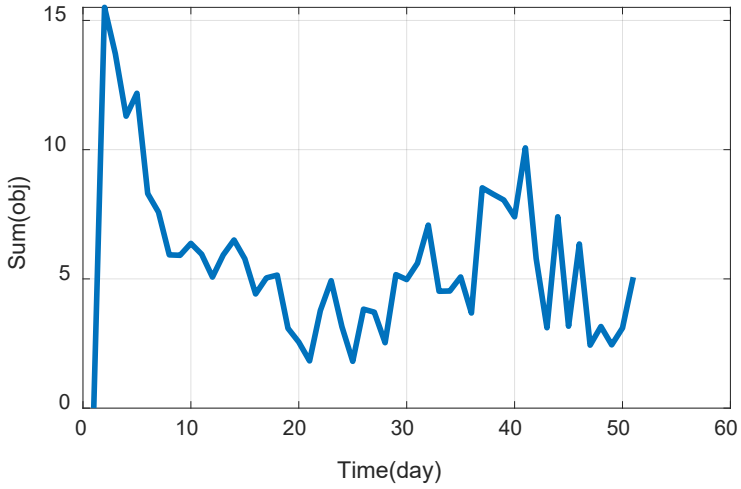

Figure 55: System performance with learning rate b=1.0

6.3.2 Partial Loss of Power Source

In this simulation, the Linear-reward inaction is tested in an environment where a sudden loss of the PV power occurs. During the operation, the PV power was cut to 80% of its estimated value, starting from day 50 to day 100. The system SoC and performance without a low-SoC barrier are shown in Figure 56 and Figure 57. As the results show, the learning algorithm failed to maintain the system SoC and did not recover from the power lost even when the power is back. In another simulation, the low-SoC barrier is applied to the agents as *top=0.3* (from section 5.3). The system SoC and performance with this setting are shown in Figure 58 and Figure 59. These results show that the system survived the power loss event with some loss of battery energy. After the power loss event, the battery SoC level was recovered to the desired level. Also, as could be seen from the obtained CTSF strategy chart shown in Figure 60, the low-SoC parts are well guarded by the high-probability low-CTSF 'walls'.

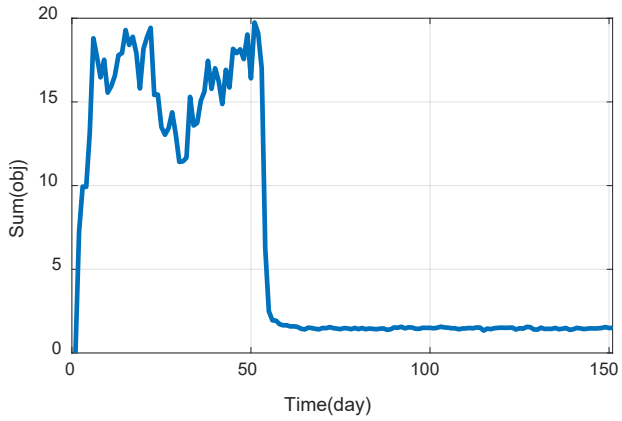

Figure 56: System performance with power lost from day 50 to day 100

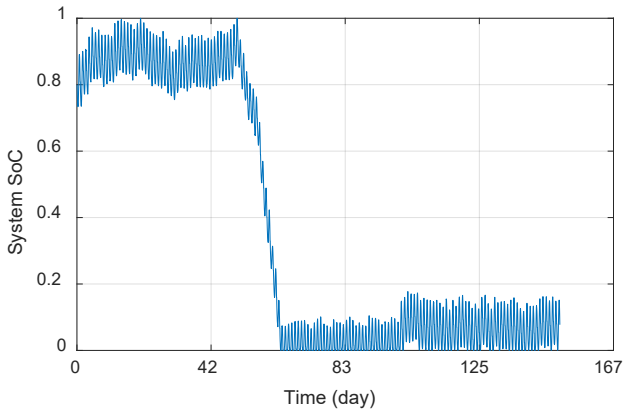

Figure 57: System SoC with power lost from day 50 to day 100

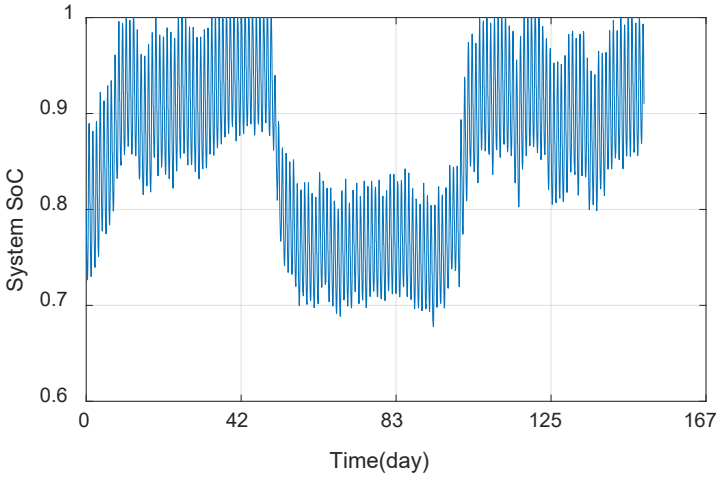

Figure 58: System SoC with power lost with low-SoC barrier

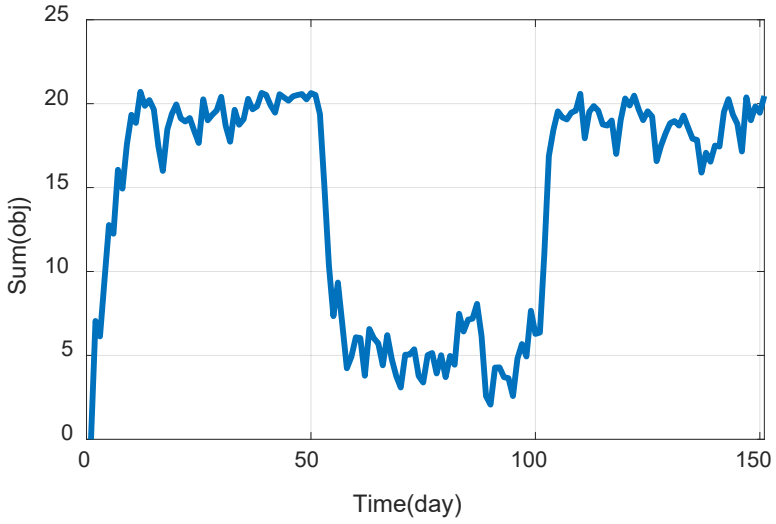

Figure 59: System performance with power lost from with low-SoC barrier

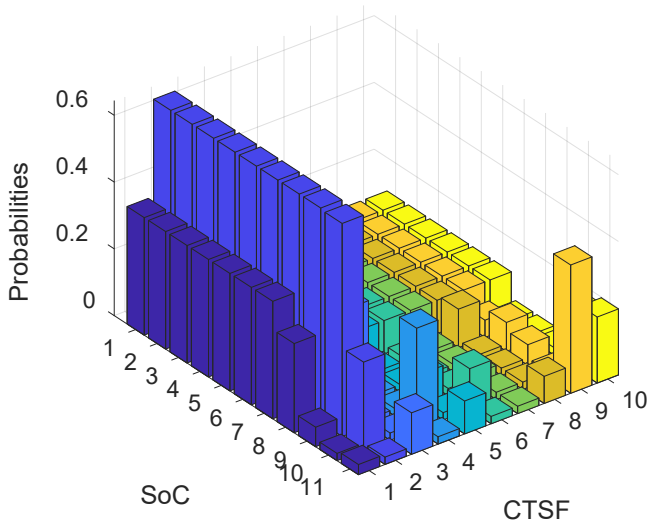

Figure 60: CTSF strategy chart obtained with low-SoC barrier

6.3.3 Loss of Communication

In this section, the behavior of the microgrid without communication is demonstrated. It is assumed in this scenario that the communication link in this microgrid is cut off, and the BSs in this system could only utilize their local information and the approximated reward function (4-18) to conduct the learning process. If the other parameters remain the same, the agents can not find a steady strategy that satisfies the original goal as shown in Figure 61. The result shows that the learning rate is too large for the agents to converge with the local data. The performance of the system is better when the learning rate b is replaced by a smaller number. As shown in another simulation, the learning rate is set to *b=0.05,* and the resulting system performance is shown in Figure 62. These results show that compared to the normal operation, the agents implementing Linear-reward inaction without communication are less stable, and the training period is longer (around 20 days).

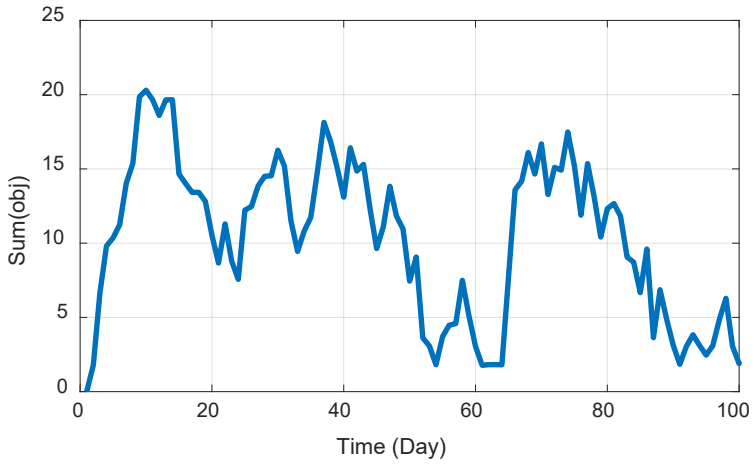

Figure 61: System performance applying local objective function (b=0.1)

However, if the number of BSs is larger, the performance of the RL using local objective function becomes unstable again, as can be seen from the simulation result in Figure 63. One attempt that has been done to overcome this is to have an even smaller learning rate. With a learning rate of b=0.01 the performance is stabilized as shown in Figure 64. But this smaller learning rate results in a longer training period as shown in Figure 65, which is approaching 100 days.

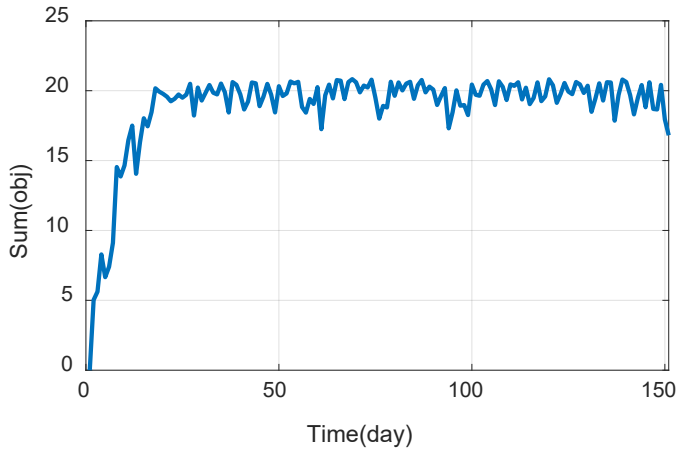

Figure 62: System performance applying local objective function (b=0.05)

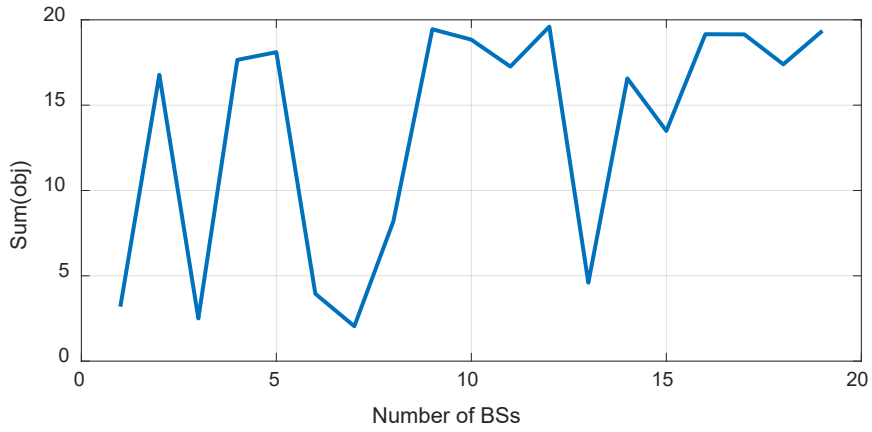

Figure 63: Performance of system applying RL with local objective function (b=0.05) and different number of BSs

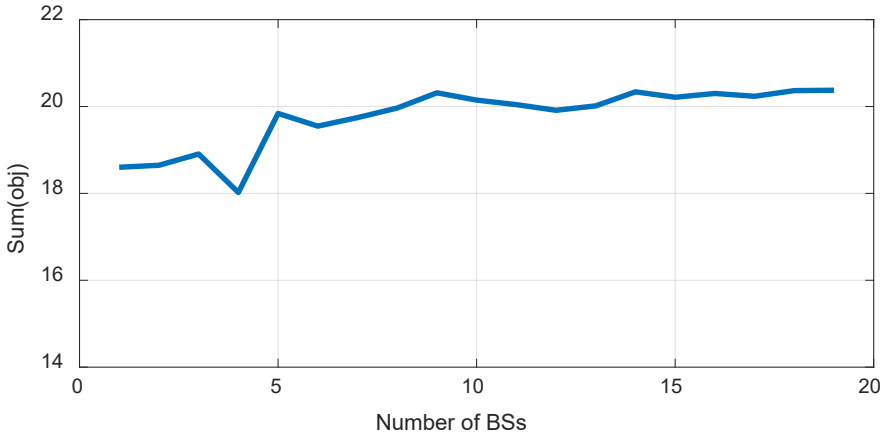

Figure 64: Performance of system applying RL with local objective function (b=0.01) and different number of BSs

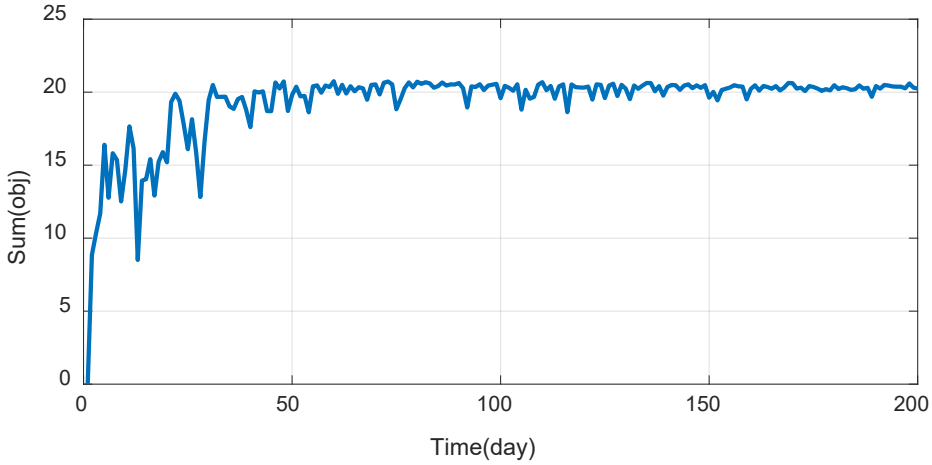

Figure 65: Learning curve of system applying RL with local objective function (b=0.01)

6.4 Load-Ratio Learning Game

6.4.1 Normal Operation

In this simulation, the load ratio learning game algorithm was applied to the communication microgrid in normal conditions. The number of BSs in this microgrid is three, and the total power generation and load consumption are the same as the Linear-reward inaction simulation. There are five available load ratio stages for the BSs: $[0.2, 0.4, 0.6, 0.8, 1.0]$. The initial load-ratio policy of the load ratios of each BS is

$$l_r = [0.1, 0.6, 0.1, 0.1, 0.1] \tag{6-4}$$

At each CTSF choosing time, each BS controller picks a load ratio according to the probabilities in the load-ratio policy. Once the load ratio r_{t_i} is chosen, the system load is computed as

$$P_{load}{}^{i} = P_B + r_{t_i}\sigma_i v P_T + (1 - r_{t_i})\sigma_o v P_T \tag{6-5}$$

Then the objective function is calculated and the two-player game with choices of σ_i and σ_o could be formed and solved either in zero-sum or non-zero-sum settings. After the CTSF decision, the system battery SoC is measured and fed into the reward function along with the CTSF chosen by BS controllers and the load-ratio list is updated. The learning rate is set to be 0.1, and the SoC weighting factor is 0.5 in this simulation.

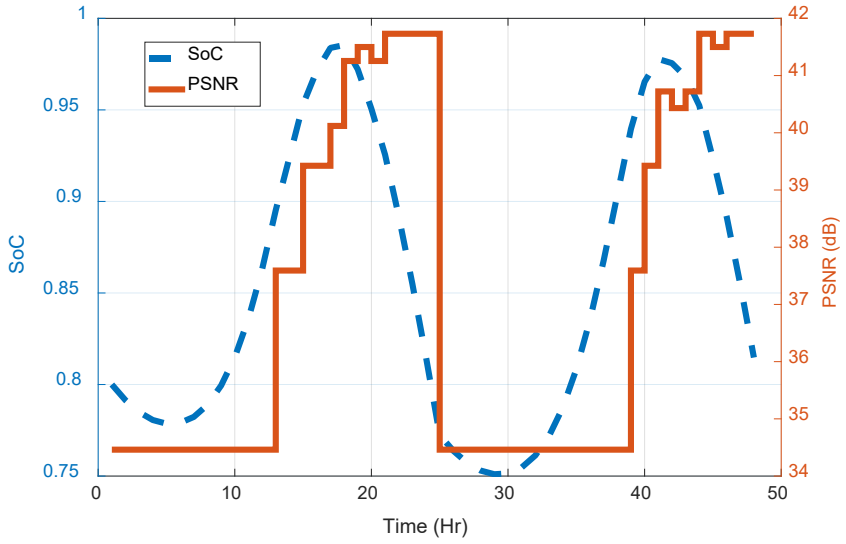

Figure 66: PSNR and SoC of BS microgrid applying load-ratio learning game algorithm, normal condition

The PSNR and system SoC of the simulated microgrid implemented with te load-ratio learning game algorithm are shown in Figure 66. As the result shows, the desired SoC goal is reached and the PSNR is above 35 dB most of the time. The complete learning curve of this algorithm is shown compared to the direct Linear-reward approach in Figure 67. As the learning curve shows, the algorithm has a higher initial performance and requires almost no learning time. Other than this, a performance comparison of the learning-gaming algorithm with different number of BSs in the

microgrid is conducted as well. The system performance applied with the two-player zero-sum game, direct RL (Linear-reward inaction), and the load-ratio learning game algorithms are plotted in the same figure as shown in Figure 68. The training times for both learning algorithms are ten days. This result reveals that the learning-gaming algorithm has a better performance compared to solely applying RL and the virtual two-player game approach.

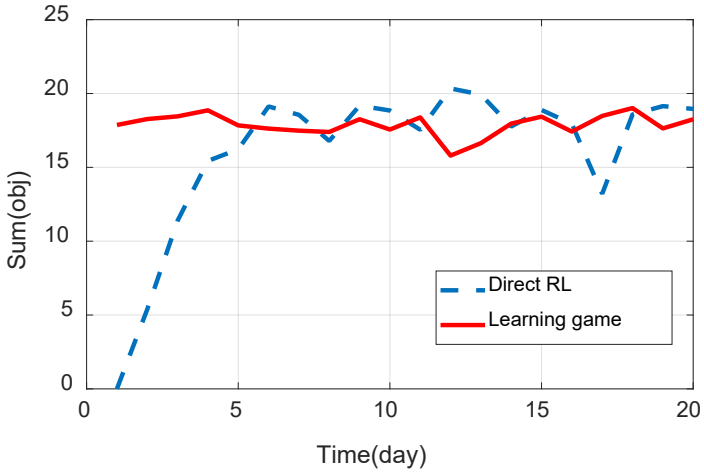

Figure 67: Comparison of learning curves of RL and learning-game algorithm

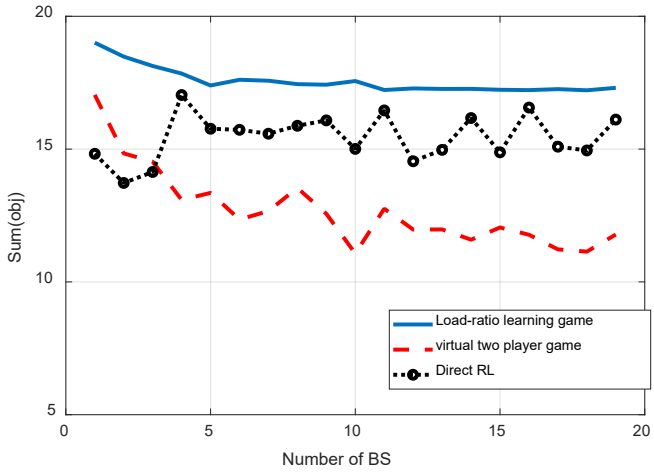

Figure 68: Comparison of algorithms with different number of BSs

6.4.2 Partial Loss of Power Source

In this scenario, the adaptation feature of the load-ratio learning game algorithm is tested when the system is affected by a natural disaster. In this simulation, it is assumed a hurricane hits the microgrid at day 20. The power lines are intact, however, due to the physical damage taken by the solar panels and BSs, the communication links between BSs are cut off, and the output of all power generator is limited to 60% of its original rating. In this condition, the BS controllers utilize local information to calculate their rewards. The resulting average system PSNR and SoC are shown in Figure 69. The system experienced an SoC decreasing when the power is lost but managed to restore the energy level in several days. The system performance, as shown in Figure 70, also demonstrated a descending after the power loss and is recovered after days of adaption. The final load-ratio strategy obtained by the learning algorithm is shown in Figure 71, where one of the BSs adjusted its confident load-ratio to 0.6 instead of the original 0.4. In this scenario, the higher load-ratios make the BS ‘think’ it needs to feed more load, thus resulting in a lower CTSF strategy.

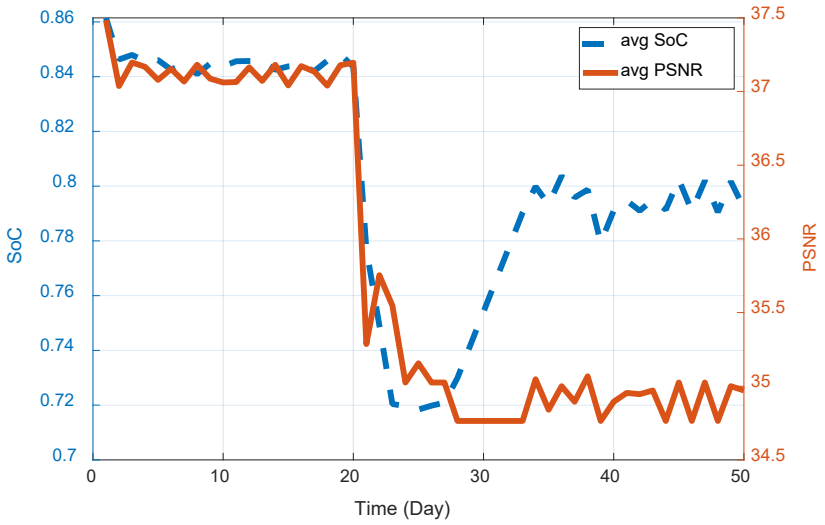

Figure 69: Average system SoC and PSNR with power loss

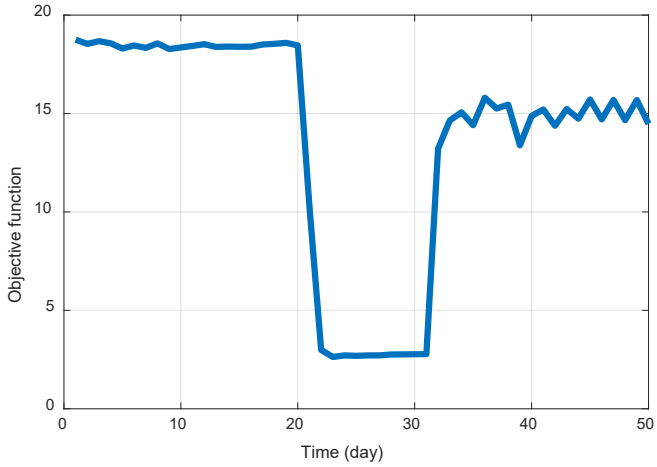

Figure 70: System performance index with power loss

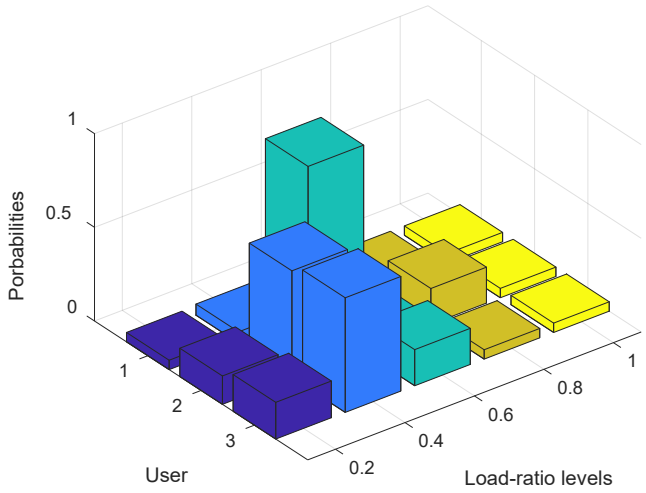

Figure 71: Obtained load-ratio strategy with power loss

6.5 Discussion

The simulation results have demonstrated the features and applicable conditions of the proposed algorithms. First, the two-player game approach works well for a small scale microgrid. The microgrid performance applying the two-player game measured in the objective function is comparable to that of an exhaustive search under most configurations. However, as the number of BSs increases, the virtual two-players game experiences a performance drop. Compared to the zero-sum game, the performance drop of a microgrid implemented with the common-interest game is more substantial. The simulation results of a microgrid implemented with the reinforcement learning algorithm also leave us some unanswered questions. The single-agent Q-learning has the highest performance score, but the multiple agent Q-learning failed the system objective in some cases. The mechanism and stability conditions of multi-agent Q-learning are yet to be discovered. The Linear Reward-Inaction algorithm guarantees a converging strategy with a sufficiently small

learning rate. However, whether the algorithm converges to the desired equilibrium is not always guaranteed, which might also be related to the design of the objective and reward functions. Another constraint that shakes the Linear Reward-Inaction algorithm's dominating role is the long training time due to its learning rate limit. Lastly, the load-ratio learning game approach seems to be the ideal solution for now: it has a better performance compared to both the RL and game approach, yet its training time is much shorter. Also, it could be easily extended to a microgrid with an arbitrary number of BSs. However, this algorithm is only tested in a limited number of cases. So its full behavior under different circumstances is yet to be discovered.

7.0 Conclusions

This dissertation studied energy management in a microgrid consists of communication base stations with the help of distributed control algorithms. The goal was to develop an autonomous control system for communication base stations so that they could perform load response and realize energy management considering communication quality and energy availability without the need for a designated communication network. This feature decreases the dependency of elements in such microgrid thus could increase the system's overall resilience against natural disasters or other accidents.

Three attempts: multi-player game, reinforcement learning, and load-ratio learning game were made to accomplish the goal. In the game approach, the original load response decision-making process is modeled as a multi-player game and solved by BS controllers independent. Depending on the setting of the game, the solution of such a game could be the form of a pure or mixed strategy. Also, the obtained energy management strategy is comparable to that of an exhaustive search. However, for a general game with an arbitrary number of players, the solving process is intractable thus is not practical to be implemented in the microgrid system. This constraint of game theory has limited its application since a communication network often consists of many BSs. To overcome this shortcoming, we have expanded the two-player game solving process to any number of players with the virtual player setting. However, such a setting experiences a performance descending as the number of players increases.

The second attempt proposed was applying a reinforcement learning algorithm to BS controllers. This approach makes the BS controllers update their load response strategies based on

their experienced interactions with the microgrid. Two different learning algorithms were applied: Q-learning and Linear Reward-Inaction. The Q-learning algorithm showed the highest performance score in the single-agent scenario, but its convergence in a multi-agent environment is not guaranteed. Linear Reward-Inaction ensures the converging of the agent's strategies, but the algorithm requires a long training time before the optimal strategy is obtained. The tuning of the learning rates is the keys to both of the learning algorithms.

In the end, a load-ratio learning game algorithm was developed and tested. This algorithm was meant to solve the virtual two-player game performance deterioration. However, it turned out that the virtual two-player game with updated load-ratio has a better performance compared to direct RL and virtual two-player games regardless of the number of players. Additionally, benefit from its smaller size of search space, the load-ratio learning converges much faster than the conventional RL algorithm with the same learning rate. Therefore, based on the results done in this project, the load-ratio learning game algorithm showed the greatest potential in satisfying the communication microgrid energy management requirement.

Appendix A Simulation code

Two-player game simulation main code:

```
function bytest_game
tic
data=zeros(1,sample_size);
day=2;
n=24*day;
SoC=zeros(1,n);
SoC(1)=0.7;
v=zeros(1,n);
alpha=zeros(1,n);
v(1)=(SoC(1)-0.5)/0.05+56;
Energy=1*25*1000;
soc_weight=0.5;
wa=1-soc_weight;
batterystage=40;
cost=zeros(1,n);
top=2;
amin=1;
daygoal=0.8;
phi=ones(1,top);
phireal=1;
rate_o=ones(1,top);
sitenumber=2;
realrate=rate_o(1:sitenumber)/sum(rate_o(1:sitenumber));
add=0;

for t0=1:n

t=rem(t0,24);
if t==0
   t=1;
end


[alpha0,~,self]=gamesolver(v(t0),t,wa,soc_weight,batterystage,Energy,amin,day
goal,phi,sitenumber,realrate(1:sitenumber));

alpha(t0)=alpha0;

SoC(t0+1)=(SoC(t0)*Energy-
integral(@(x)load2(x),t,t+1)*alpha(t0)+integral(@(y)solar(y),t,t+1))/Energy*(
1+0.0*(rand-rand));

[~,cost(t0)]=evaluation(alpha(t0),t,batterystage,Energy,SoC(t0),soc_weight,da
ygoal,phireal,add,0,0);

if SoC(t0+1)>=1
   SoC(t0+1)=1;
```

```
end
if SoC(t0+1)<=0
   SoC(t0+1)=0;
end

v(t0+1)=(SoC(t0+1)-0.5)/0.05+56;
end

cg=sum(cost)/day;

end
```

Two-player game Sub-functions:

Solar power function:

```
function Psolar=solar(ti)
t=ti*5;
p1 =      -327.4 ;
p2 =   4.435e+04;
p3 =  -7.078e+05;
q1 =      -143.6 ;
q2 =        5928 ;
Psolar =(1*(p1*t.^2 + p2*t + p3) ./ (t.^2 + q1*t + q2)+80)*(1+(0.2*rand-
0.2*rand)*1);
end
```

BS load function:

```
function Pload=load2(t)
pl1 =    0.000839 ;
pl2 =       1.205  ;
pl3 =      -12.02 ;
pl4 =       34.29  ;
ql1 =      -6.495  ;
ql2 =       43.45  ;
Pload =max(1*(500* (pl1*(t).^3 + pl2*(t).^2 + pl3*(t) + pl4) ./((t).^2 +
ql1*(t) + ql2)+500)*(1+(0.2*rand-0.2*rand)*1),0);
end
```

Two-player game solver:

```
function [ko,alphao, kself] =
gamesolver(v,t,~,b0,batterystage,Energy,amin,daygoal,phi,sitenumber,rate)
alphao=zeros(1,sitenumber);
kself=zeros(1,sitenumber);
SoC=(v-56)*0.05+0.5;
daygoal2=daygoal*batterystage;
T=1;
```

```
n=20;
dT=T/n;
realp=zeros(1,9);
coder.extrinsic('linearprograming')
coder.varsize('cost1','cost2','realrow','realcolumn')
decide=zeros(1,sitenumber);



for ppt=1:sitenumber
    opteva=zeros(9);
    cost1=zeros(9);
    loadrate=rate(ppt);
for k0=amin:1:9
 for sigmaother0=amin:1:9
        k=k0*0.1;
        sigmaother=sigmaother0*0.1;
        [~,Sn]=pdf2(t,dT,loadrate,k,sigmaother,n*(24-t),phi(ppt));
         En=-integral(@(x)load2(x),t,24)*(k*loadrate+(sigmaother*(1-
loadrate)))+1*integral(@(y)solar(y),t,24)*(1);
        if t~=24
        pr = normcdf(((Energy*daygoal-SoC*Energy)*1-En)/sqrt(Sn));
        if 1-pr>0.5 %if there is  possibility reaching 0.8 SoC?
            opteva(k0,sigmaother0)=b0*(1-pr);
            D=latency_indicator(t,k*loadrate+(sigmaother*(1-loadrate)),1);
            cost1(k0,sigmaother0)=(D)*(1-b0)+opteva(k0,sigmaother0);
        else  %if not stay at minimum alpha
            cost1(k0,sigmaother0)=(1-(k*loadrate+sigmaother*(1-
loadrate)))/10;
        end
        else
            cost1(k0,sigmaother0)=(k*loadrate+sigmaother*(1-loadrate))*(1-
b0)+b0*SoC;
        end
 end
end
 [m,~]=size(cost1);
  realrow=1:m;
  p=zeros(1,9);

%nonlinear splver
p(1:amin-1)=0;
p10=non_linear_solver(cost1(amin:9,amin:9),cost1(amin:9,amin:9)');
p(amin:9)=p10(1:10-amin);

%linear solver
% p(1:amin-1)=0;
% p10=linearprograming_real(cost1(amin:9,amin:9));
% if size(p10)==0
%     p10=ones(1:9)/9;
% end
% p(amin:9)=p10(1:(10-amin));


realn=size(realrow,2);
for i=1:realn
```

```
    realp(realrow(i))=p(i);
end
% for z=1:sitenumber
r=rand;
l=0;
    for i=1:9
        l=l+realp(i);
        if l>r
            break
        end
    end
decide(ppt)=i*0.1;

end
kself=decide;
ko=decide*rate(1:sitenumber)'/sum(rate(1:sitenumber));
end
```

Objective function:

```
function
[D,cost]=evaluation(k,t,~,Energy,SoC,w_soc,daygoal,phireal,add,deadload,guess
)
T=1;
n=20;
dT=T/n;
[En,Sn]=pdf2_add(t,dT,1,k,0,n*(24-t),phireal,add,deadload,guess);
 En=-integral(@(x)load2(x),t,24)*(k)+1*integral(@(y)solar(y),t,24)*(phireal);
if t~=24
        p = normcdf(((Energy*daygoal-SoC*Energy)-En)/sqrt(Sn));
        if 1-p > 0.5  %if there is  possibility reaching 0.8 SoC?
          D=latency_indicator(t,k,1);
%           D=k;
          opteva=w_soc*(1-p);
          cost=(D)*(1-w_soc)+opteva;
        else  %if not stay at minimum alpha
           cost=(1-k)/10;
%            D=k;
           D=latency_indicator(t,k,1);
        end

else
     D=latency_indicator(t,k,1);
     cost=D*(1-w_soc)+w_soc*SoC;

%     D=k;
end
end
```

Q-learning single-agent code:

```
function bytest_Q_learning
day=90;
n=24;
record=zeros(1,day*24);
SoC_record=zeros(1,day*24);
SoC=zeros(1,n);
soc1=0.7;
SoC(1)=soc1;
v=zeros(1,n);
alpha=zeros(1,n);
v(1)=(SoC(1)-0.5)/0.05+56;
vpre=v(1);
Energy=1*25*1000;
b0=0.2;
batterystage=20;
cost=zeros(1,n);
dump1=50;
dump2=100;
daygoal=0.8;
phireal=0.5;
%battreryin time
batteryin=inf;
%new load time
timecheckin=inf;
timecheckout=90;
%rule breaker:added load
dump11=90;
dump22=10;
deadload=-500;
limitt=3;
phi=[1 1 1 1 1];
operation=zeros(1,limitt);
operation(1:3)=ones(1,3);
sitenumber=limitt;
checkflag=0;
nstrategy=5;
batterysnumber=5;
Qvalue=load('Q_table');
Qvalue=Qvalue.Q_table;
alphaset=zeros(1,nstrategy);
for i=1:nstrategy
    alphaset(i)=1/nstrategy*i;
end
show=zeros(1,24);
cost_record=zeros(1,n*day);
kpp=1;
kt=1;
for t0=1:n*day
    daymark=floor(t0/24)+1;
    if t0>batteryin&&checkflag==0
        checkflag=1;
        SoC(t0)=SoC(t0)+0.5;
    end
    t=mod(t0,24)+(mod(t0,24)==0)*24;
```

```
    if t~=24
        tz=floor(t/8)+1;
    else
        tz=1;
    end

    if t>dump1&&t<dump2&&cheat==1
        phi=[0.5 0.5 0.5 0.5];
    end
    if t0>timecheckin
       operation(4)=1;
    end
    if t0>timecheckout
        operation(3:4)=[0 0];
    end
%state verifying
    v(t)=vpre;
    if t0<=30 && t==1
       SoC(t)=rand()*0.5+0.2;
    else
       SoC(t)=(v(t)-56)*0.05+0.5;
    end
state=floor(SoC(t)/(1/batterysnumber))+1;

%%%Q-learning process%%%
    sumQall=0;
    Qvalue_t=zeros(1,nstrategy);
    for inner=1:nstrategy
        if mod(t0,24)~=0
        Qvalue_t(inner)=(Qvalue(tz,state,inner));
        sumQall= sumQall+Qvalue_t(inner);
        else
        Qvalue_t(inner)=(Qvalue(tz,state,inner));
        sumQall= sumQall+Qvalue_t(inner);
        end
    end
%%%end learning%%%

    for inner=1:nstrategy
        chance(inner)=Qvalue_t(inner)/sumQall;
    end

        if mod(t0,24)~=0
        [~, idx]=max(chance);
        show(t)=alphaset(idx);
        else
        [~, idx]=max(chance);
        show(24)=alphaset(idx);
        end

    r=rand;
    l=0;
    for i=1:nstrategy
      l=l+chance(i);
      if l>r
```

```
            break
        end
    end
    alphareal=alphaset(idx);
    record(t0)=alphareal;

alpha(t)=alphareal;
cost(t)=evaluation(alphareal,t,batterystage,Energy,SoC(t),b0,daygoal,1-
phireal*(t0>=dump1&&t0<=dump2),phi(sitenumber)*(t>timecheckin),deadload*(t0>d
ump11&&t0<dump22),0);
SoC(t+1)=(SoC(t)*Energy-deadload*(t0>dump11&&t0<dump22)-
integral(@(x)load2(x),t,t+1)*alphareal+integral(@(y)solar(y),t,t+1)*(1-
phireal*(t0>dump1&&t0<dump2)))/Energy;
if SoC(t+1)>=1
   SoC(t+1)=1;
elseif SoC(t+1)<=0
   SoC(t+1)=0;
end

    SoC_record(kt)=SoC(t);
    kt=kt+1;
    if mod(t0,24)==0
        % record cost sum of one day
        cost_record(kpp)=sum(cost);
        kpp=kpp+1;
    end
    v(t+1)=(SoC(t+1)-0.5)/0.05+56;
    vpre=v(t+1);

end

end
```

Q-learning multi-agent code:

```
function RL_no_time_main()
tic
global random_switch
random_switch=0;
day=50;
n=24;
SoC=zeros(1,n);
soc1=0.8;
SoC(1)=soc1;

SoC_record=zeros(1,24*day);

v=zeros(1,n);
alpha=zeros(1,n);
v(1)=(SoC(1)-0.5)/0.05+56;
vpre=v(1);
Energy=1*25*1000;
cost=zeros(1,n);
dump1=500*24;
```

```
dump2=150*24;
daygoal=0.8;
phireal=0.2;
cost_record=zeros(1,day);
kpp=1;
agent_1_r=zeros(1,day);

%battreryin time
batteryin=inf;

%new load time
timecheckin=inf;
timecheckout=90;

%rule breaker:dead load
dump11=500;
dump22=10;
deadload=-500;

%number of BS controllers
limitt=10;

%BS load rate
load_rate=ones(1,limitt)/limitt;

%new connected device index
phi=[1 1 1 1 1];

%operation status index
operation=zeros(1,limitt);
operation(1:3)=ones(1,3);


%new battery plug in flag
checkflag=0;

% Q learning step size
aupdate=0.1;



% Horizon factor
gamma=0.9;

%available strategy number
nstrategy=20;

%battery level
batterysnumber=20;

%ndividualcounter for SoC levels
soc_count_t=zeros(limitt,batterysnumber+1);
```

```
%initialize Q value table
Qvalue=ones(limitt,batterysnumber+1,nstrategy)/10;

% for i=1:batterysnumber
%     for j=1:nstrategy
%         Qvalue(:,i,j)=normrnd(1,1);
%     end
% end

%initilize strategy index
alphaset=zeros(1,nstrategy);
for i=1:nstrategy
    alphaset(i)=1/nstrategy*i;
end
learning_count=0;
ini=0;

alpha_all=0;

for t0=1:n*day
    daymark=floor(t0/24)+1;

%     if t0>batteryin&&checkflag==0
%         checkflag=1;
%         SoC(t0)=SoC(t0)+0.5;
%     end

    %get time of one day
    t=mod(t0,24)+(mod(t0,24)==0)*24;

    %power outage flag
%     if t>dump1&&t<dump2&&cheat==1
%         phi=[0.5 0.5 0.5 0.5];
%     end

    % new load kick in
    if t0>timecheckin
       operation(4)=1;
    end
    % load 3 and 4 disconnected
    if t0>timecheckout
        operation(3:4)=[0 0];
    end

%SoC state verifying
% if daymark<60&&t==1
%     SoC(t)=rand*0.8+0.2;
% else
    v(t)=vpre;
    SoC(t)=(v(t)-56)*0.05+0.5;
% end

SoC_record(t0)=SoC(t);
    state=floor(SoC(t)/(1/batterysnumber))+1;
%%% making decisions %%%
```

```
% for each BS controller
     alphareal=zeros(1,limitt);
     idx=zeros(1,limitt);
    for agent=1:limitt
        sumQall=0;
        sumQvalue=zeros(1,nstrategy);

    % calculate sum of Q value rows
        for inner=1:nstrategy
            sumQvalue(inner)=(Qvalue(agent,state,inner));
            sumQall= sumQall+sumQvalue(inner);
        end

%     for inner=1:nstrategy
%         chance(inner)=sumQvalue(inner)/sumQall;
%     end
    tt=0.5;
    sumQvalue=exp(Qvalue(agent,state,:)/tt);
    % softmax function
    chance=zeros(1,nstrategy);
    for inner=1:nstrategy
        chance(inner)=exp(Qvalue(agent,state,inner)/tt)/sum(sumQvalue);
    end

    % choose the action with maximum Q value
    [~, idx(agent)]=max(chance);

    if daymark<inf
    rr=rand;
    l=0;
    for i=1:nstrategy
      l=l+chance(i);
      if l>rr
            break
      end
    end

    idx(agent)=i;
    end

    alphareal(agent)=alphaset(idx(agent));

%     record(daymark,t)=idx;
%%%end trying%%%
    end

%     cost(t)=evaluation(alphareal,t,batterystage,Energy,SoC(t),b0,daygoal,1-
phireal*(t0>=dump1&&t0<=dump2),phi(sitenumber)*(t>timecheckin),deadload*(t0>d
ump11&&t0<dump22),0);
    alphareal_total=alphareal*load_rate(1:limitt)';
    alpha(t)=alphareal_total;
    alpha_all=alpha_all+alpha(t);
    SoC(t+1)=(SoC(t)*Energy-deadload*(t0>dump11&&t0<dump22)-
integral(@(x)load2(x),t,t+1)*alphareal_total+integral(@(y)solar(y),t,t+1)*(1-
phireal*(t0>dump1&&t0<dump2)))/Energy;
```

```
    if SoC(t+1)>=1
        SoC(t+1)=1;
    elseif SoC(t+1)<=0
        SoC(t+1)=0;
    end

%     if mod(t0,24)==0
%         SoC(t+1)=soc1;
%     end
    v(t+1)=(SoC(t+1)-0.5)/0.05+56;
    vpre=v(t+1);
%%%Reinforcement learning updating%%%
%new state
    newstate=floor(SoC(t+1)/0.1)+1;
    %difference between previous two days' rewards
    r_diff=0;
    if kpp~=1
        r_diff=agent_1_r(kpp)-agent_1_r(kpp-1);
    end

    if abs(r_diff)>=5&&ini==1
        learning_count=10*24;
        ini=1;
    end
    if learning_count>0
        learning_count=learning_count-1;
        time_gap=t0-24-learning_count;
        t0_in=t0-time_gap;
    else
        t0_in=t0;
    end
    for agent=1:limitt
        soc_count_t(agent,state)=soc_count_t(agent,state)+1;

[r_inst,Qvalue(agent,state,idx(agent))]=RL_notime_max_com_soc_double(newstate
,SoC(t),SoC(1),alphareal(agent),alpha_all,Qvalue(agent,state,idx(agent)),Qval
ue(agent,:,:),batterysnumber,nstrategy,t,t0_in,gamma,aupdate,soc_count_t(agen
t,state));
    end
%         agent_1_r(kpp)=agent_1_r(kpp)+r;
        agent_1_r(kpp)=agent_1_r(kpp)+r_inst;
        if t==24
            kpp=kpp+1;
            ini=0;
            alpha_all=0;
            vpre;
        end

end


     figure('Name','Game solution')

     z=plotyy(1:n,SoC(1:n),1:n,alpha);
     tail1=sum(SoC)/n;
     tail2=sum(alpha)/n;
```

```
%       stem(1:n,cost)
     legend('SoC','\sigma')
     xlabel('Time (Hr)')
     ylabel(z(1),'SoC') % left y-axis
     ylabel(z(2),'Traffic Shaping factor \sigma','Fontsize',20)
% %     i=1:24;
% %     figure
% %     sum(cost)
% %     plot(i,show(i))
     i=1:day;
     figure
     plot(i,agent_1_r(i))
    assignin('base', 'Q_table', Qvalue)

% % figure
% % [d,y]=meshgrid(1:1:24,1:1:day+1);
% % surf(d,y,record)
% assignin('base', 'Q_table', Qvalue)
figure
plot(SoC_record);
figure
bar3(squeeze(Qvalue(1,:,:)))
toc
end
```

Linear-reward inaction code:

```
function bytest_RL_more_player_compare
tic
day=50;
batterysnumber=20;
n=24*day;
SoC=zeros(1,n);
SoC(1)=0.7;
v=zeros(1,n);
alpha=zeros(1,n);
v(1)=(SoC(1)-0.5)/0.05+56;
Energy=3*25*1000;
w_soc=0.5;
wsoc_uni=0.5;
cost=zeros(1,24);
D=zeros(1,n);
sitenumber=3;

amin=2;
dump1=24*500;
dump2=24*200;
daygoal=0.8;
phireal=1;
r0=zeros(1,day+1);
agent_1_r=zeros(1,day);
cg=zeros(1,sitenumber);
load_rate=ones(1,sitenumber)/sitenumber;
SoC_record=zeros(1,24*day);
```

```
alpha_record=zeros(1,24*day);
%rule breaker:dead load
dump11=90;
dump22=10;
deadload=-500;



%available strategy number
nstrategy=20;
alpha_all=0;
cost_record=zeros(1,day);
alphaset=zeros(1,nstrategy);
for i=1:nstrategy
    alphaset(i)=1/nstrategy*i;
end

for site_index=3:sitenumber
    limitt=site_index;
    kpp=1;
    % rate=ones(1,site_index);
    p_option=1/nstrategy*ones(limitt,batterysnumber+1,nstrategy);

 for t0=1:n
    daymark=floor(t0/24)+1;

%     if t0>batteryin&&checkflag==0
%         checkflag=1;
%         SoC(t0)=SoC(t0)+0.5;
%     end

    %get time of one day
     t=mod(t0,24)+(mod(t0,24)==0)*24;

%SoC state verifying
% if daymark<60&&t0==1
%     SoC(t0)=rand*0.8+0.2;
% else


    if SoC(t0)>1
        SoC(t0)=1;
    end
% end

    SoC_record(t0)=SoC(t0);
    state=floor(SoC(t0)/(1/batterysnumber))+1;
%%% making decisions %%%
    idx=zeros(1,limitt);
    alphareal=zeros(1,limitt);
    for agent=1:limitt

    % softmax function
    chance=zeros(1,nstrategy);
```

```
    for inner=1:nstrategy
        chance(inner)=p_option(agent,state,inner);
    end

    if daymark<inf
    rr=rand;
    l=0;
    for i=1:nstrategy
      l=l+chance(i);
      if l>rr
            break
      end
    end

    idx(agent)=i;
    end

    alphareal(agent)=alphaset(idx(agent));

%     record(daymark,t0)=idx;
%%%end trying%%%
    end
%
cost(t0)=evaluation(alphareal,t0,batterystage,Energy,SoC(t0),b0,daygoal,1-
phireal*(t0>=dump1&&t0<=dump2),phi(sitenumber)*(t0>timecheckin),deadload*(t0>
dump11&&t0<dump22),0);
    alphareal_total=alphareal*load_rate(1:limitt)'/sum(load_rate(1:limitt));
%     alpha(t0)=alphareal_total;
    alpha(t0)=alphareal(1);
    alpha_record(t0)=alphareal_total;
    alpha_all=alpha_all+alpha(t0);
%     if t0>=2500
%         k=1;
%     end
    SoC(t0+1)=(SoC(t0)*Energy-deadload*(t0>dump11&&t0<dump22)-
integral(@(x)load2(x),t,t+1)*alphareal_total+integral(@(y)solar(y),t,t+1)*(1-
phireal*(t0>dump1&&t0<dump2)))/Energy;

[D_record_day(t0),cost(t)]=evaluation(alphareal_total,t,batterysnumber,Energy
,SoC(t0),wsoc_uni,daygoal,phireal,0,0,0);
    if SoC(t0+1)>=1
        SoC(t0+1)=1;
    elseif SoC(t0+1)<=0
        SoC(t0+1)=0;
    end

%     if mod(t0,24)==0
%         SoC(t0+1)=soc1;
%     end
%     v(t0+1)=(SoC(t0+1)-0.5)/0.05+56;
%     vpre=v(t0+1);
%     SoCpre = SoC (t0+1);

    %%%Reinforcement learning updating%%%
%new state
    newstate=floor(SoC(t0+1)/(1/batterysnumber))+1;
```

```
    %difference between previous two days' rewards


    for agent=1:limitt

[r1,p_option(agent,state,:)]=RL_reard_inauction_common(newstate,SoC(t0+1),alp
hareal(agent),p_option(agent,state,:),idx(agent),nstrategy,t,limitt,w_soc,cos
t(t));
    End
% Low-soc barrier
%     for i=1:limitt
%             for j=1:nstrategy
%                 if p_option(i,state,j)>=0.8
%                     p_option(i,state,j)=0.8;
%                 end
%                 if p_option(i,state,j)<=0.2/(nstrategy-1)
%                    p_option(i,state,j)=0.2/(nstrategy-1);
%                 end
%             end
%     end
%         for i=1:limitt
%             for j=1:3
% %                 if p_option(i,state,j)>=0.3
%                     for k=1:15
%                         p_option(i,k,j)=0.3;
%                     end
% %                 end
%             end
%         end

    for i=1:limitt
            for j=1:nstrategy
            p_option(i,state,j)=p_option(i,state,j)/sum(p_option(i,state,:));
            end
    end
    r0(daymark)=r0(daymark)+r1;
%         agent_1_r(kpp)=agent_1_r(kpp)+r;
        agent_1_r(kpp)=agent_1_r(kpp)+r1;
        if t==24
            SoC(t0+1)=SoC(t0);
            cost_record(daymark)=sum(cost);
%             D_record_day=zeros(1,24);
            agent_1_r(kpp)=agent_1_r(kpp)+(SoC(t0))*10;
            kpp=kpp+1;
            ini=0;
            alpha_all=0;
        end


 end
%  plot(day_cost)
cg(site_index)=sum(cost_record(day-9:day))/10;
% eva_record(site_index)=sum()
end

figure('Name','RL solution')
```

```
    yyaxis left
    plot(1:25,SoC_record(24*(day-1):end),'-','LineWidth',1)
    ylabel('SoC','FontSize',20) % left y-axis
    xlabel('Time (Hr)','FontSize',20)
    yyaxis right
    stairs(1:24,alpha(end-23:end),'LineWidth',5)
    set(gca,'fontsize',20)
    legend({'SoC','\sigma'},'FontSize',20)
    grid on
    ylabel('Traffic Shaping factor \sigma','FontSize',20)
%     figure
%     plot(day_cost)
% cg=sum(cost((day-1)*24:end));
% cg=w_alpha*sum(D)/n+w_soc*sum(SoC(t0))/t0;

% figure
% plot(SoC_record)
% figure
% stairs(alpha_record)
% figure
% plot(r0)
figure
bar3(squeeze(p_option(1,:,:)))
figure('Name','eva')
plot(cost_record)
figure('Name','eva_number')
plot(cg(2:end))
toc
end
```

Linear reward inaction policy updating function:

```
function
[r,p_option_out]=RL_reard_inauction_common(newstate,SoC_now,alpha,p_option,id
x,nstrategy,t,n,w_soc,r_cheat)

hold=0.8;
% if t~=23
    if SoC_now>hold && t>=20
        r=1;
    elseif SoC_now>0.3
        r=w_soc*(1/(1+exp(-(SoC_now)/(alpha))))+(1-w_soc)*(1/(1+exp(-
alpha*(SoC_now))));
    else
        r=(1-alpha)/10;
    end

%cost function known
% r=r_cheat;


%updating law
    a=0.05;
    p_option_out=zeros(1,nstrategy);
```

```
    for i=1:nstrategy
        if i==idx
%             squeeze(p_option(i));
            p_option_out(i)=p_option(i)+a*r*(1-p_option(i));
%             squeeze(p_option(i));
        elseif p_option(i)>0
            p_option_out(i)=p_option(i)-a*r*(p_option(i));
        else
            p_option_out(i)=0;
        end
    end
end
```

Load-ratio learning game code:

```
function bytest_game_ratio_adapting_2_1_fore_knowedge_smaller_samples
tic
day=5;
n=24*day;
SoC=zeros(1,n);
SoC(1)=0.8;
v=zeros(1,n);
alpha=zeros(1,n);
v(1)=(SoC(1)-0.5)/0.05+56;
Energy=3*25*1000;
w_soc=0.5;
w_alpha=1-w_soc;
batterystage=40;
cost=zeros(1,n);
D=zeros(1,n);
day_cost=zeros(1,day);
top_p=0.9;
amin=1;
zoom=1;
dump1=24*50;
dump2=24*15;
daygoal=0.8;
tcheckin=30;
phireal=1;
sitenumber=3;
cg=zeros(1,sitenumber);
alpha_self_record=zeros(1,sitenumber);
alphao_record=zeros(1,sitenumber);
% rate=ones(1,sitenumber)/sitenumber;
ratio_stage=5;
reward_record=zeros(1,day+1);

% rate=[0.4 0.6 0.5];
for site_index=3:sitenumber
    real_rate=ones(1,site_index)/site_index;
    count=1;
    initial_believe=0.5;
    rate=ones(1,site_index)/site_index;
% rate=ones(1,site_index);
p_ratio=(1-initial_believe)*ones(site_index,ratio_stage)/(ratio_stage-1);
```

```
ratio_picked=zeros(1,sitenumber);
for i=1:site_index
    done=0;
    for j=1:ratio_stage
        if 1/ratio_stage*j>=rate(i)&&done==0
            p_ratio(i,j)=initial_believe;
            done=1;
        end
    end
end

 for t0=1:n
    daymark=floor(t0/24)+1;
    t=mod(t0,24)+(mod(t0,24)==0)*24;
    add=0;
   for i=1:site_index
        pick=rand;
        p_roll=0;
        flag=0;
        for j=1:ratio_stage
            p_roll=p_roll+p_ratio(i,j);
            if p_roll>pick&&flag==0
                rate(i)=j/ratio_stage;
                ratio_picked(i)=j;
                flag=1;
            end
        end
%         [~,ratio_picked(i)]=max(p_ratio(i,:));
   end
%     end
%    if  t==1||t==9||t==17
%       load_shedding action numbers
action_n=4;

[alpha(t0),alphao,alpha_self]=gamesolver_ratio_adapt_2_1(v(t0),t,t0,w_alpha,w
_soc,batterystage,Energy,amin,daygoal,phireal,site_index,rate,real_rate,ratio
_stage);


   for i=1:site_index
   alpha_self_record(i)=alpha_self_record(i)+alpha_self(i)/24;
%    alphao_record(i)=alphao_record(i)+alphao(i)/24;
   end
    if t0>=dump1&&t0<=dump2
        phireal=0.5;
    end

[D(t0),cost(t0)]=evaluation(alpha(t0),t,batterystage,Energy,SoC(t0),w_soc,day
goal,phireal,add,0,0);
    SoC(t0+1)=(SoC(t0)*Energy-
integral(@(x)load2(x),t,t+1)*alpha(t0)+phireal*integral(@(y)solar(y),t,t+1))/
Energy;

if SoC(t0+1)>=1
   SoC(t0+1)=1;
end
```

```
if SoC(t0+1)<=0
   SoC(t0+1)=0;
end
v(t0+1)=(SoC(t0+1)-0.5)/0.05+56;

%load ratio adjusting process

reward_record(daymark)=reward_record(daymark)+self_reward_1_day(SoC(t0+1),t,w
_soc,daygoal,rate(1),alpha_self(1),alphao(1));
%          if  t==8||t==16||t==24
             for i=1:site_index
                     a=0.05;
                     r=cost(t0);
           %% reward function using local information
r=self_reward_1_day(SoC(t0+1),t,w_soc,daygoal,rate(i),alpha_self(i),alphao(i)
);
                     for j=1:ratio_stage
                         if j==ratio_picked(i)
                             p_ratio(i,j)=p_ratio(i,j)+a*r*(1-p_ratio(i,j));
                         elseif p_ratio(i,j)>(1-top_p)/(ratio_stage-1)
                             p_ratio(i,j)=p_ratio(i,j)-a*r*(p_ratio(i,j));
                         else
                             p_ratio(i,j)=(1-top_p)/(ratio_stage-1);
                         end
                         if p_ratio(i,j)>=top_p
                             p_ratio(i,j)=top_p;
                         end
                         if p_ratio(i,j)<=(1-top_p)/(ratio_stage-1)
                             p_ratio(i,j)=(1-top_p)/(ratio_stage-1);
                         end
                     end
             end
%          end

         if t==24
%              SoC(1)=SoC(t);
             if count<=zoom||count>=day-zoom
                 day_cost(count)=sum(cost(t0-23:t0));
             else
                 day_cost(count)=sum(cost(t0-23*zoom:t0))/zoom;
             end
             count=count+1;
             alpha_self_record=zeros(1,sitenumber);
             alphao_record=zeros(1,sitenumber);
         end
 end
%  plot(day_cost)
cg(site_index)=sum(cost)/day;
end
figure
plot(cg(2:end))
cg(sitenumber)
figure('Name','eva')
plot(day_cost)
figure('Name','RL+game solution')
     yyaxis left
```

```
    plot(n-23:n,SoC(n-23:n),'--','LineWidth',5)
    ylabel('SoC','FontSize',20) % left y-axis
    xlabel('Time (Hr)','FontSize',20)
    yyaxis right
%     plot(1:n,alpha,'LineWidth',5)
    plot(n-23:n,D(n-23:n)*10+31,'LineWidth',5)
    set(gca,'fontsize',20)
    legend({'SoC','PSNR'},'FontSize',20)
    grid on
    ylabel('PSNR','FontSize',20)

% figure
% plot(cost)
figure
bar3(squeeze(p_ratio))
toc
end
```

Bibliography

[1] A. Kwasinski, W. W. Weaver, P. L. Chapman, and P. T. Krein, "Telecommunications Power Plant Damage Assessment Caused by Hurricane Katrina - Site Survey and Follow-Up Results," in *INTELEC 06 - Twenty-Eighth International Telecommunications Energy Conference*, 2006, pp. 1-8.

[2] A. Kwasinski, "Effects of notable natural disasters from 2005 to 2011 on telecommunications infrastructure: Lessons from on-site damage assessments," in *2011 IEEE 33rd International Telecommunications Energy Conference (INTELEC)*, 2011, pp. 1-9.

[3] J. Blanchard, "Site reliability: Fuel cells add reliability to telecom sites," in *INTELEC 07 - 29th International Telecommunications Energy Conference*, 2007, pp. 563-567.

[4] A. Kwasinski and P. T. Krein, "Telecom power planning for natural and man-made disasters," in *INTELEC 07 - 29th International Telecommunications Energy Conference*, 2007, pp. 216-222.

[5] A. Kwasinski and A. Kwasinski, "Increasing sustainability and resiliency of cellular network infrastructure by harvesting renewable energy," *IEEE Communications Magazine,* vol. 53, pp. 110-116, 2015.

[6] A. Kwasinski, "Telecom power planning for natural disasters: Technology implications and alternatives to U.S. Federal Communications Commission's "Katrina Order"; in view of the effects of 2008 Atlantic Hurricane Season," in *INTELEC 2009 - 31st International Telecommunications Energy Conference*, 2009, pp. 1-6.

[7] A. Kwasinski and A. Kwasinski, "Role of energy storage in a microgrid for increased use of photovoltaic systems in wireless communication networks," in *2014 IEEE 36th International Telecommunications Energy Conference (INTELEC)*, 2014, pp. 1-8.

[8] W. Allen, D. W. Fletcher, and K. J. Fellhoelter, "Securing critical information and communication infrastructures through electric power grid independence," in *Telecommunications Energy Conference, 2003. INTELEC '03. The 25th International*, 2003, pp. 170-177.

[9] A. Kwasinski and A. Kwasinski, "Operational aspects and power architecture design for a microgrid to increase the use of renewable energy in wireless communication networks," in *2014 International Power Electronics Conference (IPEC-Hiroshima 2014 - ECCE ASIA)*, 2014, pp. 2649-2655.

[10] A. Kwasinski, "Telecommunications outside plant power infrastructure: Past performance and technological alternatives for improved resilience to hurricanes," in *INTELEC 2009 - 31st International Telecommunications Energy Conference*, 2009, pp. 1-6.

[11] R. Hu, A. Kwasinski, and A. Kwasinski, "Mixed strategy load management strategy for wireless communication network microgrid," pp. 1-8.

[12] R. Hu and A. Kwasinski, "Energy management for microgrids using a reinforcement learning algorithm " in *2018 IEEE Green Energy and Smart Systems Conference (IGESSC)*, 2018.

[13] A. Kwasinski and A. Kwasinski, "Integrating cross-layer LTE resources and energy management for increased powering of base stations from renewable energy," pp. 498-505.

[14] L. Maharjan, S. Inoue, H. Akagi, and J. Asakura, "State-of-Charge (SOC)-Balancing Control of a Battery Energy Storage System Based on a Cascade PWM Converter," *IEEE Transactions on Power Electronics,* vol. 24, pp. 1628-1636, 2009.

[15] Z. Chmiel and S. C. Bhattacharyya, "Analysis of off-grid electricity system at Isle of Eigg (Scotland): Lessons for developing countries," *Renewable Energy,* vol. 81, pp. 578-588, 2015.

[16] C. N. Resources, "The First Canadian Smart Remote Microgrid: Hartley Bay, BC."

[17] N. Hatziargyriou, *Microgrids: Architectures and Control*. New York, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2013.

[18] C. F. Chiasserini and R. R. Rao, "Improving battery performance by using traffic shaping techniques," *IEEE Journal on Selected Areas in Communications,* vol. 19, pp. 1385-1394, 2001.

[19] N. Mitrou, "Shaping of traffic streams through data spacing," *IEEE Communications Letters,* vol. 3, pp. 300-302, 1999.

[20] A. Andreoiu, K. Bhattacharya, and C. Canizares, "Pricing power system stabilisers using game theory," *IEE Proceedings - Generation, Transmission and Distribution,* vol. 152, pp. 780-786, 2005.

[21] F. Lian, A. Duel-Hallen, and A. Chakrabortty, "Ensuring economic fairness in wide-area control for power systems via game theory," in *2016 American Control Conference (ACC)*, 2016, pp. 3231-3236.

[22] G. C. Stamtsis and I. Erlich, "Use of cooperative game theory in power system fixed-cost allocation," *IEE Proceedings - Generation, Transmission and Distribution,* vol. 151, pp. 401-406, 2004.

[23] A. Mondal, S. Misra, and M. S. Obaidat, "Distributed Home Energy Management System With Storage in Smart Grid Using Game Theory," *IEEE Systems Journal,* pp. 1-10, 2015.

[24] S. Sofana Reka and V. Ramesh, "A demand response modeling for residential consumers in smart grid environment using game theory based energy scheduling algorithm," *Ain Shams Engineering Journal,* vol. 7, pp. 835-845, 2016.

[25] Citrix, "Citrix Mobile Analytics Report," 2015.

[26] X. Lu, K. Sun, J. M. Guerrero, J. C. Vasquez, and L. Huang, "State-of-Charge Balance Using Adaptive Droop Control for Distributed Energy Storage Systems in DC Microgrid Applications," *IEEE Transactions on Industrial Electronics,* vol. 61, pp. 2804-2815, 2014.

[27] "Converters allow for battery operation," *Electronics Times,* p. 37, 2001.

[28] T.-H. Wu, C.-S. Moo, Y.-C. Hsieh, and C.-Y. Juan, "Operation of battery power modules with bidirectional DC/DC converters," pp. 443-448.

[29] M. Chenine, I. A. Khatib, J. Ivanovski, V. Maden, and L. Nordström, "PMU traffic shaping in IP-based Wide Area communication," in *2010 5th International Conference on Critical Infrastructure (CRIS)*, 2010, pp. 1-6.

[30] V. M. M. Reddy P P, "Strategy learning for autonomous agents in smart grid markets," in *Twenty-second international joint conference on artificial intelligence*, 2005.

[31] R. W. Erickson and D. Maksimović, *Fundamentals of power electronics*, 2nd ed. Norwell, Mass: Kluwer Academic Publishers, 2001.

[32] C. Solar. (2012). *Solar PV in Cambridge*.

[33] X. Lin, A. G. Stefanopoulou, Y. Li, and R. D. Anderson, "State of charge estimation of cells in series connection by using only the total voltage measurement," in *2013 American Control Conference*, 2013, pp. 704-709.

[34] T. G. Josip Lorincz , Goran Petrovic, "Measurements and Modelling of Base Station Power Consumption under Real Traffic Loads," *Sensors,* vol. 12, pp. 4281-4310, 2012.

[35] Q. Zhang, Y.-b. Zhang, and S. Ma, "Effective global searching method based on radial basis function," in *Electric Information and Control Engineering (ICEICE), 2011 International Conference on*, 2011, pp. 350-353.

[36] J. Song, V. Krishnamurthy, A. Kwasinski, and R. Sharma, "Development of a Markov-Chain-Based Energy Storage Model for Power Supply Availability Assessment of Photovoltaic Generation Plants," *IEEE Transactions on Sustainable Energy,* vol. 4, pp. 491-500, 2013.

[37] A. Leon-Garcia, *Probability, statistics, and random processes for electrical engineering*, 3rd ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2008.

[38] F. Mohamed and H. Koivo, *System modeling and online optimal management of microgrid with battery storage* vol. 1, 2007.

[39] M. Maschler, E. Solan, and S. Zamir, *Game Theory*. Cambridge: Cambridge University Press, 2013.

[40] S. Jørgensen, M. Quincampoix, and T. L. Vincent. (2007). *Advances in dynamic game theory numerical methods, algorithms, and applications to ecology and economics*. Available: http://pitt.idm.oclc.org/login?url=http://link.springer.com/openurl?genre=book&isbn=978-0-8176-4399-7

[41] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences,* vol. 36, pp. 48-49, January 1, 1950.

[42] A. Piegat, "Bayes' Rule, Principle of Indifference, and Safe Distribution." vol. 5097, ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 661-670.

[43] R. D. Luce and H. Raiffa, *Games and Decisions: Introduction and Critical Survey*: Dover Publications, 1957.

[44] H. Karloff, *Linear programming*. Boston: Birkhäuser, 1991.

[45] M. Willem, *Minimax Theorems*: Birkhäuser Boston, 1997.

[46] H. ND, "Microgrid and energy management," *European transactions on electrical power,* pp. 1139-141, 2010.

[47] S. Govindan and R. Wilson, "A global Newton method to compute Nash equilibria," *Journal of Economic Theory,* vol. 110, pp. 65-86, 2003.

[48] R. Leo, R. S. Milton, and A. Kaviya, "Multi agent reinforcement learning based distributed optimization of solar microgrid," in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, 2014, pp. 1-7.

[49] H. Rui and W. W. Weaver, "Dc microgrid droop control based on battery state of charge balancing," in *2016 IEEE Power and Energy Conference at Illinois (PECI)*, 2016, pp. 1-8.

[50] R. G. Born. (2011). *The Effect of Sky Conditions on Solar Panel Power Output*. Available: https://www.vernier.com/innovate/the-effect-of-sky-conditions-on-solar-panel-power-output/

[51] S. Sastry and M. Bodson, *Adaptive control: stability, convergence, and robustness*: Prentice-Hall, Inc., 1989.

[52] S. Sastry, M. Bodson, and J. F. Bartram, "Adaptive Control: Stability, Convergence, and Robustness," *The Journal of the Acoustical Society of America,* vol. 88, pp. 588-589, 1990.

[53] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*: Prentice Hall, 1991.

[54] C. Böhringer, M. Finus, and C. Vogt, *Controlling global warming : perspectives from economics, game theory, and public choice*. Cheltenham, UK ; Northampton, MA: E. Elgar, 2002.

[55] F. S. Keller, *Learning: reinforcement theory* vol. PP13. New York U6 - ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info%3Asid%2Fsummon.serialssolutions.com&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=book&rft.title=Learning&rft.au=Keller%2C+Fred+S&rft.series=Studies+in+psychology&rft.date=1954&rft.pub=Random+House&rft.volume=PP13&rft.externalDocID=670125&paramdict=en-US U7 - Book: Random House, 1954.

[56] R. S. Sutton, A. G. Barto, and I. netLibrary, *Reinforcement learning: an introduction*. Cambridge, Mass: MIT Press, 1998.

[57] Y. Xi, L. Chang, M. Mao, P. Jin, N. Hatziargyriou, and H. Xu, "Q-learning algorithm based multi-agent coordinated control method for microgrids," pp. 1497-1504.

[58] H. Gintis, *The bounds of reason : game theory and the unification of the behavioral sciences*. Princeton: Princeton University Press, 2009.

[59] C. Stevanoni, F. Vallee, Z. D. Greve, and O. Deblecker, "On the use of game theory to study the planning and profitability of industrial microgrids connected to the distribution network," *CIRED - Open Access Proceedings Journal,* vol. 2017, pp. 2444-2448, 2017.

[60] C. D. a. C. H. Papadimitriou, *Three-player games are hard*, 2005.

[61] A. Pauly, "The Computational Complexity of Iterated Elimination of Dominated Strategies," *Theory of Computing Systems,* vol. 59, pp. 52-75, 2016.

[62] S. Homer and A. L. Selman, *Computability and complexity theory*, 2nd;2; ed. London;New York;: Springer, 2011.

[63] P. J. Smith, A. Sathyendran, and A. R. Murch, "Analysis of traffic distribution in cellular networks," pp. 2075-2079.

[64] P. P. Vergara, R. Torquato, and L. C. P. d. Silva, "Towards a real-time Energy Management System for a Microgrid using a multi-objective genetic algorithm," in *2015 IEEE Power & Energy Society General Meeting*, 2015, pp. 1-5.

[65] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an introduction*: Prentice-Hall, Inc., 1989.

[66] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms," *Neural Computation,* vol. 6, pp. 1185-1201, 1994.

[67] L. G. Telser, *Competition, collusion, and game theory*. Chicago,: Aldine·Atherton, 1972.

[68] V. Chamola, B. Krishnamachari, and B. Sikdar, "Green Energy and Delay Aware Downlink Power Control and User Association for Off-Grid Solar-Powered Base Stations," *IEEE Systems Journal,* vol. 12, pp. 2622-2633, 2018.

[69] M. Gardner, *The colossal book of mathematics : classic puzzles, paradoxes, and problems : number theory, algebra, geometry, probability, topology, game theory, infinity, and other topics of recreational mathematics*, 1st ed. New York: Norton, 2001.